# INTRODUÇÃO À PROGRAMAÇÃO COM PYTHON

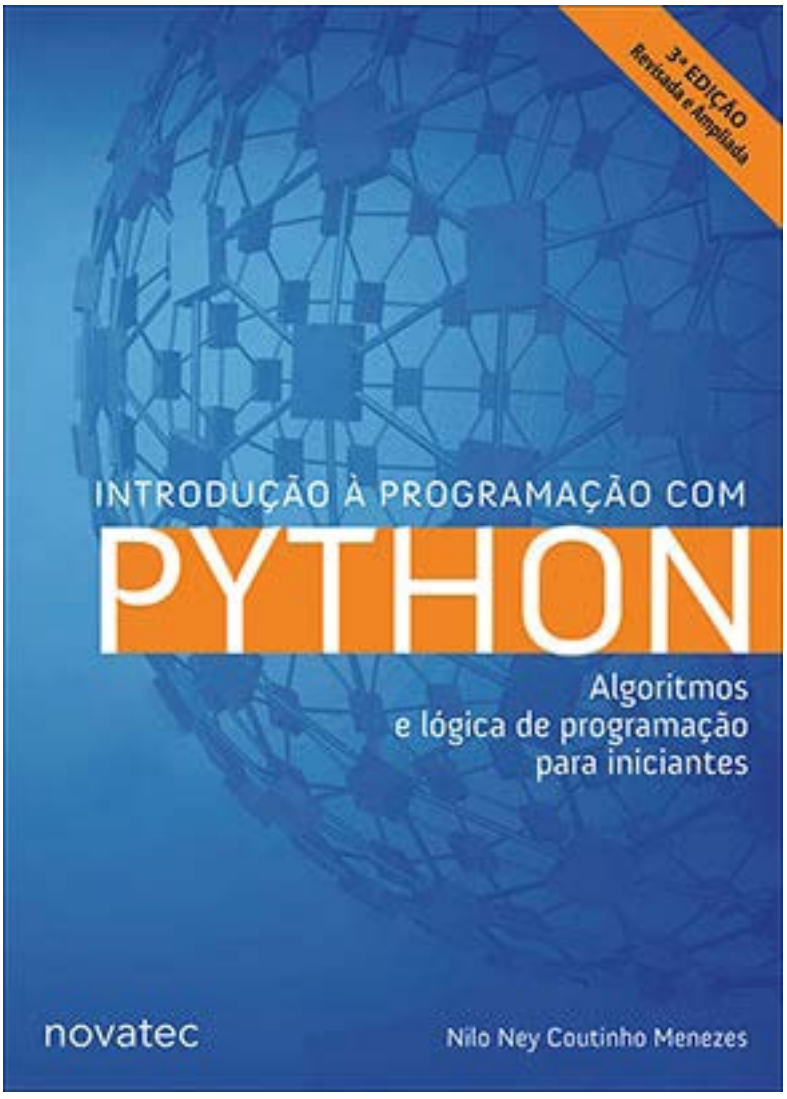
## EXERCÍCIOS RESOLVIDOS

3ªedição - atualização do 26 de março 2024

**Nilo Ney Coutinho Menezes**

livro_de_python@nilo.pro.br

Telegram: https://t.me/niloprog



# INDICE

# Exercícios Resolvidos

Nilo Ney Coutinho Menezes <livro_de_python@nilo.pro.br>

2024-03-25

## SOBRE O LIVRO

Este livro é orientado ao iniciante em programação. Os conceitos básicos de programação, como expressões, variáveis, repetições, decisões, listas, dicionários, conjuntos, funções, arquivos, classes, objetos e banco de dados com SQLite 3 são apresentados um a um com exemplos e exercícios. A obra visa a explorar a programação de computadores como ferramenta do dia a dia. Ela pode ser lida durante um curso de introdução à programação de computadores e usada como guia de estudo para autodidatas. Para aproveitamento pleno do conteúdo, apenas conhecimentos básicos de informática, como digitar textos, abrir e salvar arquivos, são suficientes. Todo software utilizado no livro pode ser baixado gratuitamente, sendo executado em Windows, Linux e Mac OS X.

Embora a linguagem Python (versão +3.7) seja muito poderosa e repleta de recursos modernos de programação, este livro não pretende ensinar a linguagem em si, mas ensinar a programar. Alguns recursos da linguagem não foram utilizados para privilegiar os exercícios de lógica de programação e oferecer uma preparação mais ampla ao leitor para outras linguagens. Essa escolha não impediu a apresentação de recursos poderosos da linguagem, e, embora o livro não seja fundamentalmente uma obra de referência, o leitor encontrará várias notas e explicações de características específicas do Python, além da lógica de programação.

Título: Introdução à Programação com Python

Autor: Nilo Ney Coutinho Menezes

Edição: Terceira

ISBN: 978-85-7522-718-3

Editora: Novatec

Ano: 2019

Páginas: 328

Para comprar o livro na Amazon, visite o link ou escaneie o qrcode com seu celular:

*Na Amazon: https://amzn.to/3Pjg4MZ*

Para comprar o livro na Editora Novatec, visite o link ou escaneie o qrcode com seu celular:

*Na Editora Novatec: http://www.novatec.com.br/livros/introducao-python-4ed/*

# INTRODUÇÃO

Este documento foi criado para disponibilizar todos os exercícios resolvidos do livro em um só arquivo. O site do livro pode ser acessado em https://python.nilo.pro.br ou pelo qr-code abaixo:

*Site do livro*

Se você não conhece o livro, visite o site web e tenha aceso às listagens, exercícios resolvidos, dúvidas e correções (errata). Para comprar o livro na Amazon, visite o link ou escaneie o qrcode com seu celular:

Lembre-se que os exercícios foram feitos para que você aprenda sozinho. Não olhe a resposta antes de tentar sozinho algumas vezes ;-D.

Não pode ser vendido.

# EXERCÍCIO 02-01

Converta as seguintes expressões matemáticas para que possam ser calculadas usando o interpretador Python. 10 + 20 × 30 42 ÷ 30 (94 + 2) × 6 - 1

```
# Para executar o cálculo e visualizar a resposta,
# copie e cole as linhas abaixo para a janela do interpretador,
# uma de cada vez.
# As respostas do exercício são as linhas abaixo:
10 + 20 * 30
4**2 / 30
(9**4 + 2) * 6 - 1
```

## EXERCÍCIO 02-02

Digite a seguinte expressão no interpretador: 10 % 3 * 10 ** 2 + 1 - 10 * 4 / 2 Tente resolver o mesmo cálculo, usando apenas lápis e papel. Observe como a prioridade das operações é importante.

```
# O resultado da expressão:
# 10 % 3 * 10 ** 2 + 1 - 10 * 4 / 2
# é 81.0
#
# Realizando o cálculo com as prioridades da página 39,
# efetuando apenas uma operação por linha,
# temos a seguinte ordem de cálculo:
# 0 --> 10 % 3 * 10 ** 2 + 1 - 10 * 4 / 2
# 1 --> 10 % 3 * 100     + 1 - 10 * 4 / 2
# 2 --> 1      * 100     + 1 - 10 * 4 / 2
# 3 -->          100     + 1 - 10 * 4 / 2
# 4 -->          100     + 1 - 40     / 2
# 5 -->          100     + 1 - 20
# 6 -->          101         - 20
# 7 -->                        81
#
# Se você estiver curioso(a) para saber por que o resultado
# é 81.0 e não 81, leia a seção 3.2, página 45.
# A operação de divisão sempre resulta em um número de ponto flutuante.
```

## EXERCÍCIO 02-03

Faça um programa que exiba seu nome na tela.

```
print("Escreva seu nome entre as aspas")
```

## EXERCÍCIO 02-04

Escreva um programa que exiba o resultado de 2a × 3b, em que a vale 3 e b vale 5.

```
a = 3
b = 5
print(2 * a * 3 * b)
```

## EXERCÍCIO 02-05

Escreva um programa que calcule a soma de três variáveis e imprima o resultado na tela.

```
a = 2
b = 3
c = 4
print(a + b + c)
```

## EXERCÍCIO 02-06

Modifique o Programa 2.2, de forma que ele calcule um aumento de 15% para um salário de R$ 750.

```
salário = 750
aumento = 15
print(salário + (salário * aumento / 100))
```

## EXERCÍCIO 03-01

Complete a tabela a seguir, marcando inteiro ou ponto flutuante dependendo do número apresentado.

```
    Número      Tipo numérico
    5      [ ] inteiro [ ] ponto flutuante
    5.0    [ ] inteiro [ ] ponto flutuante
    4.3    [ ] inteiro [ ] ponto flutuante
    -2     [ ] inteiro [ ] ponto flutuante
    100    [ ] inteiro [ ] ponto flutuante
    1.333  [ ] inteiro [ ] ponto flutuante
```

```
# inteiro
# ponto flutuante
# ponto flutuante
# inteiro
# inteiro
# ponto flutuante
```

## EXERCÍCIO 03-02

Complete a tabela a seguir, respondendo True ou False. Considere a = 4, b = 10, c = 5.0, d = 1 e f = 5.

```
Expressão Resultado        Expressão Resultado
a == c    [ ] True [ ] False   b > a     [ ] True [ ] False
a < b     [ ] True [ ] False   c >= f    [ ] True [ ] False
d > b     [ ] True [ ] False   f >= c    [ ] True [ ] False
c != f    [ ] True [ ] False   c <= c    [ ] True [ ] False
a == b    [ ] True [ ] False   c <= f    [ ] True [ ] False
c < d     [ ] True [ ] False
```

```
# False (a==c)
# True  (a<b)
# False (d>b)
# False (c!=f)
# False (a==b)
# False (c<d)
# True (b>a)
# True (c>=f)
# True (f>=c)
# True (c<=c)
# True (c<=f)
```

## EXERCÍCIO 03-03

Complete a tabela a seguir utilizando a = True, b = False e c = True.

```
Expressão Resultado          Expressão Resultado
a and a   [ ] True [ ] False    a or c    [ ] True [ ] False
b and b   [ ] True [ ] False    b or c    [ ] True [ ] False
not c     [ ] True [ ] False    c or a    [ ] True [ ] False
not b     [ ] True [ ] False    c or b    [ ] True [ ] False
not a     [ ] True [ ] False    c or c    [ ] True [ ] False
a and b   [ ] True [ ] False    b or b    [ ] True [ ] False
b and c   [ ] True [ ] False
```

```
# True (a and a)
# False (b and b)
# False (not c)
# True (not b)
# False (not a)
# False (a and b)
# False (b and c)
# True (a or c)
# True (b or c)
# True (a or c)
# True (b or c)
# True (c or a)
# True (c or b)
# True (c or c)
# False (b or b)
```

## EXERCÍCIO 03-04

Escreva uma expressão para determinar se uma pessoa deve ou não pagar imposto. Considere que pagam imposto pessoas cujo salário é maior que R$ 1.200,00.

```
salário > 1200
```

## EXERCÍCIO 03-05

Calcule o resultado da expressão A > B and C or D, utilizando os valores da tabela a seguir.

```
A  B C     D      Resultado
1  2 True  False
10 3 False False
5  1 True  True
```

```
# False
# False
# True
```

## EXERCÍCIO 03-06

Escreva uma expressão que será utilizada para decidir se um aluno foi ou não aprovado. Para ser aprovado, todas as médias do aluno devem ser maiores que 7. Considere que o aluno cursa apenas três matérias, e que a nota de cada uma está armazenada nas seguintes variáveis: matéria1, matéria2 e matéria3.

```
# Pelo enunciado:
matéria1 > 7 and matéria2 > 7 and matéria3 > 7
# Na prática, o aluno é aprovado se obtiver nota maior ou igual a média, logo:
matéria1 >= 7 and matéria2 >= 7 and matéria3 >= 7
```

## EXERCÍCIO 03-07

Faça um programa que peça dois números inteiros. Imprima a soma desses dois números na tela.

```
a = int(input("Digite o primeiro número:"))
b = int(input("Digite o segundo número:"))
print(a + b)
```

## EXERCÍCIO 03-08

Escreva um programa que leia um valor em metros e o exiba convertido em milímetros.

```
metros = float(input("Digite o valor em métros: "))
milímetros = metros * 1000
print("%10.3f metros equivalem a %10.3f milímetros." % (metros, milímetros))
```

## EXERCÍCIO 03-09

Escreva um programa que leia a quantidade de dias, horas, minutos e segundos do usuário. Calcule o total em segundos.

```
dias = int(input("Dias:"))
horas = int(input("Horas:"))
minutos = int(input("Minutos:"))
segundos = int(input("Segundos:"))
# Um minuto tem 60 segundos
# Uma hora tem 3600 (60 * 60) segundos
# Um dia tem 24 horas, logo 24 * 3600 segundos
total_em_segundos = dias * 24 * 3600 + horas * 3600 + minutos * 60 + segundos
print("Convertido em segundos é igual a %10d segundos." % total_em_segundos)
```

## EXERCÍCIO 03-10

Faça um programa que calcule o aumento de um salário. Ele deve solicitar o valor do salário e a porcentagem do aumento. Exiba o valor do aumento e do novo salário.

```
salário = float(input("Digite o salário atual:"))
p_aumento = float(input("Digite a porcentagem de aumento:"))
aumento = salário * p_aumento / 100
novo_salário = salário + aumento
print("Um aumento de %5.2f %% em um salário de R$ %7.2f" % (p_aumento, salário))
print("é igual a um aumento de R$ %7.2f" % aumento)
print("Resultando em um novo salário de R$ %7.2f" % novo_salário)
```

## EXERCÍCIO 03-11

Faça um programa que solicite o preço de uma mercadoria e o percentual de desconto. Exiba o valor do desconto e o preço a pagar.

```
preço = float(input("Digite o preço da mercadoria:"))
desconto = float(input("Digite o percentual de desconto:"))
valor_do_desconto = preço * desconto / 100
a_pagar = preço - valor_do_desconto
print("Um desconto de %5.2f %% em uma mercadoria de R$ %7.2f" % (desconto, preço))
print("vale R$ %7.2f." % valor_do_desconto)
print("O valor a pagar é de R$ %7.2f" % a_pagar)
```

## EXERCÍCIO 03-12

Escreva um programa que calcule o tempo de uma viagem de carro. Pergunte a distância a percorrer e a velocidade média esperada para a viagem.

```
distância = float(input("Digite a distância em km:"))
velocidade_média = float(input("Digite a velocidade média em km/h:"))
tempo = distância / velocidade_média
print("O tempo estimado é de %5.2f horas" % tempo)
# Opcional: imprimir o tempo em horas, minutos e segundos
tempo_s = int(tempo * 3600)  # convertemos de horas para segundos
horas = int(tempo_s / 3600)  # parte inteira
tempo_s = int(tempo_s % 3600)  # o resto
minutos = int(tempo_s / 60)
segundos = int(tempo_s % 60)
print("%05d:%02d:%02d" % (horas, minutos, segundos))
```

## EXERCÍCIO 03-13

Escreva um programa que converta uma temperatura digitada em °C em °F. A fórmula para essa conversão é:

$$F = \frac{9 \times C}{5} + 32$$

```
C = float(input("Digite a temperatura em °C:"))
F = (9 * C / 5) + 32
print("%5.2f°C é igual a %5.2f°F" % (C, F))
```

## EXERCÍCIO 03-14

Escreva um programa que pergunte a quantidade de km percorridos por um carro alugado pelo usuário, assim como a quantidade de dias pelos quais o carro foi alugado. Calcule o preço a pagar, sabendo que o carro custa R$ 60 por dia e R$ 0,15 por km rodado.

```
km = int(input("Digite a quantidade de quilometros percorridos:"))
dias = int(input("Digite quantos dias você ficou com o carro:"))
preço_por_dia = 60
preço_por_km = 0.15
preço_a_pagar = km * preço_por_km + dias * preço_por_dia
print("Total a pagar: R$ %7.2f" % preço_a_pagar)
```

## EXERCÍCIO 03-15

Escreva um programa para calcular a redução do tempo de vida de um fumante. Pergunte a quantidade de cigarros fumados por dia e quantos anos ele já fumou. Considere que um fumante perde 10 minutos de vida a cada cigarro, e calcule quantos dias de vida um fumante perderá. Exiba o total em dias.

```
cigarros_por_dia = int(input("Quantidade de cigarros por dia:"))
anos_fumando = float(input("Quantidade de anos fumando:"))
redução_em_minutos = anos_fumando * 365 * cigarros_por_dia * 10
# Um dia tem 24 x 60 minutos
redução_em_dias = redução_em_minutos / (24 * 60)
print("Redução do tempo de vida %8.2f dias." % redução_em_dias)
```

## EXERCÍCIO 04-01

Analise o Programa 4.1. Responda o que acontece se o primeiro e o segundo valores forem iguais? Explique.

```
# Se os valores forem iguais, nada será impresso.
# Isso acontece porque a > b e b > a são falsas quando a = b.
# Assim, nem o print de 2, nem o print de 3 serão executados, logo nada será
impresso.
```

## EXERCÍCIO 04-02

Escreva um programa que pergunte a velocidade do carro de um usuário. Caso ultrapasse 80 km/h, exiba uma mensagem dizendo que o usuário foi multado. Nesse caso, exiba o valor da multa, cobrando R$ 5 por km acima de 80 km/h.

```
velocidade = float(input("Digite a velocidade do seu carro:"))
if velocidade > 80:
    multa = (velocidade - 80) * 5
    print(f"Você foi multado em R$ {multa:7.2f}!")
if velocidade <= 80:
    print("Sua velocidade está ok, boa viagem!")
```

## EXERCÍCIO 04-03

Escreva um programa que leia três números e que imprima o maior e o menor.

```
a = int(input("Digite o primeiro valor:"))
b = int(input("Digite o segundo valor:"))
c = int(input("Digite o terceiro valor:"))
maior = a
if b > a and b > c:
    maior = b
if c > a and c >= b:
    maior = c
menor = a
if b < c and b < a:
    menor = b
if c <= b and c < a:
    menor = c
print(f"O menor número digitado foi {menor}")
print(f"O maior número digitado foi {maior}")
```

## EXERCÍCIO 04-04

Escreva um programa que pergunte o salário do funcionário e calcule o valor do aumento. Para salários superiores a R$ 1.250,00, calcule um aumento de 10%. Para os inferiores ou iguais, de 15%.

```
salário = float(input("Digite seu salário: "))
pc_aumento = 0.15
if salário > 1250:
    pc_aumento = 0.10
aumento = salário * pc_aumento
print(f"Seu aumento será de: R$ {aumento:7.2f}")
```

## EXERCÍCIO 04-05

Execute o Programa 4.4 e experimente alguns valores. Verifique se os resultados foram os mesmos do Programa 4.2.

```
# Sim, os resultados são os mesmos.
```

## EXERCÍCIO 04-06

Escreva um programa que pergunte a distância que um passageiro deseja percorrer em km. Calcule o preço da passagem, cobrando R$ 0,50 por km para viagens de até de 200 km, e R$ 0,45 para viagens mais longas.

```
distância = float(input("Digite a distância a percorrer: "))
if distância <= 200:
    passagem = 0.5 * distância
else:
    passagem = 0.45 * distância
print(f"Preço da passagem: R$ {passagem:7.2f}")
```

## EXERCÍCIO 04-07

Rastreie o Programa 4.6. Compare seu resultado ao apresentado na Tabela 4.2.

```
# O exercício consiste em rastrear o programa da listagem 4.7.
# O resultado deve ser o mesmo do apresentado na tabela 4.2.
# A técnica de rastreamento é apresentada na página 62,
# seção 3.6 Rastreamento.
```

## EXERCÍCIO 04-08

Escreva um programa que leia dois números e que pergunte qual operação você deseja realizar. Você deve poder calcular soma (+), subtração (-), multiplicação (*) e divisão (/). Exiba o resultado da operação solicitada.

```
a = float(input("Primeiro número:"))
b = float(input("Segundo número:"))
operação = input("Digite a operação a realizar (+,-,* ou /):")
if operação == "+":
    resultado = a + b
elif operação == "-":
    resultado = a - b
elif operação == "*":
    resultado = a * b
elif operação == "/":
    resultado = a / b
else:
```

```
    print("Operação inválida!")
    resultado = 0
print("Resultado: ", resultado)
```

## EXERCÍCIO 04-09

Escreva um programa para aprovar o empréstimo bancário para compra de uma casa. O programa deve perguntar o valor da casa a comprar, o salário e a quantidade de anos a pagar. O valor da prestação mensal não pode ser superior a 30% do salário. Calcule o valor da prestação como sendo o valor da casa a comprar dividido pelo número de meses a pagar.

```
valor = float(input("Digite o valor da casa: "))
salário = float(input("Digite o salário: "))
anos = int(input("Quantos anos para pagar: "))
meses = anos * 12
prestacao = valor / meses
if prestacao > salário * 0.3:
    print("Infelizmente você não pode obter o empréstimo")
else:
    print(f"Valor da prestação: R$ {prestacao:7.2f} Empréstimo OK")
```

## EXERCÍCIO 04-10

Escreva um programa que calcule o preço a pagar pelo fornecimento de energia elétrica. Pergunte a quantidade de kWh consumida e o tipo de instalação: R para residências, I para indústrias e C para comércios. Calcule o preço a pagar de acordo com a tabela a seguir.

```
+--------------------------------------+
|   Preço por tipo e faixa de consumo  |
+--------------------------------------+
| Tipo        | Faixa (kWh)   | Preço  |
+======================================+
| Residencial | Até 500       | R$ 0,40 |
|             | Acima de 500  | R$ 0,65 |
+--------------------------------------+
| Comercial   | Até 1000      | R$ 0,55 |
|             | Acima de 1000 | R$ 0,60 |
+--------------------------------------+
| Industrial  | Até 5000      | R$ 0,55 |
|             | Acima de 5000 | R$ 0,60 |
+--------------------------------------+
```

```
consumo = int(input("Consumo em kWh: "))
tipo = input("Tipo da instalação (R, C ou I): ")
if tipo == "R":
    if consumo <= 500:
        preço = 0.40
    else:
        preço = 0.65
```

```
elif tipo == "I":
    if consumo <= 5000:
        preço = 0.55
    else:
        preço = 0.60
elif tipo == "C":
    if consumo <= 1000:
        preço = 0.55
    else:
        preço = 0.60
else:
    preço = 0
    print("Erro ! Tipo de instalação desconhecido!")
custo = consumo * preço
print(f"Valor a pagar: R$ {custo:7.2f}")
```

## EXERCÍCIO 05-01

Modifique o programa para exibir os números de 1 a 100.

```
x = 1
while x <= 100:
    print(x)
    x = x + 1
```

## EXERCÍCIO 05-02

Modifique o programa para exibir os números de 50 a 100.

```
x = 50
while x <= 100:
    print(x)
    x = x + 1
```

## EXERCÍCIO 05-03

Faça um programa para escrever a contagem regressiva do lançamento de um foguete. O programa deve imprimir 10, 9, 8, …, 1, 0 e Fogo! na tela.

```
x = 10
while x >= 0:
    print(x)
    x = x - 1
print("Fogo!")
```

## EXERCÍCIO 05-04

Modifique o programa anterior para imprimir de 1 até o número digitado pelo usuário, mas, dessa vez, apenas os números ímpares.

```
fim = int(input("Digite o último número a imprimir:"))
x = 1
while x <= fim:
    print(x)
    x = x + 2
```

## EXERCÍCIO 05-05

Reescreva o programa anterior para escrever os 10 primeiros múltiplos de 3.

```
fim = 30
x = 3
while x <= fim:
    print(x)
    x = x + 3
```

## EXERCÍCIO 05-06

Altere o programa anterior para exibir os resultados no mesmo formato de uma tabuada: 2x1 = 2, 2x2 = 4, …

```
n = int(input("Tabuada de:"))
x = 1
while x <= 10:
    print(f"{n} x {x} = {n * x}")
    x = x + 1
```

## EXERCÍCIO 05-07

Modifique o programa anterior de forma que o usuário também digite o início e o fim da tabuada, em vez de começar com 1 e 10.

```
n = int(input("Tabuada de: "))
inicio = int(input("De: "))
fim = int(input("Até: "))
x = inicio
while x <= fim:
    print(f"{n} x {x} = {n * x}")
    x = x + 1
```

## EXERCÍCIO 05-08

Escreva um programa que leia dois números. Imprima o resultado da multiplicação do primeiro pelo segundo. Utilize apenas os operadores de soma e subtração para calcular o resultado. Lembre-se de que podemos entender a multiplicação de dois números como somas sucessivas de um deles. Assim, 4 × 5 = 5 + 5 + 5 + 5 = 4 + 4 + 4 + 4 + 4.

```
p = int(input("Primeiro número: "))
s = int(input("Segundo número: "))
x = 1
r = 0
while x <= s:
    r = r + p
    x = x + 1
print(f"{p} x {s} = {r}")
```

## EXERCÍCIO 05-09

Escreva um programa que leia dois números. Imprima a divisão inteira do primeiro pelo segundo, assim como o resto da divisão. Utilize apenas os operadores de soma e subtração para calcular o resultado. Lembre-se de que podemos entender o quociente da divisão de dois números como a quantidade de vezes que podemos retirar o divisor do dividendo. Logo, 20 ÷ 4 = 5, uma vez que podemos subtrair 4 cinco vezes de 20.

```
dividendo = int(input("Dividendo: "))
divisor = int(input("Divisor: "))
quociente = 0
x = dividendo
while x >= divisor:
    x = x - divisor
    quociente = quociente + 1
resto = x
print(f"{dividendo} / {divisor} = {quociente} (quociente) {resto} (resto)")
```

## EXERCÍCIO 05-10

Modifique o programa anterior para que aceite respostas com letras maiúsculas e minúsculas em todas as questões.

```
pontos = 0
questão = 1
while questão <= 3:
    resposta = input(f"Resposta da questão {questão}: ")
    if questão == 1 and (resposta == "b" or resposta == "B"):
        pontos = pontos + 1
    if questão == 2 and (resposta == "a" or resposta == "A"):
        pontos = pontos + 1
    if questão == 3 and (resposta == "d" or resposta == "D"):
        pontos = pontos + 1
```

```
    questão += 1
print(f"O aluno fez {pontos} ponto(s)")
```

## EXERCÍCIO 05-11

Escreva um programa que pergunte o depósito inicial e a taxa de juros de uma poupança. Exiba os valores mês a mês para os 24 primeiros meses. Escreva o total ganho com juros no período.

```
depósito = float(input("Depósito inicial: "))
taxa = float(input("Taxa de juros (Ex.: 3 para 3%): "))
mês = 1
saldo = depósito
while mês <= 24:
    saldo = saldo + (saldo * (taxa / 100))
    print(f"Saldo do mês {mês} é de R${saldo:5.2f}.")
    mês = mês + 1
print(f"O ganho obtido com os juros foi de R${saldo-depósito:8.2f}.")
```

## EXERCÍCIO 05-12

Altere o programa anterior de forma a perguntar também o valor depositado mensalmente. Esse valor será depositado no início de cada mês, e você deve considerá-lo para o cálculo de juros do mês seguinte.

```
depósito = float(input("Depósito inicial: "))
taxa = float(input("Taxa de juros (Ex.: 3 para 3%): "))
investimento = float(input("Depósito mensal: "))
mês = 1
saldo = depósito
while mês <= 24:
    saldo = saldo + (saldo * (taxa / 100)) + investimento
    print(f"Saldo do mês {mês} é de R${saldo:5.2f}.")
    mês = mês + 1
print(f"O ganho obtido com os juros foi de R${saldo-depósito:8.2f}.")
```

## EXERCÍCIO 05-13

Escreva um programa que pergunte o valor inicial de uma dívida e o juro mensal. Pergunte também o valor mensal que será pago. Imprima o número de meses para que a dívida seja paga, o total pago e o total de juros pago.

```
dívida = float(input("Dívida: "))
taxa = float(input("Juros (Ex.: 3 para 3%): "))
pagamento = float(input("Pagamento mensal:"))
mês = 1
if dívida * (taxa / 100) > pagamento:
    print(
        "Sua dívida não será paga nunca, pois os juros são superiores ao pagamento
```

```
mensal."
    )
else:
    saldo = dívida
    juros_pago = 0
    while saldo > pagamento:
        juros = saldo * taxa / 100
        saldo = saldo + juros - pagamento
        juros_pago = juros_pago + juros
        print(f"Saldo da dívida no mês {mês} é de R${saldo:6.2f}.")
        mês = mês + 1
    print(f"Para pagar uma dívida de R${dívida:8.2f}, a {taxa:5.2f} % de juros,")
    print(
        f"você precisará de {mês - 1} meses, pagando um total de R${juros_
pago:8.2f} de juros."
    )
    print(f"No último mês, você teria um saldo residual de R${saldo:8.2f} a
pagar.")
```

## EXERCÍCIO 05-14

Escreva um programa que leia números inteiros do teclado. O programa deve ler os números até que o usuário digite 0 (zero). No final da execução, exiba a quantidade de números digitados, assim como a soma e a média aritmética.

```
soma = 0
quantidade = 0
while True:
    n = int(input("Digite um número inteiro: "))
    if n == 0:
        break
    soma = soma + n
    quantidade = quantidade + 1
print("Quantidade de números digitados:", quantidade)
print("Soma: ", soma)
print(f"Média: {soma/quantidade:10.2f}")
```

## EXERCÍCIO 05-15

Escreva um programa para controlar uma pequena máquina registradora. Você deve solicitar ao usuário que digite o código do produto e a quantidade comprada. Utilize a tabela de códigos a seguir para obter o preço de cada produto:

```
Código Preço
1      0,50
2      1,00
3      4,00
5      7,00
9      8,00
```

Seu programa deve exibir o total das compras depois que o usuário digitar 0. Qualquer outro código deve gerar a mensagem de erro "Código inválido".

```
apagar = 0
while True:
    código = int(input("Código da mercadoria (0 para sair): "))
    preço = 0
    if código == 0:
        break
    elif código == 1:
        preço = 0.50
    elif código == 2:
        preço = 1.00
    elif código == 3:
        preço = 4.00
    elif código == 5:
        preço = 7.00
    elif código == 9:
        preço = 8.00
    else:
        print("Código inválido!")
    if preço != 0:
        quantidade = int(input("Quantidade: "))
        apagar = apagar + (preço * quantidade)
print(f"Total a pagar R${apagar:8.2f}")
```

## EXERCÍCIO 05-16

Execute o Programa 5.1 para os seguintes valores: 501, 745, 384, 2, 7 e 1.

```
# O programa deve funcionar normalmente com os valores solicitados pelo exercício.
```

## EXERCÍCIO 05-17

O que acontece se digitarmos 0 (zero) no valor a pagar?

```
# O programa pára logo após imprimir a quantidade de cédulas de R$50,00
```

## EXERCÍCIO 05-18

Modifique o programa para também trabalhar com notas de R$ 100.

```
valor = int(input("Digite o valor a pagar:"))
cédulas = 0
atual = 100
apagar = valor
while True:
    if atual <= apagar:
```

```
        apagar -= atual
        cédulas += 1
    else:
        print(f"{cédulas} cédula(s) de R${atual}")
        if apagar == 0:
            break
        elif atual == 100:
            atual = 50
        elif atual == 50:
            atual = 20
        elif atual == 20:
            atual = 10
        elif atual == 10:
            atual = 5
        elif atual == 5:
            atual = 1
        cédulas = 0
```

## EXERCÍCIO 05-19

Modifique o programa para aceitar valores decimais, ou seja, também contar moedas de 0,01, 0,02, 0,05, 0,10 e 0,50

```
# Atenção: alguns valores não serão calculados corretamente
# devido a problemas com arredondamento e da representação de 0.01
# em ponto flutuante. Uma alternativa é multiplicar todos os valores
# por 100 e realizar todos os cálculos com números inteiros.

valor = float(input("Digite o valor a pagar:"))
cédulas = 0
atual = 100
apagar = valor
while True:
    if atual <= apagar:
        apagar -= atual
        cédulas += 1
    else:
        if atual >= 1:
            print(f"{cédulas} cédula(s) de R${atual}")
        else:
            print(f"{cédulas} moeda(s) de R${atual:5.2f}")
        if apagar < 0.01:
            break
        elif atual == 100:
            atual = 50
        elif atual == 50:
            atual = 20
        elif atual == 20:
            atual = 10
        elif atual == 10:
            atual = 5
```

```
        elif atual == 5:
            atual = 1
        elif atual == 1:
            atual = 0.50
        elif atual == 0.50:
            atual = 0.10
        elif atual == 0.10:
            atual = 0.05
        elif atual == 0.05:
            atual = 0.02
        elif atual == 0.02:
            atual = 0.01
        cédulas = 0
```

## EXERCÍCIO 05-20

O que acontece se digitarmos 0,001 no programa anterior? Caso ele não funcione, altere-o de forma a corrigir o problema.

```
# Como preparamos o programa para valores menores que 0.01,
# este pára de executar após imprimir 0 cédula(s) de R$100.
# Ver também a nota do exercício 05.19 para compreender
# melhor este problema.
```

## EXERCÍCIO 05-21

Reescreva o Programa 5.1 de forma a continuar executando até que o valor digitado seja 0. Utilize repetições aninhadas.

```
while True:
    valor = int(input("Digite o valor a pagar:"))
    if valor == 0:
        break
    cédulas = 0
    atual = 50
    apagar = valor
    while True:
        if atual <= apagar:
            apagar -= atual
            cédulas += 1
        else:
            print(f"{cédulas} cédula(s) de R${atual}")
            if apagar == 0:
                break
            if atual == 50:
                atual = 20
            elif atual == 20:
                atual = 10
            elif atual == 10:
```

```
                atual = 5
            elif atual == 5:
                atual = 1
            cédulas = 0
```

## EXERCÍCIO 05-22

Escreva um programa que exiba uma lista de opções (menu): adição, subtração, divisão, multiplicação e sair. Imprima a tabuada da operação escolhida. Repita até que a opção saída seja escolhida.

```
while True:
    print("""

Menu
\----
1 - Adição
2 - Subtração
3 - Divisão
4 - Multiplicação
5 - Sair

""")
    opção = int(input("Escolha uma opção:"))
    if opção == 5:
        break
    elif opção >= 1 and opção < 5:
        n = int(input("Tabuada de:"))
        x = 1
        while x <= 10:
            if opção == 1:
                print(f"{n} + {x} = {n + x}")
            elif opção == 2:
                print(f"{n} - {x} = {n - x}")
            elif opção == 3:
                print(f"{n} / {x} = {n / x:5.4f}")
            elif opção == 4:
                print(f"{n} x {x} = {n * x}")
            x = x + 1
    else:
        print("Opção inválida!")
```

## EXERCÍCIO 05-23

Escreva um programa que leia um número e verifique se é ou não um número primo. Para fazer essa verificação, calcule o resto da divisão do número por 2 e depois por todos os números ímpares até o número lido. Se o resto de uma dessas divisões for igual a zero, o número não é primo. Observe que 0 e 1 não são primos e que 2 é o único número primo que é par.

```
n = int(input("Digite um número:"))
```

```
if n < 0:
    print("Número inválido. Digite apenas valores positivos")
if n == 0 or n == 1:
    print(f"{n} é um caso especial.")
else:
    if n == 2:
        print("2 é primo")
    elif n % 2 == 0:
        print(f"{n} não é primo, pois 2 é o único número par primo.")
    else:
        x = 3
        while x < n:
            if n % x == 0:
                break
            x = x + 2
        if x == n:
            print(f"{n} é primo")
        else:
            print(f"{n} não é primo, pois é divisível por {x}")
```

## EXERCÍCIO 05-24

Modifique o programa anterior de forma a ler um número n. Imprima os n primeiros números primos.

```
n = int(input("Digite um número: "))
if n < 0:
    print("Número inválido. Digite apenas valores positivos")
else:
    if n >= 1:
        print("2")
        p = 1
        y = 3
        while p < n:
            x = 3
            while x < y:
                if y % x == 0:
                    break
                x = x + 2
            if x == y:
                print(x)
                p = p + 1
            y = y + 2
```

## EXERCÍCIO 05-25

Escreva um programa que calcule a raiz quadrada de um número. Utilize o método de Newton para obter um resultado aproximado. Sendo n o número a obter a raiz quadrada, considere a base b=2. Calcule p usando a fórmula p=(b+(n/b))/2. Agora, calcule o quadrado de p. A cada passo, faça b=p e recalcule p usando a fórmula apresentada. Pare quando a diferença absoluta entre n e o quadrado de p for menor que 0,0001.

```
# Atenção: na primeira edição do livro, a fórmula foi publicada errada.
# A fórmula correta é p = ( b + ( n / b ) ) / 2
# A função abs foi utilizada para calcular o valor absoluto de um número,
# ou seja, seu valor sem sinal.
# Exemplos: abs(1) retorna 1 e abs(-1) retorna 1

n = float(input("Digite um número para encontrar a sua raiz quadrada: "))
b = 2
while abs(n - (b * b)) > 0.00001:
    p = (b + (n / b)) / 2
    b = p
print(f"A raiz quadrada de {n} é aproximadamente {p:8.4f}")
```

## EXERCÍCIO 05-26

Escreva um programa que calcule o resto da divisão inteira entre dois números. Utilize apenas as operações de soma e subtração para calcular o resultado.

```
# Atenção: este exercício é muito semelhante a exercício 5.08
dividendo = int(input("Dividendo: "))
divisor = int(input("Divisor: "))
quociente = 0
x = dividendo
while x >= divisor:
    x = x - divisor
    quociente = quociente + 1
resto = x
print(f"O resto de {dividendo} / {divisor} é {resto}")
```

## EXERCÍCIO 05-27-A

Escreva um programa que verifique se um número é palíndromo. Um número é palíndromo se continua o mesmo caso seus dígitos sejam invertidos. Exemplos: 454, 10501

```
# Para resolver este problema, podemos usar strings, apresentadas na seção 3.4 do
livro
# Veja que estamos lendo o número sem convertê-lo para int ou float,
# desta forma o valor de s será uma string
s = input("Digite o número a verificar, sem espaços:")
i = 0
f = len(s) - 1  # posição do último caracter da string
```

```
while f > i and s[i] == s[f]:
    f = f - 1
    i = i + 1
if s[i] == s[f]:
    print(f"{s} é palíndromo")
else:
    print(f"{s} não é palíndromo")
```

## EXERCÍCIO 05-27-B

Escreva um programa que verifique se um número é palíndromo. Um número é palíndromo se continua o mesmo caso seus dígitos sejam invertidos. Exemplos: 454, 10501

```
# Exercício 5.27
# Solução alternativa, usando apenas inteiros
n = int(input("Digite o número a verificar:"))
# Com n é um número inteiro, vamos calcular sua
# quantidade de dígitos, encontrado a primeira
# potência de 10, superior a n.
# Exemplo: 341 - primeira potência de 10 maior: 1000 = 10 ^ 4
# Utilizaremos 4 e não 3 para possibilitar o tratamento de números
# com um só dígito. O ajuste é feito nas fórmulas abaixo
q = 0
while 10**q < n:
    q = q + 1
i = q
f = 0
nf = ni = n  # Aqui nós copiamos n para ni e nf
pi = pf = 0  # e fazemos pi = pf (para casos especiais)
while i > f:
    pi = int(ni / (10 ** (i - 1)))  # Dígito mais à direita
    pf = nf % 10  # Dígito mais à esquerda
    if pi != pf:  # Se são diferentes, saímos
        break
    f = f + 1  # Passamos para o próximo dígito a esqueda
    i = i - 1  # Passamos para o dígito a direita seguinte
    ni = ni - (pi * (10**i))  # Ajustamos ni de forma a retirar o dígito anterior
    nf = int(nf / 10)  # Ajustamos nf para retirar o último dígito

if pi == pf:
    print(f"{n} é palíndromo")
else:
    print(f"{n} não é palíndromo")
```

## EXERCÍCIO 06-01

Modifique o Programa 6.2 para ler 7 notas em vez de 5.

```
notas = [0, 0, 0, 0, 0, 0, 0]  # Ou [0] * 7
soma = 0
x = 0
while x < 7:
    notas[x] = float(input(f"Nota {x}:"))
    soma += notas[x]
    x += 1
x = 0
while x < 7:
    print(f"Nota {x}: {notas[x]:6.2f}")
    x += 1
print(f"Média: {soma/x:5.2f}")
```

## EXERCÍCIO 06-02

Faça um programa que leia duas listas e que gere uma terceira com os elementos das duas primeiras.

```
primeira = []
segunda = []
while True:
    e = int(input("Digite um valor para a primeira lista (0 para terminar): "))
    if e == 0:
        break
    primeira.append(e)
while True:
    e = int(input("Digite um valor para a segunda lista (0 para terminar): "))
    if e == 0:
        break
    segunda.append(e)
terceira = primeira[:]  # Copia os elementos da primeira lista
terceira.extend(segunda)
x = 0
while x < len(terceira):
    print(f"{x}: {terceira[x]}")
    x = x + 1
```

## EXERCÍCIO 06-03

Faça um programa que percorra duas listas e gere uma terceira sem elementos repetidos.

```
primeira = []
segunda = []
while True:
    e = int(input("Digite um valor para a primeira lista (0 para terminar):"))
    if e == 0:
```

```
        break
    primeira.append(e)
while True:
    e = int(input("Digite um valor para a segunda lista (0 para terminar):"))
    if e == 0:
        break
    segunda.append(e)
terceira = []
# Aqui vamos criar uma outra lista, com os elementos da primeira
# e da segunda. Existem várias formas de resolver este exercício.
# Nesta solução, vamos pesquisar os valores a inserir na terceira
# lista. Se não existirem, adicionaremos à terceira. Caso contrário,
# não copiaremos, evitando assim os repetidos.
duas_listas = primeira[:]
duas_listas.extend(segunda)
x = 0
while x < len(duas_listas):
    y = 0
    while y < len(terceira):
        if duas_listas[x] == terceira[y]:
            break
        y = y + 1
    if y == len(terceira):
        terceira.append(duas_listas[x])
    x = x + 1
x = 0
while x < len(terceira):
    print(f"{x}: {terceira[x]}")
    x = x + 1
```

## EXERCÍCIO 06-04

O que acontece quando não verificamos se a lista está vazia antes de chamarmos o método pop?

```
# Se não verificarmos que a lista está vazia antes de charmos pop(),
# o programa pára com uma mensagem de erro, informando que tentamos
# retirar um elemento de uma lista vazia.
# A verificação é necessária para controlar este erro e assegurar
# o bom funcionamento do programa.
```

## EXERCÍCIO 06-05

Altere o Programa 6.7 de forma a poder trabalhar com vários comandos digitados de uma só vez. Atualmente, apenas um comando pode ser inserido por vez. Altere-o de forma a considerar operação como uma string.

Exemplo: FFFAAAS significaria três chegadas de novos clientes, três atendimentos e, finalmente, a saída do programa.

```
último = 10
fila = list(range(1, último + 1))
while True:
    print(f"\nExistem {len(fila)} clientes na fila")
    print("Fila atual:", fila)
    print("Digite F para adicionar um cliente ao fim da fila,")
    print("ou A para realizar o atendimento. S para sair.")
    operação = input("Operação (F, A ou S):")
    x = 0
    sair = False
    while x < len(operação):
        if operação[x] == "A":
            if len(fila) > 0:
                atendido = fila.pop(0)
                print(f"Cliente {atendido} atendido")
            else:
                print("Fila vazia! Ninguém para atender.")
        elif operação[x] == "F":
            último += 1  # Incrementa o ticket do novo cliente
            fila.append(último)
        elif operação[x] == "S":
            sair = True
            break
        else:
            print(
                f"Operação inválida: {operação[x]} na posição {x}! Digite apenas
F, A ou S!"
            )
        x = x + 1
    if sair:
        break
```

## EXERCÍCIO 06-06

Modifique o programa para trabalhar com duas filas. Para facilitar seu trabalho, considere o comando A para atendimento da fila 1; e B, para atendimento da fila 2. O mesmo para a chegada de clientes: F para fila 1; e G, para fila 2.

```
último = 0
fila1 = []
fila2 = []
while True:
    print(f"\nExistem {len(fila1)} clientes na fila 1 e {len(fila2)} na fila 2.")
    print("Fila 1 atual:", fila1)
    print("Fila 2 autal:", fila2)
    print("Digite F para adicionar um cliente ao fim da fila 1 (ou G para fila 2),")
    print("ou A para realizar o atendimento a fila 1 (ou B para fila 2")
    print("S para sair.")
    operação = input("Operação (F, G, A, B ou S):")
    x = 0
    sair = False
```

```
    while x < len(operação):
        # Aqui vamos usar fila como referência a fila 1
        # ou a fila 2, dependendo da operação.
        if operação[x] == "A" or operação[x] == "F":
            fila = fila1
        else:
            fila = fila2
        if operação[x] == "A" or operação[x] == "B":
            if len(fila) > 0:
                atendido = fila.pop(0)
                print(f"Cliente {atendido} atendido")
            else:
                print("Fila vazia! Ninguém para atender.")
        elif operação[x] == "F" or operação[x] == "G":
            último += 1  # Incrementa o ticket do novo cliente
            fila.append(último)
        elif operação[x] == "S":
            sair = True
            break
        else:
            print(
                f"Operação inválida: {operação[x]} na posição {x}! Digite apenas
F, A ou S!"
            )
        x = x + 1
    if sair:
        break
```

## EXERCÍCIO 06-07

Faça um programa que leia uma expressão com parênteses. Usando pilhas, verifique se os parênteses foram abertos e fechados na ordem correta.

Exemplo:

```
(()) OK
()()(()()) OK
()) Erro
```

Você pode adicionar elementos à pilha sempre que encontrar abre parênteses e desempilhá-la a cada fecha parênteses. Ao desempilhar, verifique se o topo da pilha é um abre parênteses. Se a expressão estiver correta, sua pilha estará vazia no final.

```
expressão = input("Digite a sequência de parênteses a validar:")
x = 0
pilha = []
while x < len(expressão):
    if expressão[x] == "(":
        pilha.append("(")
    if expressão[x] == ")":
        if len(pilha) > 0:
```

```
            topo = pilha.pop(-1)
        else:
            pilha.append(")")  # Força a mensagem de erro
            break
    x = x + 1
if len(pilha) == 0:
    print("OK")
else:
    print("Erro")
```

## EXERCÍCIO 06-08

Modifique o primeiro exemplo (Programa 6.9) de forma a realizar a mesma tarefa, mas sem utilizar a variável achou. Dica: observe a condição de saída do while.

```
L = [15, 7, 27, 39]
p = int(input("Digite o valor a procurar:"))
x = 0
while x < len(L):
    if L[x] == p:
        break
    x += 1
if x < len(L):
    print(f"{p} achado na posição {x}")
else:
    print(f"{p} não encontrado")
```

## EXERCÍCIO 06-09

Modifique o exemplo para pesquisar dois valores. Em vez de apenas p, leia outro valor v que também será procurado. Na impressão, indique qual dos dois valores foi achado primeiro.

```
L = [15, 7, 27, 39]
p = int(input("Digite o valor a procurar (p): "))
v = int(input("Digite o outro valor a procurar (v): "))
x = 0
achouP = False
achouV = False
primeiro = 0
while x < len(L):
    if L[x] == p:
        achouP = True
        if not achouV:
            primeiro = 1
    if L[x] == v:
        achouV = True
        if not achouP:
            primeiro = 2
    x += 1
```

```
if achouP:
    print(f"p: {p} encontrado")
else:
    print(f"p: {p} não encontrado")
if achouV:
    print(f"v: {v} encontrado")
else:
    print(f"v: {v} não encontrado")
if primeiro == 1:
    print("p foi achado antes de v")
elif primeiro == 2:
    print("v foi achado antes de p")
```

## EXERCÍCIO 06-10

Modifique o programa do Exercício 6.9 de forma a pesquisar p e v em toda a lista e informando o usuário a posição onde p e a posição onde v foram encontrados.

```
L = [15, 7, 27, 39]
p = int(input("Digite o valor a procurar (p):"))
v = int(input("Digite o outro valor a procurar (v):"))
x = 0
achouP = -1  # Aqui -1 indica que ainda não encontramos o valor procurado
achouV = -1
primeiro = 0
while x < len(L):
    if L[x] == p:
        achouP = x
    if L[x] == v:
        achouV = x
    x += 1
if achouP != -1:
    print(f"p: {p} encontrado na posição {achouP}")
else:
    print(f"p: {p} não encontrado")
if achouV != -1:
    print(f"v: {v} encontrado na posição {achouV}")
else:
    print(f"v: {v} não encontrado")
# Verifica se ambos foram encontrados
if achouP != -1 and achouV != -1:
    # como achouP e achouV guardam a posição onde foram encontrados
    if achouP <= achouV:
        print("p foi achado antes de v")
    else:
        print("v foi achado antes de p")
```

## EXERCÍCIO 06-11

Modifique o Programa 6.6 usando for. Explique por que nem todos os while podem ser transformados em for.

```
L = []
while True:
    n = int(input("Digite um número (0 sai):"))
    if n == 0:
        break
    L.append(n)
for e in L:
    print(e)
# O primeiro while não pôde ser convertido em for porque
# o número de repetições é desconhecido no início.
```

## EXERCÍCIO 06-12

Altere o Programa 6.11 de forma a imprimir o menor elemento da lista.

```
L = [4, 2, 1, 7]
mínimo = L[0]
for e in L:
    if e < mínimo:
        mínimo = e
print(mínimo)
```

## EXERCÍCIO 06-13

A lista de temperaturas de Mons, na Bélgica, foi armazenada na lista T = [ -10, -8, 0, 1, 2, 5, -2, -4]. Faça um programa que imprima a menor e a maior temperatura, assim como a temperatura média.

```
T = [-10, -8, 0, 1, 2, 5, -2, -4]
mínima = T[
    0
]  # A escolha do primeiro elemento é arbitrária, poderia ser qualquer elemento
válido
máxima = T[0]
soma = 0
for e in T:
    if e < mínima:
        mínima = e
    if e > máxima:
        máxima = e
    soma = soma + e
print(f"Temperatura máxima: {máxima} °C")
print(f"Temperatura mínima: {mínima} °C")
print(f"Temperatura média: {soma / len(T)} °C")
```

## EXERCÍCIO 06-14

O que acontece quando a lista já está ordenada? Rastreie o Programa 6.20, mas com a lista L = [1, 2, 3, 4, 5].

```
# Se a lista já estiver ordenada, nenhum elemento é maior que o elemento seguinte.
# Desta forma, após a primeira verificação de todos os elementos,
# o loop interno é interrompido pela condição de (9).
```

## EXERCÍCIO 06-15

O que acontece quando dois valores são iguais? Rastreie o Programa 6.20, mas com a lista L = [3, 3, 1, 5, 4].

```
# Como utilizamos o método de bolhas, na primeira verificação, 3, 3 são considerados
como na ordem correta.
# Quanto verificamos o segundo 3 com 1, ocorre uma troca.
# O mesmo vai ocorrer com o primeiro 3, mas apenas na próxima repetição. Veja que
o 1 subiu para a primeira posição
# como uma bolha de ar dentro d'água.
```

## EXERCÍCIO 06-16

Modifique o Programa 6.20 para ordenar a lista em ordem decrescente. L = [1, 2, 3, 4, 5] deve ser ordenada como L = [5, 4, 3, 2, 1].

```
L = [1, 2, 3, 4, 5]
fim = 5
while fim > 1:
    trocou = False
    x = 0
    while x < (fim - 1):
        if L[x] < L[x + 1]:  # Apenas a condição de verificação foi alterada
            trocou = True
            temp = L[x]
            L[x] = L[x + 1]
            L[x + 1] = temp
        x += 1
    if not trocou:
        break
    fim -= 1
for e in L:
    print(e)
```

## EXERCÍCIO 06-17

Altere o Programa 6.22 de forma a solicitar ao usuário o produto e a quantidade vendida. Verifique se o nome do produto digitado existe no dicionário, e só então efetue a baixa em estoque.

```
estoque = {
    "tomate": [1000, 2.30],
    "alface": [500, 0.45],
    "batata": [2001, 1.20],
    "feijão": [100, 1.50],
}
total = 0
print("Vendas:\n")
while True:
    produto = input("Nome do produto (fim para sair):")
    if produto == "fim":
        break
    if produto in estoque:
        quantidade = int(input("Quantidade:"))
        if quantidade <= estoque[produto][0]:
            preço = estoque[produto][1]
            custo = preço * quantidade
            print(f"{produto:12s}: {quantidade:3d} x {preço:6.2f} = {custo:6.2f}")
            estoque[produto][0] -= quantidade
            total += custo
        else:
            print("Quantidade solicitada não disponível")
    else:
        print("Nome de produto inválido")
print(f" Custo total: {total:21.2f}\n")
print("Estoque:\n")
for chave, dados in estoque.items():
    print("Descrição: ", chave)
    print("Quantidade: ", dados[0])
    print(f"Preço: {dados[1]:6.2f}\n")
```

## EXERCÍCIO 06-18-A

Escreva um programa que gere um dicionário, em que cada chave seja um caractere, e seu valor seja o número desse caractere encontrado em uma frase lida. Exemplo: O rato → { “O”:1, “r”:1, “a”:1, “t”:1, “o”:1}

```
frase = input("Digite uma frase para contar as letras:")
d = {}
for letra in frase:
    if letra in d:
        d[letra] = d[letra] + 1
    else:
        d[letra] = 1
print(d)
```

## EXERCÍCIO 06-18-B

Escreva um programa que gere um dicionário, em que cada chave seja um caractere, e seu valor seja o número desse caractere encontrado em uma frase lida. Exemplo: O rato → { “O”:1, “r”:1, “a”:1, “t”:1, “o”:1}

```
# Solução alternativa, usando o método get do dicionário

frase = input("Digite uma frase para contar as letras:")
d = {}
for letra in frase:
    # Se letra não existir no dicionário, retorna 0
    # se existir, retorna o valor anterior
    d[letra] = d.get(letra, 0) + 1
print(d)
```

## EXERCÍCIO 06-19

Escreva um programa que compare duas listas. Utilizando operações com conjuntos, imprima: • os valores comuns às duas listas • os valores que só existem na primeira • os valores que existem apenas na segunda • uma lista com os elementos não repetidos das duas listas. • a primeira lista sem os elementos repetidos na segunda

```
L1 = [1, 2, 6, 8]
L2 = [3, 6, 8, 9]

print(f"Lista 1: {L1}")
print(f"Lista 2: {L2}")

conjunto_1 = set(L1)
conjunto_2 = set(L2)

# Conjuntos suportam o operador & para realizar a interseção, ou seja,
# A & B resulta no conjunto de elementos presentes em A e B
print("Valores comuns às duas listas:", conjunto_1 & conjunto_2)
print("Valores que só existem na primeira:", conjunto_1 - conjunto_2)
print("Valores que só existem na segunda:", conjunto_2 - conjunto_1)

# Conjuntos suportam o operador ^ que realiza a subtração simétrica.
# A ^ B resulta nos elementos de A não presentes em B unidos
# com os elementos de B não presentes em A
# A ^ B = A - B | B - A
print("Elementos não repetidos nas duas listas:", conjunto_1 ^ conjunto_2)

# Repetido:
print("Primeira lista, sem os elementos repetidos na segunda:", conjunto_1 -
conjunto_2)
```

## EXERCÍCIO 06-20

Escreva um programa que compare duas listas. Considere a primeira lista como a versão inicial e a segunda como a versão após alterações. Utilizando operações com conjuntos, seu programa deverá imprimir a lista de modificações entre essas duas versões, listando: • os elementos que não mudaram • os novos elementos • os elementos que foram removidos

```
ANTES = [1, 2, 5, 6, 9]
DEPOIS = [1, 2, 8, 10]

conjunto_antes = set(ANTES)
conjunto_depois = set(DEPOIS)

# Conjuntos suportam o operador & para realizar a interseção, ou seja,
# A & B resulta no conjunto de elementos presentes em A e B
print("Antes:", ANTES)
print("Depois:", DEPOIS)
print("Elementos que não mudaram: ", conjunto_antes & conjunto_depois)
print("Elementos novos", conjunto_depois - conjunto_antes)
print("Elementos que foram removidos", conjunto_antes - conjunto_depois)
```

## EXERCÍCIO 07-01

Escreva um programa que leia duas strings. Verifique se a segunda ocorre dentro da primeira e imprima a posição de início. 1ª string: AABBEFAATT 2ª string: BE Resultado: BE encontrado na posição 3 de AABBEFAATT

```
primeira = input("Digite a primeira string: ")
segunda = input("Digite a segunda string: ")

posição = primeira.find(segunda)

if posição == -1:
    print(f"'{segunda}' não encontrada em '{primeira}'")
else:
    print(f"{segunda} encontrada na posição {posição} de {primeira}")
```

## EXERCÍCIO 07-02

Escreva um programa que leia duas strings e gere uma terceira com os caracteres comuns às duas strings lidas. 1ª string: AAACTBF 2ª string: CBT Resultado: CBT A ordem dos caracteres da string gerada não é importante, mas deve conter todas as letras comuns a ambas.

```
primeira = input("Digite a primeira string: ")
segunda = input("Digite a segunda string: ")

terceira = ""

# Para cada letra na primeira string
```

```
for letra in primeira:
    # Se a letra está na segunda string (comum a ambas)
    # Para evitar repetidas, não deve estar na terceira.
    if letra in segunda and letra not in terceira:
        terceira += letra

if terceira == "":
    print("Caracteres comuns não encontrados.")
else:
    print(f"Caracteres em comum: {terceira}")
```

## EXERCÍCIO 07-03

Escreva um programa que leia duas strings e gere uma terceira apenas com os caracteres que aparecem em uma delas. 1ª string: CTA 2ª string: ABC 3ª string: BT A ordem dos caracteres da terceira string não é importante.

```
primeira = input("Digite a primeira string: ")
segunda = input("Digite a segunda string: ")

terceira = ""

# Para cada letra na primeira string
for letra in primeira:
    # Verifica se a letra não aparece dentro da segunda string
    # e também se já não está listada na terceira
    if letra not in segunda and letra not in terceira:
        terceira += letra

# Para cada letra na segunda string
for letra in segunda:
    # Além de não estar na primeira string,
    # verifica se já não está na terceira (evitar repetições)
    if letra not in primeira and letra not in terceira:
        terceira += letra

if terceira == "":
    print("Caracteres incomuns não encontrados.")
else:
    print(f"Caracteres incomuns: {terceira}")
```

## EXERCÍCIO 07-04

Escreva um programa que leia uma string e imprima quantas vezes cada caractere aparece nessa string. String: TTAAC Resultado: T: 2x A: 2x C: 1x

```
sequencia = input("Digite a string: ")

contador = {}

for letra in sequencia:
    contador[letra] = contador.get(letra, 0) + 1

for chave, valor in contador.items():
    print(f"{chave}: {valor}x")
```

## EXERCÍCIO 07-05

Escreva um programa que leia duas strings e gere uma terceira, na qual os caracteres da segunda foram retirados da primeira. 1ª string: AATTGGAA 2ª string: TG 3ª string: AAAA

```
primeira = input("Digite a primeira string: ")
segunda = input("Digite a segunda string: ")

terceira = ""

for letra in primeira:
    if letra not in segunda:
        terceira += letra

if terceira == "":
    print("Todos os caracteres foram removidos.")
else:
    print(f"Os caracteres {segunda} foram removidos de {primeira}, gerando:
{terceira}")
```

## EXERCÍCIO 07-06

Escreva um programa que leia três strings. Imprima o resultado da substituição na primeira, dos caracteres da segunda pelos da terceira. 1ª string: AATTCGAA 2ª string: TG 3ª string: AC Resultado: AAAACCAA

```
primeira = input("Digite a primeira string: ")
segunda = input("Digite a segunda string: ")
terceira = input("Digite a terceira string: ")

if len(segunda) == len(terceira):
    resultado = ""
    for letra in primeira:
        posição = segunda.find(letra)
```

```
        if posição != -1:
            resultado += terceira[posição]
        else:
            resultado += letra

    if resultado == "":
        print("Todos os caracteres foram removidos.")
    else:
        print(
            f"Os caracteres {segunda} foram substituidos por "
            f"{terceira} em {primeira}, gerando: {resultado}"
        )
else:
    print("ERRO: A segunda e a terceira string devem ter o mesmo tamanho.")
```

## EXERCÍCIO 07-07

Modifique o o jogo da forca (Programa 7.2) de forma a escrever a palavra secreta caso o jogador perca.

```
palavra = input("Digite a palavra secreta:").lower().strip()
for x in range(100):
    print()
digitadas = []
acertos = []
erros = 0
while True:
    senha = ""
    for letra in palavra:
        senha += letra if letra in acertos else "."
    print(senha)
    if senha == palavra:
        print("Você acertou!")
        break
    tentativa = input("\nDigite uma letra:").lower().strip()
    if tentativa in digitadas:
        print("Você já tentou esta letra!")
        continue
    else:
        digitadas += tentativa
        if tentativa in palavra:
            acertos += tentativa
        else:
            erros += 1
            print("Você errou!")
    print("X==:==\nX  :   ")
    print("X  O   " if erros >= 1 else "X")
    linha2 = ""
    if erros == 2:
        # O r antes da string indica que seu conteúdo não deve ser processado
        # Desta forma, podemos usar os caracteres de \ e / sem confundi-los
        # com máscaras como \n e \t
```

```
        linha2 = r"  |   "
    elif erros == 3:
        linha2 = r" \|   "
    elif erros >= 4:
        linha2 = r" \|/ "
    print(f"X{linha2}")
    linha3 = ""
    if erros == 5:
        linha3 += r" /     "
    elif erros >= 6:
        linha3 += r" / \ "
    print(f"X{linha3}")
    print("X\n===========")
    if erros == 6:
        print("Enforcado!")
        print(f"A palavra secreta era: {palavra}")
        break
```

## EXERCÍCIO 07-08

Modifique o Programa 7.2 de forma a utilizar uma lista de palavras. No início, pergunte um número e calcule o índice da palavra a utilizar pela fórmula: índice = (número * 776) % len(lista_de_palavras).

```
palavras = [
    "casa",
    "bola",
    "mangueira",
    "uva",
    "quiabo",
    "computador",
    "cobra",
    "lentilha",
    "arroz",
]

índice = int(input("Digite um numero:"))
palavra = palavras[(índice * 776) % len(palavras)]
for x in range(100):
    print()
digitadas = []
acertos = []
erros = 0
while True:
    senha = ""
    for letra in palavra:
        senha += letra if letra in acertos else "."
    print(senha)
    if senha == palavra:
        print("Você acertou!")
        break
    tentativa = input("\nDigite uma letra:").lower().strip()
```

```
    if tentativa in digitadas:
        print("Você já tentou esta letra!")
        continue
    else:
        digitadas += tentativa
        if tentativa in palavra:
            acertos += tentativa
        else:
            erros += 1
            print("Você errou!")
    print("X==:==\nX  :   ")
    print("X  O   " if erros >= 1 else "X")
    linha2 = ""
    if erros == 2:
        linha2 = r"  |   "
    elif erros == 3:
        linha2 = r" \|   "
    elif erros >= 4:
        linha2 = r" \|/ "
    print(f"X{linha2}")
    linha3 = ""
    if erros == 5:
        linha3 += r" /     "
    elif erros >= 6:
        linha3 += r" / \ "
    print(f"X{linha3}")
    print("X\n===========")
    if erros == 6:
        print("Enforcado!")
        print(f"A palavra secreta era: {palavra}")
        break
```

## EXERCÍCIO 07-09

Modifique o Programa 7.2 para utilizar listas de strings para desenhar o boneco da forca. Você pode utilizar uma lista para cada linha e organizá-las em uma lista de listas. Em vez de controlar quando imprimir cada parte, desenhe nessas listas, substituindo o elemento a desenhar.

Exemplo:

```
>>> linha = list("X------")
>>> linha
['X', '-', '-', '-', '-', '-', '-']
>>> linha[6] = "|"
>>> linha
['X', '-', '-', '-', '-', '-', '|']
>>> "".join(linha)
'X-----|'
```

```
palavras = [
    "casa",
```

```
    "bola",
    "mangueira",
    "uva",
    "quiabo",
    "computador",
    "cobra",
    "lentilha",
    "arroz",
]

índice = int(input("Digite um numero:"))
palavra = palavras[(índice * 776) % len(palavras)]
for x in range(100):
    print()
digitadas = []
acertos = []
erros = 0

linhas_txt = """
X==:==
X  :
X
X
X
X
=======

"""

linhas = []

for linha in linhas_txt.splitlines():
    linhas.append(list(linha))

while True:
    senha = ""
    for letra in palavra:
        senha += letra if letra in acertos else "."
    print(senha)
    if senha == palavra:
        print("Você acertou!")
        break
    tentativa = input("\nDigite uma letra:").lower().strip()
    if tentativa in digitadas:
        print("Você já tentou esta letra!")
        continue
    else:
        digitadas += tentativa
        if tentativa in palavra:
            acertos += tentativa
        else:
            erros += 1
```

```
            print("Você errou!")
            if erros == 1:
                linhas[3][3] = "O"
            elif erros == 2:
                linhas[4][3] = "|"
            elif erros == 3:
                linhas[4][2] = "\\"
            elif erros == 4:
                linhas[4][4] = "/"
            elif erros == 5:
                linhas[5][2] = "/"
            elif erros == 6:
                linhas[5][4] = "\\"

    for l in linhas:
        print("".join(l))
    if erros == 6:
        print("Enforcado!")
        print(f"A palavra secreta era: {palavra}")
        break
```

## EXERCÍCIO 07-10

Escreva um jogo da velha para dois jogadores. O jogo deve perguntar onde você quer jogar e alternar entre os jogadores. A cada jogada, verifique se a posição está livre. Verifique também quando um jogador venceu a partida. Um jogo da velha pode ser visto como uma lista de 3 elementos, na qual cada elemento é outra lista, também com três elementos.

Exemplo do jogo:

```
X | O |
---+---+---
  | X | X
---+---+---
  |   | O
```

Em que cada posição pode ser vista como um número. Confira a seguir um exemplo das posições mapeadas para a mesma posição de seu teclado numérico.

```
7 | 8 | 9
---+---+---
4 | 5 | 6
---+---+---
1 | 2 | 3
```

```
#
# Jogo da Velha
#

# O tabuleiro
velha = """              Posições
```

```
   |   |      7 | 8 | 9
---+---+---   ---+---+---
   |   |      4 | 5 | 6
---+---+---   ---+---+---
   |   |      1 | 2 | 3
"""
# Uma lista de posições (linha e coluna) para cada posição válida do jogo
# Um elemento extra foi adicionado para facilitar a manipulação
# dos índices e para que estes tenham o mesmo valor da posição
#
#  7 | 8 | 9
# ---+---+---
#  4 | 5 | 6
# ---+---+---
#  1 | 2 | 3

posições = [
    None,  # Elemento adicionado para facilitar índices
    (5, 1),  # 1
    (5, 5),  # 2
    (5, 9),  # 3
    (3, 1),  # 4
    (3, 5),  # 5
    (3, 9),  # 6
    (1, 1),  # 7
    (1, 5),  # 8
    (1, 9),  # 9
]

# Posições que levam ao ganho do jogo
# Jogadas fazendo uma linha, um coluna ou as diagonais ganham
# Os números representam as posições ganhadoras
ganho = [
    [1, 2, 3],  # Linhas
    [4, 5, 6],
    [7, 8, 9],
    [7, 4, 1],  # Colunas
    [8, 5, 2],
    [9, 6, 3],
    [7, 5, 3],  # Diagonais
    [1, 5, 9],
]

# Constroi o tabuleiro a partir das strings
# gerando uma lista de listas que pode ser modificada
tabuleiro = []
for linha in velha.splitlines():
    tabuleiro.append(list(linha))

jogador = "X"  # Começa jogando com X
jogando = True
jogadas = 0  # Contador de jogadas - usado para saber se velhou
```

```
while True:
    for t in tabuleiro:  # Imprime o tabuleiro
        print("".join(t))
    if not jogando:  # Termina após imprimir o último tabuleiro
        break
    if (
        jogadas == 9
    ):  # Se 9 jogadas foram feitas, todas as posições já foram preenchidas
        print("Deu velha! Ninguém ganhou.")
        break
    jogada = int(input(f"Digite a posição a jogar 1-9 (jogador {jogador}):"))
    if jogada < 1 or jogada > 9:
        print("Posição inválida")
        continue
    # Verifica se a posição está livre
    if tabuleiro[posições[jogada][0]][posições[jogada][1]] != " ":
        print("Posição ocupada.")
        continue
    # Marca a jogada para o jogador
    tabuleiro[posições[jogada][0]][posições[jogada][1]] = jogador
    # Verfica se ganhou
    for p in ganho:
        for x in p:
            if tabuleiro[posições[x][0]][posições[x][1]] != jogador:
                break
        else:  # Se o for terminar sem break, todas as posicoes de p pertencem ao
mesmo jogador
            print(f"O jogador {jogador} ganhou ({p}): ")
            jogando = False
            break
    jogador = "X" if jogador == "O" else "O"  # Alterna jogador
    jogadas += 1  # Contador de jogadas

# Sobre a conversão de coordenadas:
# tabuleiro[posições[x][0]][posições[x][1]]
#
# Como tabuleiro é uma lista de listas, podemos acessar cada caracter
# especificando uma linha e uma coluna. Para obter a linha e a coluna, com base
# na posição jogada, usamos a lista de posições que retorna uma tupla com 2
elementos:
# linha e coluna. Sendo linha o elemento [0] e coluna o elemento [1].
# O que estas linhas realizam é a conversão de uma posição de jogo (1-9)
# em linhas e colunas do tabuleiro. Veja que neste exemplo usamos o tabuleiro como
# memória de jogadas, além da exibição do estado atual do jogo.
```

## EXERCÍCIO 08-01

Escreva uma função que retorne o maior de dois números. Valores esperados: máximo(5, 6) == 6 máximo(2, 1) == 2 máximo(7, 7) == 7

```
def máximo(a, b):
    if a > b:
        return a
    else:
        return b


print(f"máximo(5,6) == 6 -> obtido: {máximo(5,6)}")
print(f"máximo(2,1) == 2 -> obtido: {máximo(2,1)}")
print(f"máximo(7,7) == 7 -> obtido: {máximo(7,7)}")
```

## EXERCÍCIO 08-02

Escreva uma função que receba dois números e retorne True se o primeiro número for múltiplo do segundo. Valores esperados: múltiplo(8, 4) == True múltiplo(7, 3) == False múltiplo(5, 5) == True

```
def múltiplo(a, b):
    return a % b == 0


print(f"múltiplo(8,4) == True  -> obtido: {múltiplo(8,4)}")
print(f"múltiplo(7,3) == False -> obtido: {múltiplo(7,3)}")
print(f"múltiplo(5,5) == True  -> obtido: {múltiplo(5,5)}")
```

## EXERCÍCIO 08-03

Escreva uma função que receba o lado de um quadrado e retorne sua área (A = lado²). Valores esperados: área_quadrado(4) == 16 área_quadrado(9) == 81

```
def área_quadrado(l):
    return l**2


print(f"área_quadrado(4) == 16 -> obtido: {área_quadrado(4)}")
print(f"área_quadrado(9) == 81 -> obtido: {área_quadrado(9)}")
```

## EXERCÍCIO 08-04

Escreva uma função que receba a base e a altura de um triângulo e retorne sua área (A = (base x altura) / 2). Valores esperados: área_triângulo(6, 9) == 27 área_triângulo(5, 8) == 20

```
def área_triângulo(b, h):
```

```
    return (b * h) / 2


print(f"área_triângulo(6, 9) == 27 -> obtido: {área_triângulo(6,9)}")
print(f"área_triângulo(5, 8) == 20 -> obtido: {área_triângulo(5,8)}")
```

## EXERCÍCIO 08-05

Reescreva a função do Programa 8.1 de forma a utilizar os métodos de pesquisa em lista, vistos no Capítulo 7.

```
def pesquise(lista, valor):
    if valor in lista:
        return lista.index(valor)
    return None


L = [10, 20, 25, 30]
print(pesquise(L, 25))
print(pesquise(L, 27))
```

## EXERCÍCIO 08-06

Reescreva o Programa 8.2 de forma a utilizar for em vez de while.

```
def soma(L):
    total = 0
    for e in L:
        total += e
    return total


L = [1, 7, 2, 9, 15]
print(soma(L))
print(soma([7, 9, 12, 3, 100, 20, 4]))
```

## EXERCÍCIO 08-07

Defina uma função recursiva que calcule o maior divisor comum (M.D.C.) entre dois números a e b, em que a > b. Ver representação no livro

```
def mdc(a, b):
    if b == 0:
        return a
    return mdc(b, a % b)
```

```
print(f"MDC 10 e 5 -->  {mdc(10,5)}")
print(f"MDC 32 e 24 --> {mdc(32,24)}")
print(f"MDC 5 e 3 -->   {mdc(5,3)}")
```

## EXERCÍCIO 08-08

Usando a função mdc definida no exercício anterior, defina uma função para calcular o menor múltiplo comum (M.M.C.) entre dois números. mmc(a, b) = |a × b| / mdc(a, b) Em que |a × b| pode ser escrito em Python como: abs(a * b).

```
def mdc(a, b):
    if b == 0:
        return a
    return mdc(b, a % b)


def mmc(a, b):
    return abs(a * b) / mdc(a, b)


print(f"MMC 10 e 5 -->  {mmc(10, 5)}")
print(f"MMC 32 e 24 --> {mmc(32, 24)}")
print(f"MMC 5 e 3 -->   {mmc(5, 3)}")
```

## EXERCÍCIO 08-09

Rastreie o Programa 8.6 e compare o seu resultado com o apresentado

```
# Comparando o programa da listagem 8.12 com o resultado
# da listagem 8.13.
#
# O programa calcula o fatorial de 4
# Pelas mensagens impressas na listagem 8.13 e pelo rastreamento do programa,
# podemos concluir que o fatorial de 4 é calculado com chamadas recursivas
# na linha: fat = n * fatorial(n-1)
#
# Como a chamada do fatorial precede a impressão da linha Fatorial de,
# podemos visualizar a sequencia em forma de pilha, onde o cálculo é feito de fora
# para dentro: Calculo do fatorial de 4, 3 , 2 e 1
# para então prosseguir na linha seguinte, que faz a impressão dos resultados:
# fatorial de 1,2,3,4
```

## EXERCÍCIO 08-10

Reescreva a função para cálculo da sequência de Fibonacci, sem utilizar recursão.

```
def fibonacci(n):
    p = 0
    s = 1
    while n > 0:
        p, s = s, s + p
        n -= 1
    return p


for x in range(10):
    print(f"fibonacci({x}) = {fibonacci(x)}")
```

## EXERCÍCIO 08-11

Escreva uma função para validar uma variável string. Essa função recebe como parâmetro a string, o número mínimo e máximo de caracteres. Retorne verdadeiro se o tamanho da string estiver entre os valores de máximo e mínimo, e falso, caso contrário.

```
def valida_string(s, mín, máx):
    tamanho = len(s)
    return mín <= tamanho <= máx


print(valida_string("", 1, 5))
print(valida_string("ABC", 2, 5))
print(valida_string("ABCEFG", 3, 5))
print(valida_string("ABCEFG", 1, 10))
```

## EXERCÍCIO 08-12

Escreva uma função que receba uma string e uma lista. A função deve comparar a string passada com os elementos da lista, também passada como parâmetro. Retorne verdadeiro se a string for encontrada dentro da lista, e falso, caso contrário.

```
def procura_string(s, lista):
    return s in lista


L = ["AB", "CD", "EF", "FG"]

print(procura_string("AB", L))
print(procura_string("CD", L))
print(procura_string("EF", L))
print(procura_string("FG", L))
print(procura_string("XYZ", L))
```

## EXERCÍCIO 08-13-A

Altere o Programa 8.20 de forma que o usuário tenha três chances de acertar o número. O programa termina se o usuário acertar ou errar três vezes.

```
def valida_entrada(mensagem, opções_válidas):
    opções = opções_válidas.lower()
    while True:
        escolha = input(mensagem)
        if escolha.lower() in opções:
            break
        print("Erro: opção inválida. Redigite.\n")
    return escolha


# Exemplo:print(valida_entrada("Escolha uma opção:", "abcde"))
#
# Questão extra: o que acontece se o usuário digitar mais de uma opção?
# Por exemplo, ab.
```

## EXERCÍCIO 08-13-B

Altere o Programa 8.20 de forma que o usuário tenha três chances de acertar o número. O programa termina se o usuário acertar ou errar três vezes.

```
import random

n = random.randint(1, 10)
tentativas = 0
while tentativas < 3:
    x = int(input("Escolha um número entre 1 e 10: "))
    if x == n:
        print("Você acertou!")
        break
    else:
        print("Você errou.")
    tentativas += 1
```

## EXERCÍCIO 08-13-C

Escreva uma função que receba uma string com as opções válidas a aceitar (cada opção é uma letra). Converta as opções válidas para letras minúsculas. Utilize input para ler uma opção, converter o valor para letras minúsculas e verificar se a opção é válida. Em caso de opção inválida, a função deve pedir ao usuário que digite novamente outra opção.

```
def valida_opções(validas):
    validas = validas.lower()
    while True:
        opção = input("Digite uma opção:").lower()
```

```
        if opção in validas:
            return opção
        print("Opção inválida, por favor escolha novamente.")
```

## EXERCÍCIO 08-14

Altere o Programa 7.2, o jogo da forca. Escolha a palavra a adivinhar utilizando números aleatórios.

```
import random

palavras = [
    "casa",
    "bola",
    "mangueira",
    "uva",
    "quiabo",
    "computador",
    "cobra",
    "lentilha",
    "arroz",
]
# Escolhe uma palavra aleatoriamente
palavra = palavras[random.randint(0, len(palavras) - 1)]

digitadas = []
acertos = []
erros = 0

linhas_txt = """
X==:==
X  :
X
X
X
X
=======

"""

linhas = []

for linha in linhas_txt.splitlines():
    linhas.append(list(linha))

while True:
    senha = ""
    for letra in palavra:
        senha += letra if letra in acertos else "."
    print(senha)
    if senha == palavra:
        print("Você acertou!")
```

```
        break
    tentativa = input("\nDigite uma letra:").lower().strip()
    if tentativa in digitadas:
        print("Você já tentou esta letra!")
        continue
    else:
        digitadas += tentativa
        if tentativa in palavra:
            acertos += tentativa
        else:
            erros += 1
            print("Você errou!")
            if erros == 1:
                linhas[3][3] = "O"
            elif erros == 2:
                linhas[4][3] = "|"
            elif erros == 3:
                linhas[4][2] = "\\"
            elif erros == 4:
                linhas[4][4] = "/"
            elif erros == 5:
                linhas[5][2] = "/"
            elif erros == 6:
                linhas[5][4] = "\\"

    for l in linhas:
        print("".join(l))
    if erros == 6:
        print("Enforcado!")
        print(f"A palavra secreta era: {palavra}")
        break
```

## EXERCÍCIO 08-15

Utilizando a função type, escreva uma função recursiva que imprima os elementos de uma lista. Cada elemento deve ser impresso separadamente, um por linha. Considere o caso de listas dentro de listas, como L = [1, [2, 3, 4, [5, 6, 7]]]. A cada nível, imprima a lista mais à direita, como fazemos ao indentar blocos em Python. Dica: envie o nível atual como parâmetro e utilize-o para calcular a quantidade de espaços em branco à esquerda de cada elemento.

```
ESPAÇOS_POR_NÍVEL = 4


def imprime_elementos(l, nivel=0):
    espacos = " " * ESPAÇOS_POR_NÍVEL * nivel
    if type(l) == list:
        print(espacos, "[")
        for e in l:
            imprime_elementos(e, nivel + 1)
        print(espacos, "]")
    else:
```

```
        print(espacos, l)


L = [1, [2, 3, 4, [5, 6, 7]]]

imprime_elementos(L)
```

## EXERCÍCIO 08-16

Escreva um generator capaz de gerar a série dos números primos.

```
def primos(n):
    p = 1  # Posição na sequencia
    yield 2  # 2 é o único primo que é par
    d = 3  # divisor começa com 3
    b = 3  # dividendo começa com 3, é o número que testaremos ser primo ou não
    while p < n:
        # print(d, b, d % b, p, n)
        if b % d == 0:  # Se b é divisível por d, o resto será 0
            if b == d:  # Se b igual a d, todos os valores d já foram testados
                yield b  # b é primo
                p += 1  # incrementa a sequencia
            b += 2  # Passa para o próximo número ímpar
            d = 3  # Recomeça a dividir por 3
        elif d < b:  # Continua tentando?
            d += 2  # Incrementa o divisor para o próximo ímpar
        else:
            b += 2  # Tenta outro número ímpar


for primo in primos(10):
    print(primo)
```

## EXERCÍCIO 08-17

Escreva um generator capaz de gerar a série de Fibonacci.

```
def fibonacci(n):
    p = 0
    s = 1
    while n > 0:
        yield p
        p, s = s, s + p
        n -= 1


for f in fibonacci(10):
    print(f)
```

## EXERCÍCIO 09-01

Escreva um programa que receba o nome de um arquivo pela linha de comando e que imprima todas as linhas desse arquivo.

```
import sys

# Verifica se o parâmetro foi passado
if len(sys.argv) != 2:  # Lembre-se que o nome do programa é o primeiro da lista
    print("\nUso: e09-01.py nome_do_arquivo\n\n")
else:
    nome = sys.argv[1]
    arquivo = open(nome, "r")
    for linha in arquivo.readlines():
        # Como a linha termina com ENTER,
        # retiramos o último caractere antes de imprimir
        print(linha[:-1])
    arquivo.close()

# Não esqueça de ler sobre encodings
# Dependendo do tipo de arquivo e de seu sistema operacional,
# ele pode não imprimir corretamente na tela.
```

## EXERCÍCIO 09-02

Modifique o programa do Exercício 9.1 para que receba mais dois parâmetros: a linha de início e a de fim para impressão. O programa deve imprimir apenas as linhas entre esses dois valores (incluindo as linhas de início e fim).

```
import sys

# Verifica se os parâmetros foram passados
if len(sys.argv) != 4:  # Lembre-se que o nome do programa é o primeiro da lista
    print("\nUso: e09-02.py nome_do_arquivo inicio fim\n\n")
else:
    nome = sys.argv[1]
    inicio = int(sys.argv[2])
    fim = int(sys.argv[3])
    arquivo = open(nome, "r")
    for linha in arquivo.readlines()[inicio - 1 : fim]:
        # Como a linha termina com ENTER,
        # retiramos o último caractere antes de imprimir
        print(linha[:-1])
    arquivo.close()

# Não esqueça de ler sobre encodings
# Dependendo do tipo de arquivo e de seu sistema operacional,
# ele pode não imprimir corretamente na tela.
```

## EXERCÍCIO 09-03

Crie um programa que leia os arquivos pares.txt e ímpares.txt e que crie um só arquivo pareseimpares.txt com todas as linhas dos outros dois arquivos, de forma a preservar a ordem numérica.

```
# Assume que pares e ímpares contém apenas números inteiros
# Assume que os valores em cada arquivo estão ordenados
# Os valores não precisam ser sequenciais
# Tolera linhas em branco
# Pares e ímpares podem ter número de linhas diferentes


def lê_número(arquivo):
    while True:
        número = arquivo.readline()
        # Verifica se conseguiu ler algo
        if número == "":
            return None
        # Ignora linhas em branco
        if número.strip() != "":
            return int(número)


def escreve_número(arquivo, n):
    arquivo.write(f"{n}\n")


pares = open("pares.txt", "r")
ímpares = open("ímpares.txt", "r")
pares_ímpares = open("pareseimpares.txt", "w")
npar = lê_número(pares)
nímpar = lê_número(ímpares)
while True:
    if npar is None and nímpar is None:  # Termina se ambos forem None
        break
    if npar is not None and (nímpar is None or npar <= nímpar):
        escreve_número(pares_ímpares, npar)
        npar = lê_número(pares)
    if nímpar is not None and (npar is None or nímpar <= npar):
        escreve_número(pares_ímpares, nímpar)
        nímpar = lê_número(ímpares)

pares_ímpares.close()
pares.close()
ímpares.close()
```

## EXERCÍCIO 09-04

Crie um programa que receba o nome de dois arquivos como parâmetros da linha de comando e que gere um arquivo de saída com as linhas do primeiro e do segundo arquivo.

```
import sys

# Verifica se os parâmetros foram passados
if len(sys.argv) != 4:  # Lembre-se que o nome do programa é o primeiro da lista
    print("\nUso: e09-04.py primeiro segundo saída\n\n")
else:
    primeiro = open(sys.argv[1], "r")
    segundo = open(sys.argv[2], "r")
    saída = open(sys.argv[3], "w")

    # Funciona de forma similar ao readlines
    for l1 in primeiro:
        saída.write(l1)
    for l2 in segundo:
        saída.write(l2)

    primeiro.close()
    segundo.close()
    saída.close()
```

## EXERCÍCIO 09-05

Crie um programa que inverta a ordem das linhas do arquivo pares.txt. A primeira linha deve conter o maior número; e a última, o menor.

```
pares = open("pares.txt", "r")
saída = open("pares_invertido.txt", "w")

L = pares.readlines()
L.reverse()
for l in L:
    saída.write(l)

pares.close()
saída.close()

# Observe que lemos todas as linhas antes de fazer a inversão
# Esta abordagem não funciona com arquivos grandes
# Alternativa usando with:
#
##with open("pares.txt","r") as pares, open("pares_invertido.txt","w") as saída:
##    L = pares.readlines()
##    L.reverse()
##    for l in L:
##        saída.write(l)
```

## EXERCÍCIO 09-06

Modifique o Programa 9.5 para imprimir 40 vezes o símbolo de = se este for o primeiro caractere da linha. Adicione também a opção para parar de imprimir até que se pressione a tecla Enter cada vez que uma linha iniciar com . (ponto) como primeiro caractere.

```
LARGURA = 79
entrada = open("entrada.txt")
for linha in entrada.readlines():
    if linha[0] == ";":
        continue
    elif linha[0] == ">":
        print(linha[1:].rjust(LARGURA))
    elif linha[0] == "*":
        print(linha[1:].center(LARGURA))
    elif linha[0] == "=":
        print("=" * 40)
    elif linha[0] == ".":
        input("Digite algo para continuar")
        print()
    else:
        print(linha)
entrada.close()
```

## EXERCÍCIO 09-07

Crie um programa que leia um arquivo-texto e gere um arquivo de saída paginado. Cada linha não deve conter mais de 76 caracteres. Cada página terá no máximo 60 linhas. Adicione na última linha de cada página o número da página atual e o nome do arquivo original.

```
# Uma boa fonte de textos para teste é o projeto Gutemberg
# http://www.gutenberg.org/
# Não esqueça de configurar o encoding do arquivo.
#
# Este programa foi testado com Moby Dick
# http://www.gutenberg.org/cache/epub/2701/pg2701.txt
# Gravado com o nome de mobydick.txt
#
LARGURA = 76
LINHAS = 60
NOME_DO_ARQUIVO = "mobydick.txt"


def verifica_pagina(arquivo, linha, pagina):
    if linha == LINHAS:
        rodapé = f"= {NOME_DO_ARQUIVO} - Página: {pagina} ="
        arquivo.write(rodapé.center(LARGURA - 1) + "\n")
        pagina += 1
        linha = 1
    return linha, pagina
```

```
def escreve(arquivo, linha, nlinhas, pagina):
    arquivo.write(linha + "\n")
    return verifica_pagina(arquivo, nlinhas + 1, pagina)


entrada = open(NOME_DO_ARQUIVO, encoding="utf-8")
saída = open("saida_paginada.txt", "w", encoding="utf-8")

pagina = 1
linhas = 1

for linha in entrada.readlines():
    palavras = linha.rstrip().split(" ")
    linha = ""
    for p in palavras:
        p = p.strip()
        if len(linha) + len(p) + 1 > LARGURA:
            linhas, pagina = escreve(saída, linha, linhas, pagina)
            linha = ""
        linha += p + " "
    if linha != "":
        linhas, pagina = escreve(saída, linha, linhas, pagina)

# Para imprimir o número na última página
while linhas != 1:
    linhas, pagina = escreve(saída, "", linhas, pagina)

entrada.close()
saída.close()
```

## EXERCÍCIO 09-08

Modifique o programa do Exercício 9.7 para também receber o número de caracteres por linha e o número de linhas por página pela linha de comando.

```
import sys


def verifica_pagina(arquivo, linha, pagina):
    if linha == LINHAS:
        rodapé = f"= {NOME_DO_ARQUIVO} - Página: {pagina} ="
        arquivo.write(rodapé.center(LARGURA - 1) + "\n")
        pagina += 1
        linha = 1
    return linha, pagina


def escreve(arquivo, linha, nlinhas, pagina):
    arquivo.write(linha + "\n")
```

```
    return verifica_pagina(arquivo, nlinhas + 1, pagina)


if len(sys.argv) != 4:
    print("\nUso: e09-08.py arquivo largura linhas\n\n")
    sys.exit(1)

NOME_DO_ARQUIVO = sys.argv[1]
LARGURA = int(sys.argv[2])
LINHAS = int(sys.argv[3])

entrada = open(NOME_DO_ARQUIVO, encoding="utf-8")
saída = open("saida_paginada.txt", "w", encoding="utf-8")

pagina = 1
linhas = 1

for linha in entrada.readlines():
    palavras = linha.rstrip().split(" ")
    linha = ""
    for p in palavras:
        p = p.strip()
        if len(linha) + len(p) + 1 > LARGURA:
            linhas, pagina = escreve(saída, linha, linhas, pagina)
            linha = ""
        linha += p + " "
    if linha != "":
        linhas, pagina = escreve(saída, linha, linhas, pagina)

# Para imprimir o número na última página
while linhas != 1:
    linhas, pagina = escreve(saída, "", linhas, pagina)

entrada.close()
saída.close()
```

## EXERCÍCIO 09-09

Crie um programa que receba uma lista de nomes de arquivo e os imprima, um por um.

```
import sys

if len(sys.argv) < 2:
    print("\nUso: e09-09.py arquivo1 [arquivo2 arquivo3 arquivoN]\n\n\n")
    sys.exit(1)

for nome in sys.argv[1:]:
    arquivo = open(nome, "r")
    for linha in arquivo:
        print(linha, end="")
    arquivo.close()
```

## EXERCÍCIO 09-10

Crie um programa que receba uma lista de nomes de arquivo e que gere apenas um grande arquivo de saída.

```
import sys

if len(sys.argv) < 2:
    print("\nUso: e09-10.py arquivo1 [arquivo2 arquivo3 arquivoN]\n\n\n")
    sys.exit(1)

saída = open("saida_unica.txt", "w", encoding="utf-8")
for nome in sys.argv[1:]:
    arquivo = open(nome, "r", encoding="utf-8")
    for linha in arquivo:
        saída.write(linha)
    arquivo.close()
saída.close()
```

## EXERCÍCIO 09-11

Crie um programa que leia um arquivo e crie um dicionário em que cada chave é uma palavra e cada valor é o número de ocorrências no arquivo.

```
# Atenção ao encoding no Windows

import sys

if len(sys.argv) != 2:
    print("\nUso: e09-11.py arquivo1\n\n\n")
    sys.exit(1)

nome = sys.argv[1]
contador = {}

arquivo = open(nome, "r", encoding="utf-8")
for linha in arquivo:
    linha = linha.strip().lower()
    palavras = linha.split()
    for p in palavras:
        if p in contador:
            contador[p] += 1
        else:
            contador[p] = 1
arquivo.close()

for chave in contador:
    print(f"{chave} = {contador[chave]}")
```

## EXERCÍCIO 09-12

Modifique o programa do Exercício 9.11 para também registrar a linha e a coluna de cada ocorrência da palavra no arquivo. Para isso, utilize listas nos valores de cada palavra, guardando a linha e a coluna de cada ocorrência.

```
# Atenção ao encoding no Windows
# A contagem de colunas não é tão precisa

import sys

if len(sys.argv) != 2:
    print("\nUso: e09-12.py arquivo1\n\n\n")
    sys.exit(1)

nome = sys.argv[1]
contador = {}
clinha = 1
coluna = 1

arquivo = open(nome, "r", encoding="utf-8")
for linha in arquivo:
    linha = linha.strip().lower()
    palavras = linha.split(" ")  # Com parâmetro considera os espaços repetidos
    for p in palavras:
        if p == "":
            coluna += 1
            continue
        if p in contador:
            contador[p].append((clinha, coluna))
        else:
            contador[p] = [(clinha, coluna)]
        coluna += len(p) + 1
    clinha += 1
    coluna = 1
arquivo.close()

for chave in contador:
    print(f"{chave} = {contador[chave]}")
```

## EXERCÍCIO 09-13

Crie um programa que imprima as linhas de um arquivo. Esse programa deve receber três parâmetros pela linha de comando: o nome do arquivo, a linha inicial e a última linha a imprimir

```
# Idêntico ao exercício 9.02
import sys

# Verifica se os parâmetros foram passados
if len(sys.argv) != 4:  # Lembre-se que o nome do programa é o primeiro da lista
```

```
    print("\nUso: e09-13.py nome_do_arquivo inicio fim\n\n")
else:
    nome = sys.argv[1]
    inicio = int(sys.argv[2])
    fim = int(sys.argv[3])
    arquivo = open(nome, "r")
    for linha in arquivo.readlines()[inicio - 1 : fim]:
        # Como a linha termina com ENTER,
        # retiramos o último caractere antes de imprimir
        print(linha[:-1])
    arquivo.close()

# Não esqueça de ler sobre encodings
# Dependendo do tipo de arquivo e de seu sistema operacional,
# ele pode não imprimir corretamente na tela.
```

## EXERCÍCIO 09-14

Crie um programa que leia um arquivo-texto e elimine os espaços repetidos entre as palavras e no fim das linhas. O arquivo de saída também não deve ter mais de uma linha em branco repetida.

```
# Atenção ao encoding no Windows

import sys

if len(sys.argv) != 3:
    print("\nUso: e09-14.py entrada saida\n\n\n")
    sys.exit(1)

entrada = sys.argv[1]
saida = sys.argv[2]

arquivo = open(entrada, "r", encoding="utf-8")
arq_saida = open(saida, "w", encoding="utf-8")
branco = 0

for linha in arquivo:
    # Elimina espaços a direita
    # Substitua por strip se também
    # quiser eliminar espaços a esquerda
    linha = linha.rstrip()
    linha = linha.replace("  ", "")  # Elimina espaços repetidos
    if linha == "":
        branco += 1  # Conta linhas em branco
    else:
        branco = 0  # Se a linha não está em branco, zera o contador
    if branco < 2:  # Não escreve a partir da segunda linha em branco
        arq_saida.write(linha + "\n")

arquivo.close()
arq_saida.close()
```

# EXERCÍCIO 09-15

Altere o Programa 7.2, o jogo da forca. Utilize um arquivo em que uma palavra seja gravada a cada linha. Use um editor de textos para gerar o arquivo. Ao iniciar o programa, utilize esse arquivo para carregar (ler) a lista de palavras. Experimente também perguntar o nome do jogador e gerar um arquivo com o número de acertos dos cinco melhores.

```
# Modificado para ler a lista de palavras de um arquivo
# Lê um arquivo placar.txt com o número de acertos por jogador
# Lê um arquivo palavras.txt com a lista de palavras
#
# Antes de executar:
#
# Crie um arquivo vazio com o nome placar.txt
# Crie um arquivo de palavras com nome palavras.txt
# contendo uma palavra por linha.
#
# O jogo escolhe aleatoriamente uma palavra deste arquivo
import sys
import random

palavras = []
placar = {}


def carrega_palavras():
    arquivo = open("palavras.txt", "r", encoding="utf-8")
    for palavra in arquivo.readlines():
        palavra = palavra.strip().lower()
        if palavra != "":
            palavras.append(palavra)
    arquivo.close()


def carrega_placar():
    arquivo = open("placar.txt", "r", encoding="utf-8")
    for linha in arquivo.readlines():
        linha = linha.strip()
        if linha != "":
            usuario, contador = linha.split(";")
            placar[usuario] = int(contador)
    arquivo.close()


def salva_placar():
    arquivo = open("placar.txt", "w", encoding="utf-8")
    for usuario in placar.keys():
        arquivo.write("{usuario};{placar[usuario]}\n")
    arquivo.close()


def atualize_placar(nome):
```

```
    if nome in placar:
        placar[nome] += 1
    else:
        placar[nome] = 1
    salva_placar()


def exibe_placar():
    placar_ordenado = []
    for usuario, score in placar.items():
        placar_ordenado.append([usuario, score])
    placar_ordenado.sort(key=lambda score: score[1])
    print("\n\nMelhores jogadores por número de acertos:")
    placar_ordenado.reverse()
    for up in placar_ordenado:
        print(f"{up[0]:30s} {up[1]:10d}")


carrega_palavras()
carrega_placar()

palavra = palavras[random.randint(0, len(palavras) - 1)]

digitadas = []
acertos = []
erros = 0
while True:
    senha = ""
    for letra in palavra:
        senha += letra if letra in acertos else "."
    print(senha)
    if senha == palavra:
        print("Você acertou!")
        nome = input("Digite seu nome: ")
        atualize_placar(nome)
        break
    tentativa = input("\nDigite uma letra:").lower().strip()
    if tentativa in digitadas:
        print("Você já tentou esta letra!")
        continue
    else:
        digitadas += tentativa
        if tentativa in palavra:
            acertos += tentativa
        else:
            erros += 1
            print("Você errou!")
    print("X==:==\nX  :   ")
    print("X  O   " if erros >= 1 else "X")
    linha2 = ""
    if erros == 2:
        linha2 = "  |   "
```

```
    elif erros == 3:
        linha2 = r" \|   "
    elif erros >= 4:
        linha2 = r" \|/ "
    print(f"X{linha2}")
    linha3 = ""
    if erros == 5:
        linha3 += r" /     "
    elif erros >= 6:
        linha3 += r" / \ "
    print(f"X{linha3}")
    print("X\n===========")
    if erros == 6:
        print("Enforcado!")
        break

exibe_placar()
```

## EXERCÍCIO 09-16

Explique como os campos nome e telefone são armazenados no arquivo de saída.

```
# Cada registro da agenda é gravado em uma linha do arquivo.
# Os campos são separados pelo símbolo # (Cerquilha)
# por exemplo:
# Duas entradas, Nilo e João são gravadas em 2 linhas de texto.
# O nome da entrada fica a esquerda do # e o número de telefone a direita
#
# Nilo#1234
# João#5678
```

## EXERCÍCIO 09-17

Altere o Programa 9.6 para exibir o tamanho da agenda no menu principal.

```
agenda = []


def pede_nome():
    return input("Nome: ")


def pede_telefone():
    return input("Telefone: ")


def mostra_dados(nome, telefone):
    print(f"Nome: {nome} Telefone: {telefone}")
```

```
def pede_nome_arquivo():
    return input("Nome do arquivo: ")


def pesquisa(nome):
    mnome = nome.lower()
    for p, e in enumerate(agenda):
        if e[0].lower() == mnome:
            return p
    return None


def novo():
    global agenda
    nome = pede_nome()
    telefone = pede_telefone()
    agenda.append([nome, telefone])


def apaga():
    global agenda
    nome = pede_nome()
    p = pesquisa(nome)
    if p is not None:
        del agenda[p]
    else:
        print("Nome não encontrado.")


def altera():
    p = pesquisa(pede_nome())
    if p is not None:
        nome = agenda[p][0]
        telefone = agenda[p][1]
        print("Encontrado:")
        mostra_dados(nome, telefone)
        nome = pede_nome()
        telefone = pede_telefone()
        agenda[p] = [nome, telefone]
    else:
        print("Nome não encontrado.")


def lista():
    print("\nAgenda\n\n\------")
    for e in agenda:
        mostra_dados(e[0], e[1])
    print("\------\n")


def lê():
```

```
    global agenda
    nome_arquivo = pede_nome_arquivo()
    arquivo = open(nome_arquivo, "r", encoding="utf-8")
    agenda = []
    for l in arquivo.readlines():
        nome, telefone = l.strip().split("#")
        agenda.append([nome, telefone])
    arquivo.close()


def grava():
    nome_arquivo = pede_nome_arquivo()
    arquivo = open(nome_arquivo, "w", encoding="utf-8")
    for e in agenda:
        arquivo.write(f"{e[0]}#{e[1]}\n")
    arquivo.close()


def valida_faixa_inteiro(pergunta, inicio, fim):
    while True:
        try:
            valor = int(input(pergunta))
            if inicio <= valor <= fim:
                return valor
        except ValueError:
            print(f"Valor inválido, favor digitar entre {inicio} e {fim}")


def menu():
    print("""
   1 - Novo
   2 - Altera
   3 - Apaga
   4 - Lista
   5 - Grava
   6 - Lê

   0 - Sai
""")
    print(f"\nNomes na agenda: {len(agenda)}\n")
    return valida_faixa_inteiro("Escolha uma opção: ", 0, 6)


while True:
    opção = menu()
    if opção == 0:
        break
    elif opção == 1:
        novo()
    elif opção == 2:
        altera()
    elif opção == 3:
```

```
        apaga()
    elif opção == 4:
        lista()
    elif opção == 5:
        grava()
    elif opção == 6:
        lê()
```

## EXERCÍCIO 09-18

O que acontece se nome ou telefone contiverem o caractere usado como separador em seus conteúdos? Explique o problema e proponha uma solução.

```
# Se o # aparecer no nome ou telefone de uma entrada na agenda,
# ocorrerá um erro ao ler o arquivo.
# Este erro ocorre pois o número de campos esperados na linha será diferente
# de 2 (nome e telefone).
# O programa não tem como saber que o caracter faz parte de um campo ou de outro.
# Uma solução para este problema é substituir o # dentro de um campo antes de
salvá-lo.
# Desta forma, o separador de campos no arquivo não seria confundido com o
conteúdo.
# Durante a leitura a substituição tem que ser revertida, de forma a obter o mesmo
conteúdo.
```

## EXERCÍCIO 09-19

Altere a função lista para que exiba também a posição de cada elemento.

```
agenda = []


def pede_nome():
    return input("Nome: ")


def pede_telefone():
    return input("Telefone: ")


def mostra_dados(nome, telefone):
    print(f"Nome: {nome} Telefone: {telefone}")


def pede_nome_arquivo():
    return input("Nome do arquivo: ")


def pesquisa(nome):
```

```
    mnome = nome.lower()
    for p, e in enumerate(agenda):
        if e[0].lower() == mnome:
            return p
    return None


def novo():
    global agenda
    nome = pede_nome()
    telefone = pede_telefone()
    agenda.append([nome, telefone])


def apaga():
    global agenda
    nome = pede_nome()
    p = pesquisa(nome)
    if p is not None:
        del agenda[p]
    else:
        print("Nome não encontrado.")


def altera():
    p = pesquisa(pede_nome())
    if p is not None:
        nome = agenda[p][0]
        telefone = agenda[p][1]
        print("Encontrado:")
        mostra_dados(nome, telefone)
        nome = pede_nome()
        telefone = pede_telefone()
        agenda[p] = [nome, telefone]
    else:
        print("Nome não encontrado.")


def lista():
    print("\nAgenda\n\n\------")
    # Usamos a função enumerate para obter a posição na agenda
    for posição, e in enumerate(agenda):
        # Imprimimos a posição, sem saltar linha
        print(f"Posição: {posição}", end="")
        mostra_dados(e[0], e[1])
    print("\------\n")


def lê():
    global agenda
    nome_arquivo = pede_nome_arquivo()
    arquivo = open(nome_arquivo, "r", encoding="utf-8")
```

```
    agenda = []
    for l in arquivo.readlines():
        nome, telefone = l.strip().split("#")
        agenda.append([nome, telefone])
    arquivo.close()


def grava():
    nome_arquivo = pede_nome_arquivo()
    arquivo = open(nome_arquivo, "w", encoding="utf-8")
    for e in agenda:
        arquivo.write(f"{e[0]}#{e[1]}\n")
    arquivo.close()


def valida_faixa_inteiro(pergunta, inicio, fim):
    while True:
        try:
            valor = int(input(pergunta))
            if inicio <= valor <= fim:
                return valor
        except ValueError:
            print(f"Valor inválido, favor digitar entre {inicio} e {fim}")


def menu():
    print("""
   1 - Novo
   2 - Altera
   3 - Apaga
   4 - Lista
   5 - Grava
   6 - Lê

   0 - Sai
""")
    print(f"\nNomes na agenda: {len(agenda)}\n")
    return valida_faixa_inteiro("Escolha uma opção: ", 0, 6)


while True:
    opção = menu()
    if opção == 0:
        break
    elif opção == 1:
        novo()
    elif opção == 2:
        altera()
    elif opção == 3:
        apaga()
    elif opção == 4:
        lista()
```

```
    elif opção == 5:
        grava()
    elif opção == 6:
        lê()
```

## EXERCÍCIO 09-20

Adicione a opção de ordenar a lista por nome no menu principal.

```
agenda = []


def pede_nome():
    return input("Nome: ")


def pede_telefone():
    return input("Telefone: ")


def mostra_dados(nome, telefone):
    print(f"Nome: {nome} Telefone: {telefone}")


def pede_nome_arquivo():
    return input("Nome do arquivo: ")


def pesquisa(nome):
    mnome = nome.lower()
    for p, e in enumerate(agenda):
        if e[0].lower() == mnome:
            return p
    return None


def novo():
    global agenda
    nome = pede_nome()
    telefone = pede_telefone()
    agenda.append([nome, telefone])


def apaga():
    global agenda
    nome = pede_nome()
    p = pesquisa(nome)
    if p is not None:
        del agenda[p]
    else:
        print("Nome não encontrado.")
```

```
def altera():
    p = pesquisa(pede_nome())
    if p is not None:
        nome = agenda[p][0]
        telefone = agenda[p][1]
        print("Encontrado:")
        mostra_dados(nome, telefone)
        nome = pede_nome()
        telefone = pede_telefone()
        agenda[p] = [nome, telefone]
    else:
        print("Nome não encontrado.")


def lista():
    print("\nAgenda\n\n\------")
    # Usamos a função enumerate para obter a posição na agenda
    for posição, e in enumerate(agenda):
        # Imprimimos a posição, sem saltar linha
        print(f"Posição: {posição} ", end="")
        mostra_dados(e[0], e[1])
    print("\------\n")


def lê():
    global agenda
    nome_arquivo = pede_nome_arquivo()
    arquivo = open(nome_arquivo, "r", encoding="utf-8")
    agenda = []
    for l in arquivo.readlines():
        nome, telefone = l.strip().split("#")
        agenda.append([nome, telefone])
    arquivo.close()


def ordena():
    # Você pode ordenar a lista como mostrado no livro
    # com o método de bolhas (bubble sort)
    # Ou combinar o método sort do Python com lambdas para
    # definir a chave da lista
    # agenda.sort(key=lambda e: return e[0])
    fim = len(agenda)
    while fim > 1:
        i = 0
        trocou = False
        while i < (fim - 1):
            if agenda[i] > agenda[i + 1]:
                # Opção: agenda[i], agenda[i+1] = agenda[i+1], agenda[i]
                temp = agenda[i + 1]
                agenda[i + 1] = agenda[i]
```

```
                agenda[i] = temp
                trocou = True
            i += 1
        if not trocou:
            break


def grava():
    nome_arquivo = pede_nome_arquivo()
    arquivo = open(nome_arquivo, "w", encoding="utf-8")
    for e in agenda:
        arquivo.write(f"{e[0]}#{e[1]}\n")
    arquivo.close()


def valida_faixa_inteiro(pergunta, inicio, fim):
    while True:
        try:
            valor = int(input(pergunta))
            if inicio <= valor <= fim:
                return valor
        except ValueError:
            print(f"Valor inválido, favor digitar entre {inicio} e {fim}")


def menu():
    print("""
   1 - Novo
   2 - Altera
   3 - Apaga
   4 - Lista
   5 - Grava
   6 - Lê
   7 - Ordena por nome

   0 - Sai
""")
    print(f"\nNomes na agenda: {len(agenda)}\n")
    return valida_faixa_inteiro("Escolha uma opção: ", 0, 7)


while True:
    opção = menu()
    if opção == 0:
        break
    elif opção == 1:
        novo()
    elif opção == 2:
        altera()
    elif opção == 3:
        apaga()
    elif opção == 4:
```

```
        lista()
    elif opção == 5:
        grava()
    elif opção == 6:
        lê()
    elif opção == 7:
        ordena()
```

## EXERCÍCIO 09-21

Nas funções de altera e apaga, peça que o usuário confirme a alteração e exclusão do nome antes de realizar a operação em si.

```
agenda = []


def pede_nome():
    return input("Nome: ")


def pede_telefone():
    return input("Telefone: ")


def mostra_dados(nome, telefone):
    print(f"Nome: {nome} Telefone: {telefone}")


def pede_nome_arquivo():
    return input("Nome do arquivo: ")


def pesquisa(nome):
    mnome = nome.lower()
    for p, e in enumerate(agenda):
        if e[0].lower() == mnome:
            return p
    return None


def novo():
    global agenda
    nome = pede_nome()
    telefone = pede_telefone()
    agenda.append([nome, telefone])


def confirma(operação):
    while True:
        opção = input(f"Confirma {operação} (S/N)? ").upper()
        if opção in "SN":
```

```
            return opção
        else:
            print("Resposta inválida. Escolha S ou N.")


def apaga():
    global agenda
    nome = pede_nome()
    p = pesquisa(nome)
    if p is not None:
        if confirma("apagamento") == "S":
            del agenda[p]
    else:
        print("Nome não encontrado.")


def altera():
    p = pesquisa(pede_nome())
    if p is not None:
        nome = agenda[p][0]
        telefone = agenda[p][1]
        print("Encontrado:")
        mostra_dados(nome, telefone)
        nome = pede_nome()
        telefone = pede_telefone()
        if confirma("alteração") == "S":
            agenda[p] = [nome, telefone]
    else:
        print("Nome não encontrado.")


def lista():
    print("\nAgenda\n\n\------")
    # Usamos a função enumerate para obter a posição na agenda
    for posição, e in enumerate(agenda):
        # Imprimimos a posição, sem saltar linha
        print(f"Posição: {posição} ", end="")
        mostra_dados(e[0], e[1])
    print("\------\n")


def lê():
    global agenda
    nome_arquivo = pede_nome_arquivo()
    arquivo = open(nome_arquivo, "r", encoding="utf-8")
    agenda = []
    for l in arquivo.readlines():
        nome, telefone = l.strip().split("#")
        agenda.append([nome, telefone])
    arquivo.close()
```

```
def ordena():
    # Você pode ordenar a lista como mostrado no livro
    # com o método de bolhas (bubble sort)
    # Ou combinar o método sort do Python com lambdas para
    # definir a chave da lista
    # agenda.sort(key=lambda e: return e[0])
    fim = len(agenda)
    while fim > 1:
        i = 0
        trocou = False
        while i < (fim - 1):
            if agenda[i] > agenda[i + 1]:
                # Opção: agenda[i], agenda[i+1] = agenda[i+1], agenda[i]
                temp = agenda[i + 1]
                agenda[i + 1] = agenda[i]
                agenda[i] = temp
                trocou = True
            i += 1
        if not trocou:
            break


def grava():
    nome_arquivo = pede_nome_arquivo()
    arquivo = open(nome_arquivo, "w", encoding="utf-8")
    for e in agenda:
        arquivo.write(f"{e[0]}#{e[1]}\n")
    arquivo.close()


def valida_faixa_inteiro(pergunta, inicio, fim):
    while True:
        try:
            valor = int(input(pergunta))
            if inicio <= valor <= fim:
                return valor
        except ValueError:
            print(f"Valor inválido, favor digitar entre {inicio} e {fim}")


def menu():
    print("""
   1 - Novo
   2 - Altera
   3 - Apaga
   4 - Lista
   5 - Grava
   6 - Lê
   7 - Ordena por nome

   0 - Sai
""")
```

```
    print(f"\nNomes na agenda: {len(agenda)}\n")
    return valida_faixa_inteiro("Escolha uma opção: ", 0, 7)


while True:
    opção = menu()
    if opção == 0:
        break
    elif opção == 1:
        novo()
    elif opção == 2:
        altera()
    elif opção == 3:
        apaga()
    elif opção == 4:
        lista()
    elif opção == 5:
        grava()
    elif opção == 6:
        lê()
    elif opção == 7:
        ordena()
```

## EXERCÍCIO 09-22

Ao ler ou gravar uma nova lista, verifique se a agenda atual já foi gravada. Você pode usar uma variável para controlar quando a lista foi alterada (novo, altera, apaga) e reinicializar esse valor quando ela for lida ou gravada.

```
agenda = []

# Variável para marcar uma alteração na agenda
alterada = False


def pede_nome():
    return input("Nome: ")


def pede_telefone():
    return input("Telefone: ")


def mostra_dados(nome, telefone):
    print(f"Nome: {nome} Telefone: {telefone}")


def pede_nome_arquivo():
    return input("Nome do arquivo: ")
```

```
def pesquisa(nome):
    mnome = nome.lower()
    for p, e in enumerate(agenda):
        if e[0].lower() == mnome:
            return p
    return None


def novo():
    global agenda, alterada
    nome = pede_nome()
    telefone = pede_telefone()
    agenda.append([nome, telefone])
    alterada = True


def confirma(operação):
    while True:
        opção = input(f"Confirma {operação} (S/N)? ").upper()
        if opção in "SN":
            return opção
        else:
            print("Resposta inválida. Escolha S ou N.")


def apaga():
    global agenda, alterada
    nome = pede_nome()
    p = pesquisa(nome)
    if p is not None:
        if confirma("apagamento") == "S":
            del agenda[p]
            alterada = True
    else:
        print("Nome não encontrado.")


def altera():
    global alterada
    p = pesquisa(pede_nome())
    if p is not None:
        nome = agenda[p][0]
        telefone = agenda[p][1]
        print("Encontrado:")
        mostra_dados(nome, telefone)
        nome = pede_nome()
        telefone = pede_telefone()
        if confirma("alteração") == "S":
            agenda[p] = [nome, telefone]
            alterada = True
    else:
        print("Nome não encontrado.")
```

```
def lista():
    print("\nAgenda\n\n\------")
    # Usamos a função enumerate para obter a posição na agenda
    for posição, e in enumerate(agenda):
        # Imprimimos a posição, sem saltar linha
        print(f"Posição: {posição} ", end="")
        mostra_dados(e[0], e[1])
    print("\------\n")


def lê():
    global agenda, alterada
    if alterada:
        print(
            "Você não salvou a lista desde a última alteração. Deseja gravá-la
agora?"
        )
        if confirma("gravação") == "S":
            grava()
    print("Ler\n---")
    nome_arquivo = pede_nome_arquivo()
    arquivo = open(nome_arquivo, "r", encoding="utf-8")
    agenda = []
    for l in arquivo.readlines():
        nome, telefone = l.strip().split("#")
        agenda.append([nome, telefone])
    arquivo.close()
    alterada = False


def ordena():
    global alterada
    # Você pode ordenar a lista como mostrado no livro
    # com o método de bolhas (bubble sort)
    # Ou combinar o método sort do Python com lambdas para
    # definir a chave da lista
    # agenda.sort(key=lambda e: return e[0])
    fim = len(agenda)
    while fim > 1:
        i = 0
        trocou = False
        while i < (fim - 1):
            if agenda[i] > agenda[i + 1]:
                # Opção: agenda[i], agenda[i+1] = agenda[i+1], agenda[i]
                temp = agenda[i + 1]
                agenda[i + 1] = agenda[i]
                agenda[i] = temp
                trocou = True
            i += 1
        if not trocou:
```

```
            break
    alterada = True


def grava():
    global alterada
    if not alterada:
        print("Você não alterou a lista. Deseja gravá-la mesmo assim?")
        if confirma("gravação") == "N":
            return
    print("Gravar\n\------")
    nome_arquivo = pede_nome_arquivo()
    arquivo = open(nome_arquivo, "w", encoding="utf-8")
    for e in agenda:
        arquivo.write(f"{e[0]}#{e[1]}\n")
    arquivo.close()
    alterada = False


def valida_faixa_inteiro(pergunta, inicio, fim):
    while True:
        try:
            valor = int(input(pergunta))
            if inicio <= valor <= fim:
                return valor
        except ValueError:
            print(f"Valor inválido, favor digitar entre {inicio} e {fim}")


def menu():
    print("""
   1 - Novo
   2 - Altera
   3 - Apaga
   4 - Lista
   5 - Grava
   6 - Lê
   7 - Ordena por nome

   0 - Sai
""")
    print(f"\nNomes na agenda: {len(agenda)} Alterada: {alterada}\n")
    return valida_faixa_inteiro("Escolha uma opção: ", 0, 7)


while True:
    opção = menu()
    if opção == 0:
        break
    elif opção == 1:
        novo()
    elif opção == 2:
```

```
        altera()
    elif opção == 3:
        apaga()
    elif opção == 4:
        lista()
    elif opção == 5:
        grava()
    elif opção == 6:
        lê()
    elif opção == 7:
        ordena()
```

## EXERCÍCIO 09-23

Altere o programa para ler a última agenda lida ou gravada ao inicializar. Dica: utilize outro arquivo para armazenar o nome.

```
agenda = []

# Variável para marcar uma alteração na agenda
alterada = False


def pede_nome():
    return input("Nome: ")


def pede_telefone():
    return input("Telefone: ")


def mostra_dados(nome, telefone):
    print(f"Nome: {nome} Telefone: {telefone}")


def pede_nome_arquivo():
    return input("Nome do arquivo: ")


def pesquisa(nome):
    mnome = nome.lower()
    for p, e in enumerate(agenda):
        if e[0].lower() == mnome:
            return p
    return None


def novo():
    global agenda, alterada
    nome = pede_nome()
    telefone = pede_telefone()
```

```
    agenda.append([nome, telefone])
    alterada = True


def confirma(operação):
    while True:
        opção = input(f"Confirma {operação} (S/N)? ").upper()
        if opção in "SN":
            return opção
        else:
            print("Resposta inválida. Escolha S ou N.")


def apaga():
    global agenda, alterada
    nome = pede_nome()
    p = pesquisa(nome)
    if p is not None:
        if confirma("apagamento") == "S":
            del agenda[p]
            alterada = True
    else:
        print("Nome não encontrado.")


def altera():
    global alterada
    p = pesquisa(pede_nome())
    if p is not None:
        nome = agenda[p][0]
        telefone = agenda[p][1]
        print("Encontrado:")
        mostra_dados(nome, telefone)
        nome = pede_nome()
        telefone = pede_telefone()
        if confirma("alteração") == "S":
            agenda[p] = [nome, telefone]
            alterada = True
    else:
        print("Nome não encontrado.")


def lista():
    print("\nAgenda\n\n\------")
    # Usamos a função enumerate para obter a posição na agenda
    for posição, e in enumerate(agenda):
        # Imprimimos a posição, sem saltar linha
        print(f"Posição: {posição} ", end="")
        mostra_dados(e[0], e[1])
    print("\------\n")
```

```
def lê_última_agenda_gravada():
    última = última_agenda()
    if última is not None:
        leia_arquivo(última)


def última_agenda():
    try:
        arquivo = open("ultima agenda.dat", "r", encoding="utf-8")
        última = arquivo.readline()[:-1]
        arquivo.close()
    except FileNotFoundError:
        return None
    return última


def atualiza_última(nome):
    arquivo = open("ultima agenda.dat", "w", encoding="utf-8")
    arquivo.write(f"{nome}\n")
    arquivo.close()


def leia_arquivo(nome_arquivo):
    global agenda, alterada
    arquivo = open(nome_arquivo, "r", encoding="utf-8")
    agenda = []
    for l in arquivo.readlines():
        nome, telefone = l.strip().split("#")
        agenda.append([nome, telefone])
    arquivo.close()
    alterada = False


def lê():
    global alterada
    if alterada:
        print(
            "Você não salvou a lista desde a última alteração. Deseja gravá-la
agora?"
        )
        if confirma("gravação") == "S":
            grava()
    print("Ler\n---")
    nome_arquivo = pede_nome_arquivo()
    leia_arquivo(nome_arquivo)
    atualiza_última(nome_arquivo)


def ordena():
    global alterada
    # Você pode ordenar a lista como mostrado no livro
    # com o método de bolhas (bubble sort)
```

```
    # Ou combinar o método sort do Python com lambdas para
    # definir a chave da lista
    # agenda.sort(key=lambda e: return e[0])
    fim = len(agenda)
    while fim > 1:
        i = 0
        trocou = False
        while i < (fim - 1):
            if agenda[i] > agenda[i + 1]:
                # Opção: agenda[i], agenda[i+1] = agenda[i+1], agenda[i]
                temp = agenda[i + 1]
                agenda[i + 1] = agenda[i]
                agenda[i] = temp
                trocou = True
            i += 1
        if not trocou:
            break
    alterada = True


def grava():
    global alterada
    if not alterada:
        print("Você não alterou a lista. Deseja gravá-la mesmo assim?")
        if confirma("gravação") == "N":
            return
    print("Gravar\n\------")
    nome_arquivo = pede_nome_arquivo()
    arquivo = open(nome_arquivo, "w", encoding="utf-8")
    for e in agenda:
        arquivo.write(f"{e[0]}#{e[1]}\n")
    arquivo.close()
    atualiza_última(nome_arquivo)
    alterada = False


def valida_faixa_inteiro(pergunta, inicio, fim):
    while True:
        try:
            valor = int(input(pergunta))
            if inicio <= valor <= fim:
                return valor
        except ValueError:
            print(f"Valor inválido, favor digitar entre {inicio} e {fim}")


def menu():
    print("""
   1 - Novo
   2 - Altera
   3 - Apaga
   4 - Lista
```

```
    5 - Grava
    6 - Lê
    7 - Ordena por nome

    0 - Sai
""")
    print(f"\nNomes na agenda: {len(agenda)} Alterada: {alterada}\n")
    return valida_faixa_inteiro("Escolha uma opção: ", 0, 7)


lê_última_agenda_gravada()

while True:
    opção = menu()
    if opção == 0:
        break
    elif opção == 1:
        novo()
    elif opção == 2:
        altera()
    elif opção == 3:
        apaga()
    elif opção == 4:
        lista()
    elif opção == 5:
        grava()
    elif opção == 6:
        lê()
    elif opção == 7:
        ordena()
```

## EXERCÍCIO 09-24

O que acontece com a agenda se ocorrer um erro de leitura ou gravação? Explique.

```
# Em caso de erro de leitura, o programa pára de executar.
# Se o erro ocorrer durante a gravação, os dados não gravados
# serão perdidos.
# Estes problemas podem ser resolvidos com a alteração das
# funções de leitura e gravação, adicionando-se blocos try/except
# O ideal é que o programa exiba a mensagem de erro e continue rodando.
# No caso da gravação, os dados não devem ser perdidos e o usuário deve poder
# tentar novamente.
```

## EXERCÍCIO 09-25

Altere as funções pede_nome e pede_telefone de forma a receberem um parâmetro opcional. Caso esse parâmetro seja passado, utilize-o como retorno caso a entrada de dados seja vazia.

```
agenda = []

# Variável para marcar uma alteração na agenda
alterada = False


def pede_nome(padrão=""):
    nome = input("Nome: ")
    if nome == "":
        nome = padrão
    return nome


def pede_telefone(padrão=""):
    telefone = input("Telefone: ")
    if telefone == "":
        telefone = padrão
    return telefone


def mostra_dados(nome, telefone):
    print(f"Nome: {nome} Telefone: {telefone}")


def pede_nome_arquivo():
    return input("Nome do arquivo: ")


def pesquisa(nome):
    mnome = nome.lower()
    for p, e in enumerate(agenda):
        if e[0].lower() == mnome:
            return p
    return None


def novo():
    global agenda, alterada
    nome = pede_nome()
    telefone = pede_telefone()
    agenda.append([nome, telefone])
    alterada = True


def confirma(operação):
    while True:
        opção = input(f"Confirma {operação} (S/N)? ").upper()
        if opção in "SN":
            return opção
        else:
            print("Resposta inválida. Escolha S ou N.")
```

```
def apaga():
    global agenda, alterada
    nome = pede_nome()
    p = pesquisa(nome)
    if p is not None:
        if confirma("apagamento") == "S":
            del agenda[p]
            alterada = True
    else:
        print("Nome não encontrado.")


def altera():
    global alterada
    p = pesquisa(pede_nome())
    if p is not None:
        nome = agenda[p][0]
        telefone = agenda[p][1]
        print("Encontrado:")
        mostra_dados(nome, telefone)
        nome = pede_nome(nome)  # Se nada for digitado, mantém o valor
        telefone = pede_telefone(telefone)
        if confirma("alteração") == "S":
            agenda[p] = [nome, telefone]
            alterada = True
    else:
        print("Nome não encontrado.")


def lista():
    print("\nAgenda\n\n\------")
    # Usamos a função enumerate para obter a posição na agenda
    for posição, e in enumerate(agenda):
        # Imprimimos a posição, sem saltar linha
        print(f"Posição: {posição} ", end="")
        mostra_dados(e[0], e[1])
    print("\------\n")


def lê_última_agenda_gravada():
    última = última_agenda()
    if última is not None:
        leia_arquivo(última)


def última_agenda():
    try:
        arquivo = open("ultima agenda.dat", "r", encoding="utf-8")
        última = arquivo.readline()[:-1]
        arquivo.close()
    except FileNotFoundError:
```

```
        return None
    return última


def atualiza_última(nome):
    arquivo = open("ultima agenda.dat", "w", encoding="utf-8")
    arquivo.write(f"{nome}\n")
    arquivo.close()


def leia_arquivo(nome_arquivo):
    global agenda, alterada
    arquivo = open(nome_arquivo, "r", encoding="utf-8")
    agenda = []
    for l in arquivo.readlines():
        nome, telefone = l.strip().split("#")
        agenda.append([nome, telefone])
    arquivo.close()
    alterada = False


def lê():
    global alterada
    if alterada:
        print(
            "Você não salvou a lista desde a última alteração. Deseja gravá-la
agora?"
        )
        if confirma("gravação") == "S":
            grava()
    print("Ler\n---")
    nome_arquivo = pede_nome_arquivo()
    leia_arquivo(nome_arquivo)
    atualiza_última(nome_arquivo)


def ordena():
    global alterada
    # Você pode ordenar a lista como mostrado no livro
    # com o método de bolhas (bubble sort)
    # Ou combinar o método sort do Python com lambdas para
    # definir a chave da lista
    # agenda.sort(key=lambda e: return e[0])
    fim = len(agenda)
    while fim > 1:
        i = 0
        trocou = False
        while i < (fim - 1):
            if agenda[i] > agenda[i + 1]:
                # Opção: agenda[i], agenda[i+1] = agenda[i+1], agenda[i]
                temp = agenda[i + 1]
                agenda[i + 1] = agenda[i]
```

```
                agenda[i] = temp
                trocou = True
            i += 1
        if not trocou:
            break
    alterada = True


def grava():
    global alterada
    if not alterada:
        print("Você não alterou a lista. Deseja gravá-la mesmo assim?")
        if confirma("gravação") == "N":
            return
    print("Gravar\n\------")
    nome_arquivo = pede_nome_arquivo()
    arquivo = open(nome_arquivo, "w", encoding="utf-8")
    for e in agenda:
        arquivo.write(f"{e[0]}#{e[1]}\n")
    arquivo.close()
    atualiza_última(nome_arquivo)
    alterada = False


def valida_faixa_inteiro(pergunta, inicio, fim):
    while True:
        try:
            valor = int(input(pergunta))
            if inicio <= valor <= fim:
                return valor
        except ValueError:
            print(f"Valor inválido, favor digitar entre {inicio} e {fim}")


def menu():
    print("""
   1 - Novo
   2 - Altera
   3 - Apaga
   4 - Lista
   5 - Grava
   6 - Lê
   7 - Ordena por nome

   0 - Sai
""")
    print(f"\nNomes na agenda: {len(agenda)} Alterada: {alterada}\n")
    return valida_faixa_inteiro("Escolha uma opção: ", 0, 7)


lê_última_agenda_gravada()
```

```
while True:
    opção = menu()
    if opção == 0:
        break
    elif opção == 1:
        novo()
    elif opção == 2:
        altera()
    elif opção == 3:
        apaga()
    elif opção == 4:
        lista()
    elif opção == 5:
        grava()
    elif opção == 6:
        lê()
    elif opção == 7:
        ordena()
```

## EXERCÍCIO 09-26

Altere o programa de forma a verificar a repetição de nomes. Gere uma mensagem de erro caso duas entradas na agenda tenham o mesmo nome

```
agenda = []

# Variável para marcar uma alteração na agenda
alterada = False


def pede_nome(padrão=""):
    nome = input("Nome: ")
    if nome == "":
        nome = padrão
    return nome


def pede_telefone(padrão=""):
    telefone = input("Telefone: ")
    if telefone == "":
        telefone = padrão
    return telefone


def mostra_dados(nome, telefone):
    print(f"Nome: {nome} Telefone: {telefone}")


def pede_nome_arquivo():
    return input("Nome do arquivo: ")
```

```
def pesquisa(nome):
    mnome = nome.lower()
    for p, e in enumerate(agenda):
        if e[0].lower() == mnome:
            return p
    return None


def novo():
    global agenda, alterada
    nome = pede_nome()
    if pesquisa(nome) is not None:
        print("Nome já existe!")
        return
    telefone = pede_telefone()
    agenda.append([nome, telefone])
    alterada = True


def confirma(operação):
    while True:
        opção = input(f"Confirma {operação} (S/N)? ").upper()
        if opção in "SN":
            return opção
        else:
            print("Resposta inválida. Escolha S ou N.")


def apaga():
    global agenda, alterada
    nome = pede_nome()
    p = pesquisa(nome)
    if p is not None:
        if confirma("apagamento") == "S":
            del agenda[p]
            alterada = True
    else:
        print("Nome não encontrado.")


def altera():
    global alterada
    p = pesquisa(pede_nome())
    if p is not None:
        nome = agenda[p][0]
        telefone = agenda[p][1]
        print("Encontrado:")
        mostra_dados(nome, telefone)
        nome = pede_nome(nome)  # Se nada for digitado, mantém o valor
        telefone = pede_telefone(telefone)
        if confirma("alteração") == "S":
```

```
            agenda[p] = [nome, telefone]
            alterada = True
    else:
        print("Nome não encontrado.")


def lista():
    print("\nAgenda\n\n\------")
    # Usamos a função enumerate para obter a posição na agenda
    for posição, e in enumerate(agenda):
        # Imprimimos a posição, sem saltar linha
        print(f"Posição: {posição} ", end="")
        mostra_dados(e[0], e[1])
    print("\------\n")


def lê_última_agenda_gravada():
    última = última_agenda()
    if última is not None:
        leia_arquivo(última)


def última_agenda():
    try:
        arquivo = open("ultima agenda.dat", "r", encoding="utf-8")
        última = arquivo.readline()[:-1]
        arquivo.close()
    except FileNotFoundError:
        return None
    return última


def atualiza_última(nome):
    arquivo = open("ultima agenda.dat", "w", encoding="utf-8")
    arquivo.write(f"{nome}\n")
    arquivo.close()


def leia_arquivo(nome_arquivo):
    global agenda, alterada
    arquivo = open(nome_arquivo, "r", encoding="utf-8")
    agenda = []
    for l in arquivo.readlines():
        nome, telefone = l.strip().split("#")
        agenda.append([nome, telefone])
    arquivo.close()
    alterada = False


def lê():
    global alterada
    if alterada:
```

```
        print(
            "Você não salvou a lista desde a última alteração. Deseja gravá-la
agora?"
        )
        if confirma("gravação") == "S":
            grava()
    print("Ler\n---")
    nome_arquivo = pede_nome_arquivo()
    leia_arquivo(nome_arquivo)
    atualiza_última(nome_arquivo)


def ordena():
    global alterada
    # Você pode ordenar a lista como mostrado no livro
    # com o método de bolhas (bubble sort)
    # Ou combinar o método sort do Python com lambdas para
    # definir a chave da lista
    # agenda.sort(key=lambda e: return e[0])
    fim = len(agenda)
    while fim > 1:
        i = 0
        trocou = False
        while i < (fim - 1):
            if agenda[i] > agenda[i + 1]:
                # Opção: agenda[i], agenda[i+1] = agenda[i+1], agenda[i]
                temp = agenda[i + 1]
                agenda[i + 1] = agenda[i]
                agenda[i] = temp
                trocou = True
            i += 1
        if not trocou:
            break
    alterada = True


def grava():
    global alterada
    if not alterada:
        print("Você não alterou a lista. Deseja gravá-la mesmo assim?")
        if confirma("gravação") == "N":
            return
    print("Gravar\n\------")
    nome_arquivo = pede_nome_arquivo()
    arquivo = open(nome_arquivo, "w", encoding="utf-8")
    for e in agenda:
        arquivo.write(f"{e[0]}#{e[1]}\n")
    arquivo.close()
    atualiza_última(nome_arquivo)
    alterada = False
```

```
def valida_faixa_inteiro(pergunta, inicio, fim):
    while True:
        try:
            valor = int(input(pergunta))
            if inicio <= valor <= fim:
                return valor
        except ValueError:
            print(f"Valor inválido, favor digitar entre {inicio} e {fim}")


def menu():
    print("""
   1 - Novo
   2 - Altera
   3 - Apaga
   4 - Lista
   5 - Grava
   6 - Lê
   7 - Ordena por nome

   0 - Sai
""")
    print(f"\nNomes na agenda: {len(agenda)} Alterada: {alterada}\n")
    return valida_faixa_inteiro("Escolha uma opção: ", 0, 7)


lê_última_agenda_gravada()


while True:
    opção = menu()
    if opção == 0:
        break
    elif opção == 1:
        novo()
    elif opção == 2:
        altera()
    elif opção == 3:
        apaga()
    elif opção == 4:
        lista()
    elif opção == 5:
        grava()
    elif opção == 6:
        lê()
    elif opção == 7:
        ordena()
```

## EXERCÍCIO 09-27

Modifique o programa para também controlar a data de aniversário e o email de cada pessoa.

```
agenda = []

# Variável para marcar uma alteração na agenda
alterada = False


def pede_nome(padrão=""):
    nome = input("Nome: ")
    if nome == "":
        nome = padrão
    return nome


def pede_telefone(padrão=""):
    telefone = input("Telefone: ")
    if telefone == "":
        telefone = padrão
    return telefone


def pede_email(padrão=""):
    email = input("Email: ")
    if email == "":
        email = padrão
    return email


def pede_aniversário(padrão=""):
    aniversário = input("Data de aniversário: ")
    if aniversário == "":
        aniversário = padrão
    return aniversário


def mostra_dados(nome, telefone, email, aniversário):
    print(
        f"Nome: {nome}\nTelefone: {telefone}\n"
        f"Email: {email}\nAniversário: {aniversário}\n"
    )


def pede_nome_arquivo():
    return input("Nome do arquivo: ")


def pesquisa(nome):
    mnome = nome.lower()
    for p, e in enumerate(agenda):
```

```
        if e[0].lower() == mnome:
            return p
    return None


def novo():
    global agenda, alterada
    nome = pede_nome()
    if pesquisa(nome) is not None:
        print("Nome já existe!")
        return
    telefone = pede_telefone()
    email = pede_email()
    aniversário = pede_aniversário()
    agenda.append([nome, telefone, email, aniversário])
    alterada = True


def confirma(operação):
    while True:
        opção = input(f"Confirma {operação} (S/N)? ").upper()
        if opção in "SN":
            return opção
        else:
            print("Resposta inválida. Escolha S ou N.")


def apaga():
    global agenda, alterada
    nome = pede_nome()
    p = pesquisa(nome)
    if p is not None:
        if confirma("apagamento") == "S":
            del agenda[p]
            alterada = True
    else:
        print("Nome não encontrado.")


def altera():
    global alterada
    p = pesquisa(pede_nome())
    if p is not None:
        nome = agenda[p][0]
        telefone = agenda[p][1]
        email = agenda[p][2]
        aniversário = agenda[p][3]
        print("Encontrado:")
        mostra_dados(nome, telefone, email, aniversário)
        nome = pede_nome(nome)  # Se nada for digitado, mantém o valor
        telefone = pede_telefone(telefone)
        email = pede_email(email)
```

```
        aniversário = pede_aniversário(aniversário)
        if confirma("alteração") == "S":
            agenda[p] = [nome, telefone, email, aniversário]
            alterada = True
    else:
        print("Nome não encontrado.")


def lista():
    print("\nAgenda\n\n\------")
    # Usamos a função enumerate para obter a posição na agenda
    for posição, e in enumerate(agenda):
        # Imprimimos a posição
        print(f"\nPosição: {posição}")
        mostra_dados(e[0], e[1], e[2], e[3])
    print("\------\n")


def lê_última_agenda_gravada():
    última = última_agenda()
    if última is not None:
        leia_arquivo(última)


def última_agenda():
    try:
        arquivo = open("ultima agenda.dat", "r", encoding="utf-8")
        última = arquivo.readline()[:-1]
        arquivo.close()
    except FileNotFoundError:
        return None
    return última


def atualiza_última(nome):
    arquivo = open("ultima agenda.dat", "w", encoding="utf-8")
    arquivo.write(f"{nome}\n")
    arquivo.close()


def leia_arquivo(nome_arquivo):
    global agenda, alterada
    arquivo = open(nome_arquivo, "r", encoding="utf-8")
    agenda = []
    for l in arquivo.readlines():
        nome, telefone, email, aniversário = l.strip().split("#")
        agenda.append([nome, telefone, email, aniversário])
    arquivo.close()
    alterada = False


def lê():
```

```
    global alterada
    if alterada:
        print(
            "Você não salvou a lista desde a última alteração. Deseja gravá-la
agora?"
        )
        if confirma("gravação") == "S":
            grava()
    print("Ler\n---")
    nome_arquivo = pede_nome_arquivo()
    leia_arquivo(nome_arquivo)
    atualiza_última(nome_arquivo)


def ordena():
    global alterada
    # Você pode ordenar a lista como mostrado no livro
    # com o método de bolhas (bubble sort)
    # Ou combinar o método sort do Python com lambdas para
    # definir a chave da lista
    # agenda.sort(key=lambda e: return e[0])
    fim = len(agenda)
    while fim > 1:
        i = 0
        trocou = False
        while i < (fim - 1):
            if agenda[i] > agenda[i + 1]:
                # Opção: agenda[i], agenda[i+1] = agenda[i+1], agenda[i]
                temp = agenda[i + 1]
                agenda[i + 1] = agenda[i]
                agenda[i] = temp
                trocou = True
            i += 1
        if not trocou:
            break
    alterada = True


def grava():
    global alterada
    if not alterada:
        print("Você não alterou a lista. Deseja gravá-la mesmo assim?")
        if confirma("gravação") == "N":
            return
    print("Gravar\n\------")
    nome_arquivo = pede_nome_arquivo()
    arquivo = open(nome_arquivo, "w", encoding="utf-8")
    for e in agenda:
        arquivo.write(f"{e[0]}#{e[1]}#{e[2]}#{e[3]}\n")
    arquivo.close()
    atualiza_última(nome_arquivo)
    alterada = False
```

```
def valida_faixa_inteiro(pergunta, inicio, fim):
    while True:
        try:
            valor = int(input(pergunta))
            if inicio <= valor <= fim:
                return valor
        except ValueError:
            print("Valor inválido, favor digitar entre {inicio} e {fim}")


def menu():
    print("""
   1 - Novo
   2 - Altera
   3 - Apaga
   4 - Lista
   5 - Grava
   6 - Lê
   7 - Ordena por nome

   0 - Sai
""")
    print(f"\nNomes na agenda: {len(agenda)} Alterada: {alterada}\n")
    return valida_faixa_inteiro("Escolha uma opção: ", 0, 7)


lê_última_agenda_gravada()


while True:
    opção = menu()
    if opção == 0:
        break
    elif opção == 1:
        novo()
    elif opção == 2:
        altera()
    elif opção == 3:
        apaga()
    elif opção == 4:
        lista()
    elif opção == 5:
        grava()
    elif opção == 6:
        lê()
    elif opção == 7:
        ordena()
```

## EXERCÍCIO 09-28

Modifique o programa de forma a poder registrar vários telefones para a mesma pessoa. Permita também cadastrar o tipo de telefone: celular, fixo, residência ou trabalho.

```
# Como o formato do arquivo se torna cada vez mais complicado,
# vamos usar o módulo pickle do Python para gravar e ler a agenda.
#
# Desafio extra:
# Modifique o programa para exibir um submenu de gerência de telefones.
# Este sub menu seria exibido na hora de adicionar e alterar telefones.
# Operações: adicionar novo telefone, apagar telefone, alterar telefone
import pickle

agenda = []

# Variável para marcar uma alteração na agenda
alterada = False

tipos_de_telefone = ["celular", "fixo", "residência", "trabalho", "fax"]


def pede_nome(padrão=""):
    nome = input("Nome: ")
    if nome == "":
        nome = padrão
    return nome


def pede_telefone(padrão=""):
    telefone = input("Telefone: ")
    if telefone == "":
        telefone = padrão
    return telefone


def pede_tipo_telefone(padrão=""):
    while True:
        tipo = input("Tipo de telefone [%s]: " % ",".join(tipos_de_telefone)).
lower()
        if tipo == "":
            tipo = padrão
        for t in tipos_de_telefone:
            if t.startswith(tipo):
                return t  # Retorna o nome completo
        else:
            print("Tipo de telefone inválido!")


def pede_email(padrão=""):
    email = input("Email: ")
    if email == "":
```

```
        email = padrão
    return email


def pede_aniversário(padrão=""):
    aniversário = input("Data de aniversário: ")
    if aniversário == "":
        aniversário = padrão
    return aniversário


def mostra_dados(nome, telefones, email, aniversário):
    print(f"Nome: {nome.capitalize()}")
    print("Telefone(s):")
    for telefone in telefones:
        print(f"\tNúmero: {telefone[0]:15s} Tipo: {telefone[1].capitalize()}")
    print(f"Email: {email}\nAniversário: {aniversário}\n")


def pede_nome_arquivo():
    return input("Nome do arquivo: ")


def pesquisa(nome):
    mnome = nome.lower()
    for p, e in enumerate(agenda):
        if e[0].lower() == mnome:
            return p
    return None


def novo():
    global agenda, alterada
    nome = pede_nome()
    if pesquisa(nome) is not None:
        print("Nome já existe!")
        return
    telefones = []
    while True:
        numero = pede_telefone()
        tipo = pede_tipo_telefone()
        telefones.append([numero, tipo])
        if confirma("que deseja cadastrar outro telefone") == "N":
            break
    email = pede_email()
    aniversário = pede_aniversário()
    agenda.append([nome, telefones, email, aniversário])
    alterada = True


def confirma(operação):
    while True:
```

```
        opção = input(f"Confirma {operação} (S/N)? ").upper()
        if opção in "SN":
            return opção
        else:
            print("Resposta inválida. Escolha S ou N.")


def apaga():
    global agenda, alterada
    nome = pede_nome()
    p = pesquisa(nome)
    if p is not None:
        if confirma("apagamento") == "S":
            del agenda[p]
            alterada = True
    else:
        print("Nome não encontrado.")


def altera():
    global alterada
    p = pesquisa(pede_nome())
    if p is not None:
        nome, telefones, email, aniversário = agenda[p]
        print("Encontrado:")
        mostra_dados(nome, telefones, email, aniversário)
        nome = pede_nome(nome)  # Se nada for digitado, mantém o valor
        for telefone in telefones:
            numero, tipo = telefone
            telefone[0] = pede_telefone(numero)
            telefone[1] = pede_tipo_telefone(tipo)
        email = pede_email(email)
        aniversário = pede_aniversário(aniversário)
        if confirma("alteração") == "S":
            agenda[p] = [nome, telefones, email, aniversário]
            alterada = True
    else:
        print("Nome não encontrado.")


def lista():
    print("\nAgenda\n\n\------")
    # Usamos a função enumerate para obter a posição na agenda
    for posição, e in enumerate(agenda):
        # Imprimimos a posição
        print(f"\nPosição: {posição}")
        mostra_dados(e[0], e[1], e[2], e[3])
    print("\------\n")


def lê_última_agenda_gravada():
    última = última_agenda()
```

```
    if última is not None:
        leia_arquivo(última)


def última_agenda():
    try:
        arquivo = open("ultima agenda picke.dat", "r", encoding="utf-8")
        última = arquivo.readline()[:-1]
        arquivo.close()
    except FileNotFoundError:
        return None
    return última


def atualiza_última(nome):
    arquivo = open("ultima agenda picke.dat", "w", encoding="utf-8")
    arquivo.write(f"{nome}\n")
    arquivo.close()


def leia_arquivo(nome_arquivo):
    global agenda, alterada
    arquivo = open(nome_arquivo, "rb")
    agenda = pickle.load(arquivo)
    arquivo.close()
    alterada = False


def lê():
    global alterada
    if alterada:
        print(
            "Você não salvou a lista desde a última alteração. Deseja gravá-la
agora?"
        )
        if confirma("gravação") == "S":
            grava()
    print("Ler\n---")
    nome_arquivo = pede_nome_arquivo()
    leia_arquivo(nome_arquivo)
    atualiza_última(nome_arquivo)


def ordena():
    global alterada
    # Você pode ordenar a lista como mostrado no livro
    # com o método de bolhas (bubble sort)
    # Ou combinar o método sort do Python com lambdas para
    # definir a chave da lista
    # agenda.sort(key=lambda e: return e[0])
    fim = len(agenda)
    while fim > 1:
```

```
        i = 0
        trocou = False
        while i < (fim - 1):
            if agenda[i] > agenda[i + 1]:
                # Opção: agenda[i], agenda[i+1] = agenda[i+1], agenda[i]
                temp = agenda[i + 1]
                agenda[i + 1] = agenda[i]
                agenda[i] = temp
                trocou = True
            i += 1
        if not trocou:
            break
    alterada = True


def grava():
    global alterada
    if not alterada:
        print("Você não alterou a lista. Deseja gravá-la mesmo assim?")
        if confirma("gravação") == "N":
            return
    print("Gravar\n\------")
    nome_arquivo = pede_nome_arquivo()

    arquivo = open(nome_arquivo, "wb")
    pickle.dump(agenda, arquivo)
    arquivo.close()
    atualiza_última(nome_arquivo)
    alterada = False


def valida_faixa_inteiro(pergunta, inicio, fim):
    while True:
        try:
            valor = int(input(pergunta))
            if inicio <= valor <= fim:
                return valor
        except ValueError:
            print(f"Valor inválido, favor digitar entre {inicio} e {fim}")


def menu():
    print("""
   1 - Novo
   2 - Altera
   3 - Apaga
   4 - Lista
   5 - Grava
   6 - Lê
   7 - Ordena por nome

   0 - Sai
```

```
""")
    print(f"\nNomes na agenda: {len(agenda)} Alterada: {alterada}\n")
    return valida_faixa_inteiro("Escolha uma opção: ", 0, 7)


lê_última_agenda_gravada()

while True:
    opção = menu()
    if opção == 0:
        break
    elif opção == 1:
        novo()
    elif opção == 2:
        altera()
    elif opção == 3:
        apaga()
    elif opção == 4:
        lista()
    elif opção == 5:
        grava()
    elif opção == 6:
        lê()
    elif opção == 7:
        ordena()
```

## EXERCÍCIO 09-29

Modifique o Programa 9.8 para utilizar o elemento p em vez de h2 nos filmes.

```
filmes = {
    "drama": ["Cidadão Kane", "O Poderoso Chefão"],
    "comédia": ["Tempos Modernos", "American Pie", "Dr. Dolittle"],
    "policial": ["Chuva Negra", "Desejo de Matar", "Difícil de Matar"],
    "guerra": ["Rambo", "Platoon", "Tora!Tora!Tora!"],
}

pagina = open("filmes.html", "w", encoding="utf-8")
pagina.write("""
<!DOCTYPE html>
<html lang="pt-BR">
<head>
<meta charset="utf-8">
<title>Filmes</title>
</head>
<body>
""")
for c, v in filmes.items():
    pagina.write(f"<h1>{c}</h1>")
    for e in v:
        pagina.write(f"<p>{e}</p>")
```

```
pagina.write("""
</body>
</html>
""")
pagina.close()
```

## EXERCÍCIO 09-30

Modifique o Programa 9.8 para gerar uma lista html, usando os elementos ul e li. Todos os elementos da lista devem estar dentro do elemento ul, e cada item dentro de um elemento li. Exemplo: <ul><li>Item1</li><li>Item2</li><li>Item3</li></ul>V

```
filmes = {
    "drama": ["Cidadão Kane", "O Poderoso Chefão"],
    "comédia": ["Tempos Modernos", "American Pie", "Dr. Dolittle"],
    "policial": ["Chuva Negra", "Desejo de Matar", "Difícil de Matar"],
    "guerra": ["Rambo", "Platoon", "Tora!Tora!Tora!"],
}

pagina = open("filmes.html", "w", encoding="utf-8")
pagina.write("""
<!DOCTYPE html>
<html lang="pt-BR">
<head>
<meta charset="utf-8">
<title>Filmes</title>
</head>
<body>
""")
for c, v in filmes.items():
    pagina.write(f"<h1>{c.capitalize()}</h1>")
    pagina.write("<ul>")
    for e in v:
        pagina.write(f"<li>{e}</li>")
    pagina.write("</ul>")
pagina.write("""
</body>
</html>
""")
pagina.close()
```

## EXERCÍCIO 09-31

Crie um programa que corrija o Programa 9.9 de forma a verificar se z existe e é um diretório.

```
import os.path

if os.path.isdir("z"):
    print("O diretório z existe.")
```

```
elif os.path.isfile("z"):
    print("z existe, mas é um arquivo e não um diretório.")
else:
    print("O diretório z não existe.")
```

## EXERCÍCIO 09-32

Modifique o Programa 9.9 de forma a receber o nome do arquivo ou diretório a verificar pela linha de comando. Imprima se existir e se for um arquivo ou um diretório.

```
import sys
import os.path

if len(sys.argv) < 2:
    print("Digite o nome do arquivo ou diretório a verificar como parâmatro!")
    sys.exit(1)

nome = sys.argv[1]
if os.path.isdir(nome):
    print(f"O diretório {nome} existe.")
elif os.path.isfile(nome):
    print(f"O arquivo {nome} existe.")
else:
    print(f"{nome} não existe.")
```

## EXERCÍCIO 09-33

Crie um programa que gere uma página html com links para todos os arquivos jpg e png encontrados a partir de um diretório informado na linha de comando.

```
# Esta exercício pode ser realizado também com o módulo glob
# Consulte a documentação do Python para mais informações
import sys
import os
import os.path

# este módulo ajuda com a conversão de nomes de arquivos para links
# válidos em HTML
import urllib.request

if len(sys.argv) < 2:
    print("Digite o nome do diretório para coletar os arquivos jpg e png!")
    sys.exit(1)

diretório = sys.argv[1]

pagina = open("imagens.html", "w", encoding="utf-8")
pagina.write("""
<!DOCTYPE html>
```

```
<html lang="pt-BR">
<head>
<meta charset="utf-8">
<title>Imagens PNG e JPG</title>
</head>
<body>
""")
pagina.write(f"Imagens encontradas no diretório: {diretório}")
for entrada in os.listdir(diretório):
    nome, extensão = os.path.splitext(entrada)
    if extensão in [".jpg", ".png"]:
        caminho_completo = os.path.join(diretório, entrada)
        link = urllib.request.pathname2url(caminho_completo)
        pagina.write(f"<p><a href='{link}'>{entrada}</a></p>")

pagina.write("""
</body>
</html>
""")
pagina.close()
```

## EXERCÍCIO 09-34

Altere o Programa 7.2, o jogo da forca. Dessa vez, utilize as funções de tempo para cronometrar a duração das partidas.

```
import time

palavra = input("Digite a palavra secreta:").lower().strip()
for x in range(100):
    print()
digitadas = []
acertos = []
erros = 0

inicio = time.time()  # Registra o início da partida
while True:
    senha = ""
    for letra in palavra:
        senha += letra if letra in acertos else "."
    print(senha)
    if senha == palavra:
        print("Você acertou!")
        break
    tentativa = input("\nDigite uma letra:").lower().strip()
    if tentativa in digitadas:
        print("Você já tentou esta letra!")
        continue
    else:
        digitadas += tentativa
        if tentativa in palavra:
```

```
            acertos += tentativa
        else:
            erros += 1
            print("Você errou!")
    print("X==:==\nX  :   ")
    print("X  O   " if erros >= 1 else "X")
    linha2 = ""
    if erros == 2:
        linha2 = "  |   "
    elif erros == 3:
        linha2 = r" \|   "
    elif erros >= 4:
        linha2 = r" \|/ "
    print(f"X{linha2}")
    linha3 = ""
    if erros == 5:
        linha3 += " /     "
    elif erros >= 6:
        linha3 += r" / \ "
    print(f"X{linha3}")
    print("X\n===========")
    if erros == 6:
        print("Enforcado!")
        break
fim = time.time()  # tempo no fim da partida
print(f"Duração da partida {fim - inicio} segundos")
```

## EXERCÍCIO 09-35

Utilizando a função os.walk, crie uma página HTML com o nome e tamanho de cada arquivo de um diretório passado e de seus subdiretórios.

```
import sys
import os
import os.path

# este módulo ajuda com a conversão de nomes de arquivos para links
# válidos em HTML
import urllib.request

mascara_do_estilo = "'margin: 5px 0px 5px %dpx;'"


def gera_estilo(nível):
    return mascara_do_estilo % (nível * 20)


def gera_listagem(página, diretório):
    nraiz = os.path.abspath(diretório).count(os.sep)
    for raiz, diretórios, arquivos in os.walk(diretório):
        nível = raiz.count(os.sep) - nraiz
```

```
        página.write(f"<p style={gera_estilo(nível)}>{raiz}</p>")
        estilo = gera_estilo(nível + 1)
        for a in arquivos:
            caminho_completo = os.path.join(raiz, a)
            tamanho = os.path.getsize(caminho_completo)
            link = urllib.request.pathname2url(caminho_completo)
            página.write(
                f"<p style={estilo}><a href='{link}'>{a}</a>  ({tamanho} bytes)</
p>"
            )


if len(sys.argv) < 2:
    print("Digite o nome do diretório para coletar os arquivos!")
    sys.exit(1)

diretório = sys.argv[1]

página = open("arquivos.html", "w", encoding="utf-8")
página.write("""
<!DOCTYPE html>
<html lang="pt-BR">
<head>
<meta charset="utf-8">
<title>Arquivos</title>
</head>
<body>
""")
página.write(f"Arquivos encontrados a partir do diretório: {diretório}")
gera_listagem(página, diretório)
página.write("""
</body>
</html>
""")
página.close()
```

## EXERCÍCIO 09-36

Utilizando a função os.walk, crie um programa que calcule o espaço ocupado por cada diretório e subdiretório, gerando uma página html com os resultados.

```
import sys
import os
import os.path
import math


# Esta função converte o tamanho
# em unidades mais legíveis, evitando
# retornar e imprimir valores muito grandes.
def tamanho_para_humanos(tamanho):
```

```
    if tamanho == 0:
        return "0 byte"
    grandeza = math.log(tamanho, 10)
    if grandeza < 3:
        return f"{tamanho} bytes"
    elif grandeza < 6:
        return f"{tamanho / 1024.0:7.3f} KB"
    elif grandeza < 9:
        return f"{tamanho / pow(1024, 2)} MB"
    elif grandeza < 12:
        return f"{tamanho / pow(1024, 3)} GB"


mascara_do_estilo = "'margin: 5px 0px 5px %dpx;'"


def gera_estilo(nível):
    return mascara_do_estilo % (nível * 30)


# Retorna uma função, onde o parâmetro nraiz é utilizado
# para calcular o nível da identação
def gera_nível_e_estilo(raiz):
    def nivel(caminho):
        xnivel = caminho.count(os.sep) - nraiz
        return gera_estilo(xnivel)

    nraiz = raiz.count(os.sep)
    return nivel


# Usa a os.walk para percorrer os diretórios
# E uma pilha para armazenar o tamanho de cada diretório
def gera_listagem(página, diretório):
    diretório = os.path.abspath(diretório)
    # identador é uma função que calcula quantos níveis
    # a partir do nível de diretório um caminho deve possuir.
    identador = gera_nível_e_estilo(diretório)
    pilha = [[diretório, 0]]  # Elemento de guarda, para evitar pilha vazia
    for raiz, diretórios, arquivos in os.walk(diretório):
        # Se o diretório atual: raiz
        # Não for um subdiretório do último percorrido
        # Desempilha até achar um pai comum
        while not raiz.startswith(pilha[-1][0]):
            último = pilha.pop()
            página.write(
                f"<p style={identador(último[0])}>Tamanho: (tamanho_para_
humanos(último[1]))</p>"
            )
            pilha[-1][1] += último[1]
        página.write(f"<p style={identador(raiz)}>{raiz}</p>")
        d_tamanho = 0
```

```
        for a in arquivos:
            caminho_completo = os.path.join(raiz, a)
            d_tamanho += os.path.getsize(caminho_completo)
        pilha.append([raiz, d_tamanho])
    # Se a pilha tem mais de um elemento
    # os desempilha
    while len(pilha) > 1:
        último = pilha.pop()
        página.write(
            f"<p style={identador(último[0])}>Tamanho: ({tamanho_para_
humanos(último[1])})</p>"
        )
        pilha[-1][1] += último[1]


if len(sys.argv) < 2:
    print("Digite o nome do diretório para coletar os arquivos!")
    sys.exit(1)

diretório = sys.argv[1]

página = open("tarquivos.html", "w", encoding="utf-8")
página.write("""
<!DOCTYPE html>
<html lang="pt-BR">
<head>
<meta charset="utf-8">
<title>Arquivos</title>
</head>
<body>
""")
página.write(f"Arquivos encontrados a partir do diretório: {diretório}")
gera_listagem(página, diretório)
página.write("""
</body>
</html>
""")
página.close()
```

## EXERCÍCIO 10-01

Adicione os atributos tamanho e marca à classe Televisão. Crie dois objetos Televisão e atribua tamanhos e marcas diferentes. Depois, imprima o valor desses atributos de forma a confirmar a independência dos valores de cada instância (objeto).

```
class Televisão:
    def __init__(self):
        self.ligada = False
        self.canal = 2
        self.tamanho = 20
        self.marca = "Ching-Ling"
```

```
tv = Televisão()
tv.tamanho = 27
tv.marca = "LongDang"
tv_sala = Televisão()
tv_sala.tamanho = 52
tv_sala.marca = "XangLa"

print(f"tv tamanho={tv.tamanho} marca={tv.marca}")
print(f"tv_sala tamanho={tv_sala.tamanho} marca={tv_sala.marca}")
```

## EXERCÍCIO 10-02

Atualmente, a classe Televisão inicializa o canal com 2. Modifique a classe Televisão de forma a receber o canal inicial em seu construtor.

```
class Televisão:
    def __init__(self, canal_inicial, min, max):
        self.ligada = False
        self.canal = canal_inicial
        self.cmin = min
        self.cmax = max

    def muda_canal_para_baixo(self):
        if self.canal - 1 >= self.cmin:
            self.canal -= 1

    def muda_canal_para_cima(self):
        if self.canal + 1 <= self.cmax:
            self.canal += 1


tv = Televisão(5, 1, 99)

print(tv.canal)
```

## EXERCÍCIO 10-03

Modifique a classe Televisão de forma que, se pedirmos para mudar o canal para baixo, além do mínimo, ela vá para o canal máximo. Se mudarmos para cima, além do canal máximo, que volte ao canal mínimo. Exemplo:

```
>>> tv = Televisão(2, 10)
>>> tv.muda_canal_para_baixo()
>>> tv.canal
10
>>> tv.muda_canal_para_cima()
>>> tv.canal
```

```
2
```

```python
class Televisão:
    def __init__(self, min, max):
        self.ligada = False
        self.canal = min
        self.cmin = min
        self.cmax = max

    def muda_canal_para_baixo(self):
        if self.canal - 1 >= self.cmin:
            self.canal -= 1
        else:
            self.canal = self.cmax

    def muda_canal_para_cima(self):
        if self.canal + 1 <= self.cmax:
            self.canal += 1
        else:
            self.canal = self.cmin


tv = Televisão(2, 10)
tv.muda_canal_para_baixo()
print(tv.canal)
tv.muda_canal_para_cima()
print(tv.canal)
```

## EXERCÍCIO 10-04

Utilizando o que aprendemos com funções, modifique o construtor da classe Televisão de forma que min e max sejam parâmetros opcionais, em que min vale 2 e max vale 14, caso outro valor não seja passado.

```python
class Televisão:
    def __init__(self, min=2, max=14):
        self.ligada = False
        self.canal = min
        self.cmin = min
        self.cmax = max

    def muda_canal_para_baixo(self):
        if self.canal - 1 >= self.cmin:
            self.canal -= 1
        else:
            self.canal = self.cmax

    def muda_canal_para_cima(self):
        if self.canal + 1 <= self.cmax:
            self.canal += 1
        else:
            self.canal = self.cmin
```

```
tv = Televisão()
tv.muda_canal_para_baixo()
print(tv.canal)
tv.muda_canal_para_cima()
print(tv.canal)
```

## EXERCÍCIO 10-05

Utilizando a classe Televisão modificada no exercício anterior, crie duas instâncias (objetos), especificando o valor de min e max por nome.

```
class Televisão:
    def __init__(self, min=2, max=14):
        self.ligada = False
        self.canal = min
        self.cmin = min
        self.cmax = max

    def muda_canal_para_baixo(self):
        if self.canal - 1 >= self.cmin:
            self.canal -= 1
        else:
            self.canal = self.cmax

    def muda_canal_para_cima(self):
        if self.canal + 1 <= self.cmax:
            self.canal += 1
        else:
            self.canal = self.cmin


tv = Televisão(min=1, max=22)
tv.muda_canal_para_baixo()
print(tv.canal)
tv.muda_canal_para_cima()
print(tv.canal)

tv2 = Televisão(min=2, max=64)
tv2.muda_canal_para_baixo()
print(tv2.canal)
tv2.muda_canal_para_cima()
print(tv2.canal)
```

## EXERCÍCIO 10-06

Altere o programa de forma que a mensagem saldo insuficiente seja exibida caso haja tentativa de sacar mais dinheiro que o saldo disponível.

```
# Modifique o arquivo contas.py das listagens


class Conta:
    def __init__(self, clientes, número, saldo=0):
        self.saldo = 0
        self.clientes = clientes
        self.número = número
        self.operações = []
        self.deposito(saldo)

    def resumo(self):
        print(f"CC N°{self.número} Saldo: {self.saldo:10.2f}")

    def saque(self, valor):
        if self.saldo >= valor:
            self.saldo -= valor
            self.operações.append(["SAQUE", valor])
        else:
            print("Saldo insuficiente!")

    def deposito(self, valor):
        self.saldo += valor
        self.operações.append(["DEPÓSITO", valor])

    def extrato(self):
        print(f"Extrato CC N° {self.número}\n")
        for o in self.operações:
            print(f"{o[0]:10s} {o[1]:10.2f}")
        print(f"\n    Saldo: {self.saldo:10.2f}\n")


class ContaEspecial(Conta):
    def __init__(self, clientes, número, saldo=0, limite=0):
        Conta.__init__(self, clientes, número, saldo)
        self.limite = limite

    def saque(self, valor):
        if self.saldo + self.limite >= valor:
            self.saldo -= valor
            self.operações.append(["SAQUE", valor])
        else:
            Conta.saque(self, valor)
```

## EXERCÍCIO 10-07

Modifique o método resumo da classe Conta para exibir o nome e o telefone de cada cliente.

```
# Aqui contas.py e clientes.py foram copiados para um só arquivo.
# Esta mudança serve apenas para facilitar a visualização
# da resposta deste exercício.
```

```
class Cliente:
    def __init__(self, nome, telefone):
        self.nome = nome
        self.telefone = telefone


class Conta:
    def __init__(self, clientes, número, saldo=0):
        self.saldo = 0
        self.clientes = clientes
        self.número = número
        self.operações = []
        self.deposito(saldo)

    def resumo(self):
        print(f"CC N°{self.número} Saldo: {self.saldo:10.2f}\n")
        for cliente in self.clientes:
            print(f"Nome: {cliente.nome}\nTelefone: {cliente.telefone}\n")

    def saque(self, valor):
        if self.saldo >= valor:
            self.saldo -= valor
            self.operações.append(["SAQUE", valor])
        else:
            print("Saldo insuficiente!")

    def deposito(self, valor):
        self.saldo += valor
        self.operações.append(["DEPÓSITO", valor])

    def extrato(self):
        print(f"Extrato CC N° {self.número}\n")
        for o in self.operações:
            print(f"{o[0]:10s} {o[1]:10.2f}")
        print(f"\n    Saldo: {self.saldo:10.2f}\n")


maria = Cliente("Maria", "1243-3321")
joão = Cliente("João", "5554-3322")

conta = Conta([maria, joão], 1234, 5000)
conta.resumo()
```

## EXERCÍCIO 10-08

Crie uma nova conta, agora tendo João e José como clientes e saldo igual a 500.

```
# Aqui contas.py e clientes.py foram copiados para um só arquivo.
# Esta mudança serve apenas para facilitar a visualização
```

```
# da resposta deste exercício.


class Cliente:
    def __init__(self, nome, telefone):
        self.nome = nome
        self.telefone = telefone


class Conta:
    def __init__(self, clientes, número, saldo=0):
        self.saldo = 0
        self.clientes = clientes
        self.número = número
        self.operações = []
        self.deposito(saldo)

    def resumo(self):
        print(f"CC N°{self.número} Saldo: {self.saldo:10.2f}\n")
        for cliente in self.clientes:
            print(f"Nome: {cliente.nome}\nTelefone: {cliente.telefone}\n")

    def saque(self, valor):
        if self.saldo >= valor:
            self.saldo -= valor
            self.operações.append(["SAQUE", valor])
        else:
            print("Saldo insuficiente!")

    def deposito(self, valor):
        self.saldo += valor
        self.operações.append(["DEPÓSITO", valor])

    def extrato(self):
        print("fExtrato CC N° {self.número}\n")
        for o in self.operações:
            print(f"{o[0]:10s} {o[1]:10.2f}")
        print(f"\n    Saldo: {self.saldo:10.2f}\n")


joão = Cliente("João", "5554-3322")
josé = Cliente("José", "1243-3321")

conta = Conta([joão, josé], 2341, 500)
conta.resumo()
```

## EXERCÍCIO 10-09

Crie classes para representar estados e cidades. Cada estado tem um nome, sigla e cidades. Cada cidade tem nome e população. Escreva um programa de testes que crie três estados com algumas cidades em

cada um. Exiba a população de cada estado como a soma da população de suas cidades.

```
class Estado:
    def __init__(self, nome, sigla):
        self.nome = nome
        self.sigla = sigla
        self.cidades = []

    def adiciona_cidade(self, cidade):
        cidade.estado = self
        self.cidades.append(cidade)

    def população(self):
        return sum([c.população for c in self.cidades])


class Cidade:
    def __init__(self, nome, população):
        self.nome = nome
        self.população = população
        self.estado = None

    def __str__(self):
        return f"Cidade (nome={self.nome}, população={self.população},
estado={self.estado})"


# Populações obtidas no site da Wikipédia
# IBGE estimativa 2012
am = Estado("Amazonas", "AM")
am.adiciona_cidade(Cidade("Manaus", 1861838))
am.adiciona_cidade(Cidade("Parintins", 103828))
am.adiciona_cidade(Cidade("Itacoatiara", 89064))

sp = Estado("São Paulo", "SP")
sp.adiciona_cidade(Cidade("São Paulo", 11376685))
sp.adiciona_cidade(Cidade("Guarulhos", 1244518))
sp.adiciona_cidade(Cidade("Campinas", 1098630))

rj = Estado("Rio de Janeiro", "RJ")
rj.adiciona_cidade(Cidade("Rio de Janeiro", 6390290))
rj.adiciona_cidade(Cidade("São Gonçalo", 1016128))
rj.adiciona_cidade(Cidade("Duque de Caixias", 867067))


for estado in [am, sp, rj]:
    print(f"Estado: {estado.nome} Sigla: {estado.sigla}")
    for cidade in estado.cidades:
        print(f"Cidade: {cidade.nome} População: {cidade.população}")
    print(f"População do Estado: {estado.população()}\n")
```

## EXERCÍCIO 10-10

Modifique as classes Conta e ContaEspecial para que a operação de saque retorne verdadeiro se o saque foi efetuado e falso, caso contrário.

```
# Aqui contas.py e clientes.py foram copiados para um só arquivo.
# Esta mudança serve apenas para facilitar a visualização
# da resposta deste exercício.


class Cliente:
    def __init__(self, nome, telefone):
        self.nome = nome
        self.telefone = telefone


# Modifiaque o arquivo contas.py das listagens


class Conta:
    def __init__(self, clientes, número, saldo=0):
        self.saldo = 0
        self.clientes = clientes
        self.número = número
        self.operações = []
        self.deposito(saldo)

    def resumo(self):
        print(f"CC Nº{self.número} Saldo: {self.saldo:10.2f}")

    def saque(self, valor):
        if self.saldo >= valor:
            self.saldo -= valor
            self.operações.append(["SAQUE", valor])
            return True
        else:
            print("Saldo insuficiente!")
            return False

    def deposito(self, valor):
        self.saldo += valor
        self.operações.append(["DEPÓSITO", valor])

    def extrato(self):
        print(f"Extrato CC Nº {self.número}\n")
        for o in self.operações:
            print("f{o[0]:%10s} {o[1]:10.2f}")
        print(f"\n      Saldo: {self.saldo:%10.2f}\n")


class ContaEspecial(Conta):
    def __init__(self, clientes, número, saldo=0, limite=0):
```

```
        Conta.__init__(self, clientes, número, saldo)
        self.limite = limite

    def saque(self, valor):
        if self.saldo + self.limite >= valor:
            self.saldo -= valor
            self.operações.append(["SAQUE", valor])
            return True
        else:
            return Conta.saque(self, valor)


joão = Cliente("João", "5554-3322")
josé = Cliente("José", "1243-3321")

conta = Conta([joão, josé], 2341, 500)
conta.resumo()
print(conta.saque(1000))
print(conta.saque(100))
conta.resumo()

conta2 = ContaEspecial([josé], 3432, 50000, 10000)
conta2.resumo()
print(conta2.saque(100000))
print(conta2.saque(500))
conta2.resumo()
```

## EXERCÍCIO 10-11

Altere a classe ContaEspecial de forma que seu extrato exiba o limite e o total disponível para saque.

```
# Aqui contas.py e clientes.py foram copiados para um só arquivo.
# Esta mudança serve apenas para facilitar a visualização
# da resposta deste exercício.


class Cliente:
    def __init__(self, nome, telefone):
        self.nome = nome
        self.telefone = telefone


# Modifiaque o arquivo contas.py das listagens


class Conta:
    def __init__(self, clientes, número, saldo=0):
        self.saldo = 0
        self.clientes = clientes
        self.número = número
        self.operações = []
```

```
        self.deposito(saldo)

    def resumo(self):
        print(f"CC Nº{self.número} Saldo: {self.saldo:10.2f}")

    def saque(self, valor):
        if self.saldo >= valor:
            self.saldo -= valor
            self.operações.append(["SAQUE", valor])
            return True
        else:
            print("Saldo insuficiente!")
            return False

    def deposito(self, valor):
        self.saldo += valor
        self.operações.append(["DEPÓSITO", valor])

    def extrato(self):
        print(f"Extrato CC Nº {self.número}\n")
        for o in self.operações:
            print(f"{o[0]:10s}   {o[1]:10.2f}")
        print(f"\n      Saldo: {self.saldo:10.2f}\n")


class ContaEspecial(Conta):
    def __init__(self, clientes, número, saldo=0, limite=0):
        Conta.__init__(self, clientes, número, saldo)
        self.limite = limite

    def saque(self, valor):
        if self.saldo + self.limite >= valor:
            self.saldo -= valor
            self.operações.append(["SAQUE", valor])
            return True
        else:
            return Conta.saque(self, valor)

    def extrato(self):
        Conta.extrato(self)
        print(f"\n     Limite: {self.limite:10.2f}\n")
        print(f"\n Disponivel: {self.limite + self.saldo:10.2f}\n")


josé = Cliente("José", "1243-3321")

conta = ContaEspecial([josé], 3432, 50000, 10000)
conta.extrato()
```

## EXERCÍCIO 10-12

Observe o método saque das classes Conta e ContaEspecial. Modifique o método saque da classe Conta de forma que a verificação da possibilidade de saque seja feita por um novo método, substituindo a condição atual. Esse novo método retornará verdadeiro se o saque puder ser efetuado, e falso, caso contrário. Modifique a classe ContaEspecial de forma a trabalhar com esse novo método. Verifique se você ainda precisa trocar o método saque de ContaEspecial ou apenas o novo método criado para verificar a possibilidade de saque.

```
# Aqui contas.py e clientes.py foram copiados para um só arquivo.
# Esta mudança serve apenas para facilitar a visualização
# da resposta deste exercício.


class Cliente:
    def __init__(self, nome, telefone):
        self.nome = nome
        self.telefone = telefone


# Modifiaque o arquivo contas.py das listagens


class Conta:
    def __init__(self, clientes, número, saldo=0):
        self.saldo = 0
        self.clientes = clientes
        self.número = número
        self.operações = []
        self.deposito(saldo)

    def resumo(self):
        print(f"CC Nº{self.número} Saldo: {self.saldo:10.2f}")

    def pode_sacar(self, valor):
        return self.saldo >= valor

    def saque(self, valor):
        if self.pode_sacar(valor):
            self.saldo -= valor
            self.operações.append(["SAQUE", valor])
            return True
        else:
            print("Saldo insuficiente!")
            return False

    def deposito(self, valor):
        self.saldo += valor
        self.operações.append(["DEPÓSITO", valor])

    def extrato(self):
        print(f"Extrato CC Nº {self.número}\n")
```

```
        for o in self.operações:
            print(f"{o[0]:10s}   {o[1]:10.2f}")
        print(f"\n      Saldo: {self.saldo:10.2f}\n")


class ContaEspecial(Conta):
    def __init__(self, clientes, número, saldo=0, limite=0):
        Conta.__init__(self, clientes, número, saldo)
        self.limite = limite

    def pode_sacar(self, valor):
        return self.saldo + self.limite >= valor

    def extrato(self):
        Conta.extrato(self)
        print(f"\n     Limite: {self.limite:10.2f}\n")
        print(f"\n Disponivel: {self.limite + self.saldo:%10.2f}\n")


# Veja que com o método pode_sacar de ContaEspecial
# nem precisamos escrever um método especial de saque!

josé = Cliente("José", "1243-3321")

conta = ContaEspecial([josé], 3432, 5000, 1000)
conta.extrato()
conta.saque(6000)
conta.saque(3000)
conta.saque(1000)
conta.extrato()
```

## EXERCÍCIO 11-01

Faça um programa que crie o banco de dados preços.db com a tabela preços para armazenar uma lista de preços de venda de produtos. A tabela deve conter o nome do produto e seu respectivo preço. O programa também deve inserir alguns dados para teste.

```
import sqlite3
from contextlib import closing

with sqlite3.connect("precos.db") as conexao:
    with closing(conexao.cursor()) as cursor:
        cursor.execute("""
                create table preços(
                    nome text,
                    preço numeric)
                """)
        cursor.execute(
            """
                insert into preços (nome, preço)
                    values(?, ?)
```

```
                """,
            ("Batata", "3.20"),
        )
        cursor.execute(
            """
                insert into preços (nome, preço)
                    values(?, ?)
                """,
            ("Pão", "1.20"),
        )
        cursor.execute(
            """
                insert into preços (nome, preço)
                    values(?, ?)
                """,
            ("Mamão", "2.14"),
        )
```

## EXERCÍCIO 11-02

Faça um programa para listar todos os preços do banco preços.db.

```
import sqlite3
from contextlib import closing

with sqlite3.connect("precos.db") as conexao:
    with closing(conexao.cursor()) as cursor:
        cursor.execute("""select * from preços""")
        for resultado in cursor.fetchall():
            print("Nome: {0:30s} Preço: {1:6.2f}".format(*resultado))
```

## EXERCÍCIO 11-03

Escreva um programa que realize consultas do banco de dados preços. db, criado no Exercício 11.1. O programa deve perguntar o nome do produto e listar seu preço.

```
import sqlite3
from contextlib import closing

with sqlite3.connect("precos.db") as conexao:
    with closing(conexao.cursor()) as cursor:
        while True:
            nome = input("Nome do produto a pesquisar [em branco sai]: ")
            if not nome:
                break
            cursor.execute("""select * from preços where nome = ?""", (nome,))
            achados = 0
            for resultado in cursor.fetchall():
                print("Nome: {0:30s} Preço: {1:6.2f}".format(*resultado))
```

```
                achados += 1
            if achados == 0:
                print("Não encontrado.")
            else:
                print("{} produto(s) encontrado(s).".format(achados))
```

## EXERCÍCIO 11-04

Modifique o programa do Exercício 11.3 de forma a perguntar dois valores e listar todos os produtos com preços entre esses dois valores.

```
import sqlite3
from contextlib import closing

with sqlite3.connect("precos.db") as conexao:
    with closing(conexao.cursor()) as cursor:
        preço1 = input("Digite o menor preço a listar: ")
        preço2 = input("Digite o maior preço a listar: ")

        cursor.execute(
            """select * from preços
                        where preço >= ? and preço  <= ?""",
            (preço1, preço2),
        )
        achados = 0
        for resultado in cursor.fetchall():
            print("Nome: {0:30s} Preço: {1:6.2f}".format(*resultado))
            achados += 1
        if achados == 0:
            print("Não encontrado.")
        else:
            print("{} produto(s) encontrado(s).".format(achados))
```

## EXERCÍCIO 11-05

Escreva um programa que aumente o preço de todos os produtos do banco preços.db em 10%.

```
import sqlite3
from contextlib import closing

with sqlite3.connect("precos.db") as conexao:
    with closing(conexao.cursor()) as cursor:
        cursor.execute("""update preços
                        set preço = preço * 1.1""")
        # Mostra os novos preços
        cursor.execute("""select * from preços""")
        for resultado in cursor.fetchall():
            print("Nome: {0:30s} Preço: {1:6.2f}".format(*resultado))
```

## EXERCÍCIO 11-06

Escreva um programa que pergunte o nome do produto e um novo preço. Usando o banco preços.db, atualize o preço desse produto no banco de dados.

```
import sqlite3
from contextlib import closing

with sqlite3.connect("precos.db") as conexao:
    with closing(conexao.cursor()) as cursor:
        nome = input("Digite o nome do produto a alterar o preço: ")

        cursor.execute(
            """select * from preços
                        where nome = ?""",
            (nome,),
        )

        resultado = cursor.fetchone()
        if resultado:
            print("Nome: {0:30s} Preço: {1:6.2f}".format(*resultado))
            novo_preço = input("Digite o novo preço: ")
            cursor.execute(
                """update preços
                            set preço = ?
                            where nome = ?""",
                (novo_preço, nome),
            )
        else:
            print("Não encontrado.")
```