Stenio Longo Araújo

Exercícios resolvidos utilizando



Python

UESB
Universidade Estadual do Sudoeste da Bahia

EDIÇÕES UESB

Série Textos Didáticos, v. 35

UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA

Reitor
Prof. Dr. Luiz Otávio de Magalhães

Vice-Reitor
Prof. Dr. Marcos Henrique Fernandes

Pró-Reitora de Extensão e Assuntos Comunitários (PROEX)
Profª Drª Gleide Magali Lemos Pinheiro

Diretor da Edições UESB
Cássio Marcilio Matos Santos

Editor
Yuri Chaves Souza Lima

COMITÊ EDITORIAL

Presidente
Profª Drª Gleide Magali Lemos Pinheiro (PROEX)

Representantes dos Departamentos/Áreas de Conhecimento
Profª Drª Adriana Dias Cardoso (DFZ/VC)
Profª Drª Alba Benemérita Alves Vilela (DS II/Jequié)
Prof. Dr. Prof. Cezar Augusto Casotti (DS I Jequié)
Profª Drª Delza Rodrigues de Carvalho (DCSA/VC)
Prof. Drª Jose Antonio Gonçalves dos Santos (DCSA/VC)
Prof. Dr. Flávio Antônio Fernandes Reis (DELL/VC)
Prof. Dr. José Rubens Mascarenhas de Almeida (DH/VC)
Prof. Dr. Luciano Brito Rodrigues (DTRA/Itapetinga)

Representantes da Edições UESB
Esp. Cássio Marcilio Matos Santos (Diretor)
Esp. Yuri Chaves Souza Lima (Editor)
Adm. Jacinto Braz David Filho (Revisor)
Dr. Natalino Perovano Filho (Portal de Periódicos)

Produção Editorial

Editoração Eletrônica
Ana Cristina Novais Menezes (DRT-BA 1613)

Revisão de linguagem
Tauana Lucena Novaes

Publicado em: Julho de 2023

SUMÁRIO

Tipos de Dados em Python.....5

Comandos, Operadores e Bibliotecas em Python 8

Exercícios Resolvidos.....13

Referências bibliográficas.....47

Copyright © 2023 by Autor
Todos os direitos desta edição são reservados a Edições UESB.
A reprodução não autorizada desta publicação, no todo ou em parte,
constitui violação de direitos autorais (Lei 9.610/98).

A691e

Araújo, Stenio Longo.
Exercícios resolvidos utilizando Python./ Stenio Longo Araújo. - -Vitória da Conquista: Edições UESB, 2023.

48 p. (Série Textos Didáticos, v. 35)

ISBN 978-65-87106-60-1

1. Linguagem de programação – Python – Exercícios. 2. Pyton básico. 3. Comando – Operadores – Biblioteca em Python. I. T.

CDD: 005.133

Catalogação na fonte: Juliana Teixeira de Assunção – CRB 5/1890
Biblioteca Universitária Professor Antonio de Moura Pereira
UESB – Campus de Vitória da Conquista

Editora filiada à:







Campus Universitário – Caixa Postal 95 – Fone: 77 3424-8716
Estrada do Bem-Querer, s/n – Módulo da Biblioteca, 1° andar
45031-900 – Vitória da Conquista – Bahia
www2.uesb.br/editora – E-mail: edicoesuesb@uesb.edu.br

Exercícios resolvidos utilizando Python

Python é uma linguagem de programação e seu uso vem crescendo em diversas universidades ao redor do mundo. No mercado de trabalho muitas organizações a utilizam para desenvolver aplicações. Sendo assim, este material, desenvolvido a partir do curso de extensão "Python Básico", tem como objetivo apresentar diversos exercícios resolvidos em Python visando auxiliar os alunos na compreensão da linguagem e de lógica de programação.

Inicialmente apresentamos um resumo sobre os principais tipos de dados, comandos e bibliotecas utilizadas, em seguida os exercícios resolvidos. Destacamos que os exercícios poderão ser estudados e ampliados em disciplinas de introdução à lógica de programação, estrutura de dados e paradigmas de linguagens de programação.

Prof. Stenio Longo Araújo
DCET/UESB

Tipos de Dados em Python

1) Números:

int: Números inteiros.

Exemplo: 5

Exemplo: 0b1111

Exemplo: 0xff

Exemplo: 0o17

float: Números de ponto flutuante.

Exemplo: 5.6

complex: Números complexos.

Exemplo: 2+3j

2) Texto:

str: Sequência de caracteres.

Exemplo: “python”

Exemplo: ‘python’

3) Booleanos:

bool: Valores booleanos.

Exemplo: True

Exemplo: False

4) Coleções:

list: Lista ordenada de elementos.

Exemplo: [1,2,3]

Exemplo: [“carlos”,”ana”,”rita”]

Exemplo: [10,”python”,30.45]

Exemplo: [[10,20], [30,40,50]]

tuple: Tupla imutável de elementos.

Exemplo: (1,2,3)

Exemplo: 1,2,3

Exemplo: (1000,”Carlos”,3459.66)

Exemplo: (100,”Carlos”,(10,12,2008))

dict: Dicionário de pares chave-valor.

Exemplo: {“nome”:”Carlos”,”idade”:45 }

set: Conjunto não ordenado de elementos.

Exemplo: {1,2,3}

5) Classe:

class: Define uma classe e permite a criação de objetos.

Exemplo:

```
class Pessoa:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade
    def imprimir(self):
        print("nome:",self.nome,"idade:",self.idade)

p1 = Pessoa("Maria", 20)
p1.imprimir()
```

6) type(var): exibir o tipo da variável var

```
s='python'
i=4
f=4.5
b=True
t=1,2,3
l=[1,2,3]
c={1,2,3}
d={"one":"um","two":"dois"}

print(type(s))
print(type(i))
print(type(f))
print(type(b))
print(type(t))
print(type(l))
print(type(c))
print(type(d))
```

7) Tipagem dinâmica: Python não exige declaração de tipo de dados. O tipo pode mudar durante a execução do programa.

```
x="python"
print(type(x))
x=123
print(type(x))
```

Comandos, Operadores e Bibliotecas em Python

1) Entrada/Saída: input e print

Exemplo:

```
nome = input("Digite o seu nome: ")
print("Nome:", nome)
pi=3.1415
print(f'Valor de pi aproximado é {pi:.3f}')
```

2) Estruturas de Seleção e Repetição: if, if/else, if/elif/else, for, while

Exemplo:

```
idade = 18
if idade >= 18:
   print("Você é maior de idade.")
```

Exemplo:

```
idade = 16
if idade >= 18:
   print("Você é maior de idade.")
else:
   print("Você é menor de idade.")
```

Exemplo:

```
numero = 10
if numero > 100:
   print("maior do que 100")
elif numero == 100:
   print("igual a 100")
else:
   print("menor do que 100")
```

Exemplo:

```
nomes = ["Rita", "Maria", "Carol","Mariana","Joana"]
for nome in nomes:
   print(nome)
```

Exemplo:

```
for numero in range(1, 60):
   print(numero)
```

Exemplo:

```
contador = 1
while contador <= 20:
   print(contador)
   contador += 1
```

3) Funções: def

Exemplo:

```
def imprimir():
   print("Oi")
```

Exemplo:

```
def dobrar(n):
   return 2*n
```

Exemplo:

```
def soma(a,b):
   resultado = a + b
   return resultado
```

Exemplo:

```
def calcular(a,b):
   return a+b,a-b
```

4) Importação de Módulos: import

Exemplo:

```
from math import sqrt
print(sqrt(25))
```

Exemplo:

```
import math
print(math.sqrt(16))
print(math.pi)
```

Exemplo:

```
import random
print(random.randint(1,10))
```

Exemplo:

```
import fractions
a = fractions.Fraction(1,2)
b = fractions.Fraction(3,4)
print(a+b)
```

Exemplo:

```
import decimal
D=decimal.Decimal
print((D('1.1')+D('2.2')-D('3.3'))*1000000)
```

5) Operadores Aritméticos: +, -, *, /, %, **, //

Exemplo:

```
a = 10
b = 3
x = a + b
y = a - b
z = a * b
w = a / b
r = a % b
t = a ** b
u = a // b
print("Soma:", x)
print("Subtração:", y)
print("Multiplicação:", z)
print("Divisão:", w)
print("Resto da divisão:", r)
print("Potência:", t)
print("Quociente da divisão:",u)
```

6) Operadores de Comparação: ==, !=, >, <, >=, <=

Exemplos:

```
a = 10
b = 5
x = a == b
y = a != b
z = a > b
w = a < b
t = a >= b
u = a <= b
print("Igual:", x)
print("Diferente:", y)
print("Maior:", z)
print("Menor:", w)
print("Maior ou igual:", t)
print("Menor ou igual:", u)
```

7) Operadores Lógicos: and, or, not

Exemplos:

```
a = True
b = False
x = a and b
y = a or b
z = not a
print("AND lógico:", x)
print("OR lógico:", y)
print("NOT lógico:", z)
```

8) Tratamento de Exceção: try/except

Exemplo:

```
try:
    n1 = int(input("Digite o primeiro número: "))
    n2 = int(input("Digite o segundo número: "))
    resultado = n1 / n2
    print("O resultado da divisão é:", resultado)
except ValueError:
    print("Erro de Entrada inválida")
except ZeroDivisionError:
    print("Erro de Divisão por zero")
```

9) Arquivos:

open: Abre um arquivo.
read: Lê o conteúdo de um arquivo.
write: Escreve conteúdo em um arquivo.
close: Fecha um arquivo.

Exemplo:

```
arquivo = open("dados.txt", "r",encoding="utf-8")
conteudo = arquivo.read()
print(conteudo)
arquivo.close()
```

Exemplo:

```
arquivo=open("nomes.txt","w")
arquivo.write("Carlos\n")
arquivo.write("Maria\n")
arquivo.write("Joana\n")
arquivo.close()
```

10) Bibliotecas

re: Biblioteca para manipulação de expressões regulares.
matplotlib: Biblioteca para criação de gráficos e visualização de dados.
turtle: Biblioteca para criação de gráficos baseada em tartarugas.
tkinter: Biblioteca para criação de interfaces gráficas
numpy: Biblioteca para computação numérica

11) Instalação de bibliotecas utilizando comando pip

Exemplos:

```
pip install matplotlib
pip install tkinter
pip install numpy
```

12) Para obter ajuda no Python digite help() no console da IDLE

Exemplo:

```
help(print)
```

Exemplo:
import math
help(math.sqrt)

13) Comentários: Comentários de um linha iniciam com o caracter #, e se estendem até o final da linha. Comentários de várias linhas são delimitados por 3 aspas simples ou duplas.

Exemplo:
Programa para calcular o fatorial de um número

Exemplo:
‘’’
Programa em Python
Hello World!!!
‘’’

Exemplo:
“””
Programa em Python
Hello World!!!
“””

Exercícios Resolvidos

1) Elaborar um programa Python para somar dois números.

```
# Somar dois números inteiros
# Entrada dos dados
n1 = int(input("Digite o primeiro número: "))
n2 = int(input("Digite o segundo número: "))
# Processamento dos dados
soma = n1 + n2
# Impressão do resultado
print("soma:", soma)
```

2) Elaborar um programa Python para somar os digitos de um número menor que 100.

```
# Somar os digítos de um número menor que 100
numero = int(input("Digite um número menor que 100: "))
if numero >= 100:
   print("O número deve ser menor que 100.")
else:
   dezena = numero // 10
   unidade = numero % 10
```

```
    soma = dezena + unidade
    print("soma:", soma)
```

3) Elaborar um programa Python para calcular a área de um triângulo.

```
# Calcular a área de um triângulo
base = float(input("Digite a base do triângulo: "))
altura = float(input("Digite a altura do triângulo: "))
area = (base * altura) / 2
print("Área:", area)
```

4) Elaborar um programa Python para imprimir os divisores de um número.

```
# Calcular os divisores de um número
n = int(input("Digite um número: "))
print("Os divisores de", n, "são:")
for i in range(1, n + 1):
    if n % i == 0:
        print(i,end=" ")
```

5) Elaborar um programa Python para verificar se um número é par ou ímpar.

```
# Verificar se um número é par ou ímpar
n = int(input("Digite um número: "))
if n % 2 == 0:
    print("Par")
else:
    print("Ímpar")
```

6) Elaborar um programa Python para calcular as raízes da equação do segundo grau.

```
# Calcular as raízes de uma equação do segundo grau
import math
import cmath

# Entrada dos dados
a = float(input("a:"))
b = float(input("b:"))
c = float(input("c:"))
if a==0:
    print("Não é uma equação do segundo grau!!!")
else:
    delta = b**2 - 4*a*c
```

```
    if delta < 0: # Raízes complexas
        x1= (-b + cmath.sqrt(delta))/(2*a)
        x2= (-b - cmath.sqrt(delta))/(2*a)
        print("Raizes complexas:",x1," e ",x2)
    elif delta == 0: # Apenas uma raiz
        x = -b / (2*a)
        print("Uma raiz real:", x)
    else: # Duas raízes
        x1 = (-b + math.sqrt(delta)) / (2*a)
        x2 = (-b - math.sqrt(delta)) / (2*a)
        print("Duas raízes reais:", x1, "e", x2)
```

7) Elaborar um programa Python para calcular o fatorial de um número.

```
# Calcular o fatorial de um número
num = int(input("Digite um número: "))
if num>=0:
    fatorial = 1
    for i in range(1, num + 1):
        fatorial *= i
    print("O fatorial de", num, "é", fatorial)
else:
    print("Entrada inválida!")
```

8) Elaborar um programa Python para gerar a sequência de Fibonacci.

```
# Gerar a sequência de Fibonacci
n = int(input("Digite o número: "))
print("Sequência de Fibonacci")
fib1 = 0
fib2 = 1
print(fib1,end=' ')
print(fib2,end=' ')
for i in range(2, n):
    fib = fib1 + fib2
    fib1 = fib2
    fib2 = fib
    print(fib,end=' ')
```

9) Elaborar um programa Python para imprimir a tabuada de um número.

```
# Imprimir a tabuada de um número
continua = True
```

```
while continua:
  n = int(input("Digite um número: "))
  for i in range(1, 11):
     print(n, "x", i, "=", n*i)
  resposta=input("Deseja continuar (s/n)?")
  if resposta=='n':
     continua=False
```

10) Elaborar um programa Python para ler 10 números e imprimir a soma, o maior e o menor desses números.

```
# Ler 10 números e imprimir a soma, o maior e o menor
numeros = []
for i in range(10):
   num = int(input("Digite um número: "))
   numeros.append(num)
soma = sum(numeros)
maior = max(numeros)
menor = min(numeros)
print("Soma:", soma)
print("Maior:", maior)
print("Menor:", menor)
```

11) Elaborar um programa Python para verificar se uma string é uma palindrome.

```
# Verificar se uma string é uma palíndrome
palavra = input("Digite uma palavra: ")
inversa = string[::-1]
if palavra == inversa:
   print("É uma palindrome.")
else:
   print("Não é uma palindrome.")
```

12) Elaborar um programa Python para intercalar duas listas ordenadas.

```
# Intercalar duas listas ordenadas

# Entrada dos dados da lista 1
lista1=list()
i=1
while i<=10:
   elem = int(input("Digite um elemento da lista:"))
   lista1.append(elem)
   i+=1
```

```
print(lista1)
lista1ordenada = sorted(lista1)
print(lista1ordenada)

# Entrada dos dados da lista 2
lista2=list()
i=1
while i<=10:
   elem = int(input("Digite um elemento da lista:"))
   lista2.append(elem)
   i+=1
print(lista2)
lista2ordenada = sorted(lista2)
print(lista2ordenada)

# Processamento dos dados
intercalada = []
i = j = 0
while i < len(lista1ordenada) and j < len(lista2ordenada):
   if lista1ordenada[i] < lista2ordenada[j]:
      intercalada.append(lista1ordenada[i])
      i += 1
   else:
      intercalada.append(lista2ordenada[j])
      j += 1
intercalada += lista1ordenada[i:]
intercalada += lista2ordenada[j:]

# Resultado final
print(intercalada)
```

13) Elaborar um programa Python para calcular a soma de 1 até 50.

```
# Somar os valores de 1 até 50, inclusive.
soma = 0
for i in range(1, 51):
   soma += i
print("Soma:", soma)
```

14) Elaborar um programa Python para criptografar uma string utilizando a cifra de César.

```
# Criptografia utilizando a cifra de César
deslocamento = int(input("Digite o deslocamento: "))
```

```
texto = input("Digite o texto a ser criptografado: ")
texto_criptografado = ""
for letra in texto:
    if letra.isupper():
        letra_criptografada = chr((ord(letra.lower()) + deslocamento - 97) % 26 + 65)
    elif letra.islower():
        letra_criptografada = chr((ord(letra) + deslocamento - 97) % 26 + 97)
    else:
        letra_criptografada = letra
    texto_criptografado += letra_criptografada
print("Texto criptografado:", texto_criptografado)
```

15) Elaborar um programa Python para ler uma temperatura em Fahrenheint e converter para Celsius.

```
# Converter de fahrenheit para celsius
fahrenheit = float(input("Digite a temperatura em Fahrenheit: "))
celsius = (fahrenheit - 32) * 5/9
print("Temperatura em Celsius é:", celsius)
```

16) Elaborar uma função para calcular o maior de três números.

```
# Calcular o maior de três números
def maior3(a, b, c):
    if a >= b and a >= c:
        return a
    elif b >= c:
        return b
    else:
        return c

# Entrada dos dados
n1=int(input("Digite um número:"))
n2=int(input("Digite um número:"))
n3=int(input("Digite um número:"))
# Processamento dos dados
resultado = maior3(n1,n2,n3)
# Impressão do resultado
print(resultado)
```

17) Elaborar uma função para dobrar os elementos de uma lista.

```
# Dobrar os elementos de uma lista
def dobrar_lista(lista):
```

```
    nova_lista = []
    for elemento in lista:
        novo_elemento = elemento * 2
        nova_lista.append(novo_elemento)
    return nova_lista

# Entrada dos dados
lista=list()
i=1
while i<=10:
    elem = int(input("Digite um elemento da lista:"))
    lista.append(elem)
    i+=1

print(lista)
nova_lista = dobrar_lista(lista)
print(nova_lista)
```

18) Elaborar uma função em Python para computar o maior e o menor elemento de uma lista.

```
# Calcular o maior e o menor elemento de uma lista
def maior_menor(lista):
    maior = lista[0]
    menor = lista[0]
    for elemento in lista:
        if elemento > maior:
            maior = elemento
        elif elemento < menor:
            menor = elemento
    return maior, menor

# Entrada dos dados
lista=list()
i=1
while i<=10:
    elem = int(input("Digite um elemento da lista:"))
    lista.append(elem)
    i+=1
print(lista)

# Resultados
maior, menor = maior_menor(lista)
print("Maior:", maior)
```

```
print("Menor:", menor)
```

19) Elaborar uma função recursiva em Python para calcular o MDC de dois números.

```
# Calcular o mdc de dois números utilizando o algoritmo de Euclides
def mdc(a, b):
    if b == 0:
        return a
    else:
        return mdc(b, a % b)

# Entrada dos dados
num1 = int(input("Digite um número:"))
num2 = int(input("Digite outro número:"))
# Processamento dos dados
resultado = mdc(num1, num2)
# Resultado
print("MDC:", resultado)
```

20) Elaborar um função em Python para ordenar uma lista utilizando a ordenação por inserção (Insertion Sort).

```
# Ordenar uma lista utilizando ordenação por inserção
def insertion_sort(lista):
    for i in range(1, len(lista)):
        chave = lista[i]
        j = i - 1
        while j >= 0 and chave < lista[j]:
            lista[j + 1] = lista[j]
            j -= 1
        lista[j + 1] = chave

# Entrada dos dados
lista=list()
i=1
while i<=10:
    elem = int(input("Digite um elemento da lista:"))
    lista.append(elem)
    i+=1

print(lista)
insertion_sort(lista)
print(lista)
```

21) Elaborar um função para retornar o último elemento de uma lista.

```
# Obter o último elemento de uma lista
def obter_ultimo_elemento(lista):
    if lista:
        return lista[-1]
    else:
        return None

# Entrada dos Dados
lista=list()
i=1
while i<=5:
    elem = int(input("Digite um elemento da lista:"))
    lista.append(elem)
    i+=1
print(lista)

ultimo_elemento = obter_ultimo_elemento(lista)
print("Último elemento da lista:", ultimo_elemento)
```

22) Elaborar em Python uma agenda de Contatos. Um contato tem os seguintes atributos: nome, telefone e e-mail.

```
class Contato:
    def __init__(self, nome, endereco, email):
        self.nome = nome
        self.endereco = endereco
        self.email = email

# Uma agenda tem vários contatos
class Agenda:
    def __init__(self):
        self.contatos = []
    def adicionar_contato(self, contato):
        self.contatos.append(contato)
    def remover_contato(self, contato):
        self.contatos.remove(contato)
    def listar_contatos(self):
        for contato in self.contatos:
            print("Nome:", contato.nome)
            print("Endereço:", contato.endereco)
            print("E-mail:", contato.email)
# Criando objetos
```

```
agenda = Agenda()
contato1 = Contato(“João”, “Rua A, 123”, “joao@example.com”)
contato2 = Contato(“Maria”, “Rua B, 456”, “maria@example.com”)
agenda.adicionar_contato(contato1)
agenda.adicionar_contato(contato2)
agenda.listar_contatos()
agenda.remover_contato(contato1)
agenda.listar_contatos()
```

23) Elaborar uma função para converter de kilometros para metros.

```
# Converter de km para metros
def converter_quilometros_para_metros(quilometros):
    metros = quilometros * 1000
    return metros

try:
 quilometros = float(input(“Digite a distância em quilômetros: “))
 metros = converter_quilometros_para_metros(quilometros)
 print(“metros:”, metros)
except ValueError:
    print(“Entrada inválida!”)
```

24) Elaborar uma função para calcular o fatorial de um número utilizando recursão com cauda.

```
# Calcular o fatorial de um número
def fatorial(n, resultado=1):
    “”
    Função que lê um número inteiro n >= 0 e imprime n!
    “”
    if n == 0 or n == 1:  # caso base
        return resultado
    else:  # passo recursivo
        return fatorial(n - 1, n * resultado)

# Função principal
def main():
    n = int(input(“Digite um número inteiro: “))
    resultado = fatorial(n)
    print(20*”#”)
    print(“Fatorial:”, resultado)
    print(20*”#”)

main()
```

25) Elaborar um programa Python para imprimir os números ímpares entre 1 e 100, inclusive.

```
# Imprimir números ímpares de 1 até 100, inclusive.
for numero in range(1, 101):
   if numero % 2 != 0:
      print(numero,end=" ")
```

26) Elaborar um programa Python que leia uma lista com 10 inteiros e imprima a soma e média dos números

```
# Imprimir a soma e a média de uma lista com 10 inteiros
numeros = []
for i in range(10):
   try:
     numero = int(input("Digite um número inteiro: "))
     numeros.append(numero)
   except ValueError:
     print("Entrada inválida!!!")
soma = sum(numeros)
media = soma / len(numeros)
print("Soma:", soma)
print("Média:", media)
```

27) Elaborar um programa Python para realizar a soma dos elementos de uma matriz N por N.

```
# Somar os elementos de uma matriz NxN
def somar_elementos_matriz(matriz):
   soma = 0
   for linha in matriz:
      for elemento in linha:
         soma += elemento
   return soma

# Entrada dos dados
n = int(input("Digite o tamanho da matriz N por N: "))
matriz = []
for i in range(n):
   linha = []
   for j in range(n):
      elemento = int(input("Digite um elemento da matriz:"))
      linha.append(elemento)
   matriz.append(linha)

# Processamento dos dados
```

```
soma = somar_elementos_matriz(matriz)

# Impressão do resultado
print("*"*20)
print("Soma:", soma)
print("*"*20)
```

28) Elaborar um programa Python para contar o número de vogais e espaços de uma string.

```
# Contar número de vogais e espaços de uma string
def contar_espacos_vogais(string):
    espacos = 0
    vogais = 0
    for caractere in string:
        if caractere.isspace():
            espacos += 1
        elif caractere.lower() in 'aeiou':
            vogais += 1
    return espacos, vogais

texto = input("Digite uma string: ")

num_espacos, num_vogais = contar_espacos_vogais(texto)

print("*"*25)
print("Número de espaços:", num_espacos)
print("Número de vogais:", num_vogais)
print("*"*25)
```

29) Elaborar um programa Python para ler uma data no formato "DDMMAAAA" e devolver o mês.

```
def obter_mes(data):
    dia = int(data[0:2])
    mes = int(data[2:4])
    ano = int(data[4:8])

    meses = [
        "Janeiro", "Fevereiro", "Março", "Abril",
        "Maio", "Junho", "Julho", "Agosto",
        "Setembro", "Outubro", "Novembro", "Dezembro"
    ]
    if mes >= 1 and mes <= 12:
        return meses[mes - 1]
```

```
    else:
        return "Mês inválido"

data = input("Digite a data no formato DDMMAAAA: ")
mes = obter_mes(data)
print("O mês correspondente é:", mes)
```

30) Elaborar um programa Python para ler um arquivo texto contendo uma lista de nomes e gerar outro arquivo contendo a lista de nomes ordenada.

```
nome_arquivo_entrada = input("Digite o nome do arquivo de entrada: ")
nome_arquivo_saida = input("Digite o nome do arquivo de saída para a lista ordenada: ")
nomes = []

try:
    with open(nome_arquivo_entrada, 'r') as arquivo_entrada:
        for linha in arquivo_entrada:
            nome = linha.strip()
            nomes.append(nome)
except FileNotFoundError:
    print(f"Arquivo '{nome_arquivo_entrada}' não encontrado.")

nomes_ordenados = sorted(nomes)

try:
    with open(nome_arquivo_saida, 'w') as arquivo_saida:
        for nome in nomes_ordenados:
            arquivo_saida.write(nome + "\n")
    print(f"Lista ordenada gerada no arquivo '{nome_arquivo_saida}'.")
except:
    print(f"Erro ao escrever no arquivo '{nome_arquivo_saida}'. ")
```

31) Elaborar um programa Python para calcular a distância entre dois pontos no eixo cartesiano

```
import math

def calcular_distancia(x1, y1, x2, y2):
    distancia = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
    return distancia

# Entrada dos dados
x1 = float(input("Digite a coordenada x do primeiro ponto: "))
y1 = float(input("Digite a coordenada y do primeiro ponto: "))
```

```
x2 = float(input("Digite a coordenada x do segundo ponto: "))
y2 = float(input("Digite a coordenada y do segundo ponto: "))
# Processamento dos dados
distancia = calcular_distancia(x1, y1, x2, y2)
# Resultado
print("Distância entre os pontos:", distancia)
```

32) Elaborar um programa Python para calcular a soma dos elementos de uma lista utilizando recursão.

```
# Somar os elementos de uma lista
def soma_lista_recursiva(lista):
    if len(lista) == 0: # caso base
        return 0
    else:  # passo recursivo
        return lista[0] + soma_lista_recursiva(lista[1:])

# Entrada dos dados
lista=list()
i=1
while i<=10:
    elem = int(input("Digite um elemento da lista:"))
    lista.append(elem)
    i+=1
print(lista)
resultado = soma_lista_recursiva(lista)

print("*"*20)
print("Soma:", resultado)
print("*"*20)
```

33) Elaborar um programa Python para calcular a frequência das letras em um texto.

```
def calcular_frequencia_letras(texto):
    texto = texto.lower()
    frequencia_letras = {}

    for caractere in texto:
        if caractere.isalpha():
            if caractere in frequencia_letras:
                frequencia_letras[caractere] += 1
            else:
                frequencia_letras[caractere] = 1
```

```
    return frequencia_letras

texto = input("Digite um texto: ")
frequencia = calcular_frequencia_letras(texto)

for letra in sorted(frequencia):
   print(f"A letra '{letra}' aparece {frequencia[letra]} vezes.")
```

34) Elaborar um programa Python para verifica se uma matriz 3x3 é simétrica. Uma matriz é dita simétrica se ela for igual a sua transposta.

```
# Verificar se uma matriz é simétrica
def verificar_simetria(matriz):
   for i in range(3):
      for j in range(3):
         if matriz[i][j] != matriz[j][i]:
            return False
   return True

# Entrada dos dados
matriz = []
print("Digite os elementos da matriz 3x3:")
for i in range(3):
   linha = []
   for j in range(3):
      elemento = int(input(f"Elemento [{i}][{j}]: "))
      linha.append(elemento)
   matriz.append(linha)

# Resultado
if verificar_simetria(matriz):
   print("A matriz é simétrica.")
else:
   print("A matriz não é simétrica.")
```

35) Elaborar um programa Python para calcular a união e a intersecção entre dois conjuntos.

```
def calcular_uniao(conjunto1, conjunto2):
   uniao = conjunto1.union(conjunto2)
   return uniao

def calcular_intersecao(conjunto1, conjunto2):
   intersecao = conjunto1.intersection(conjunto2)
```

```
    return intersecao

conjunto1 = set()
conjunto1.add(1)
conjunto1.add(2)

conjunto2 = set()
conjunto2.add(2)
conjunto2.add(3)
conjunto2.add(4)

uniao = calcular_uniao(conjunto1, conjunto2)
intersecao = calcular_intersecao(conjunto1, conjunto2)
print(f"União: {uniao}")
print(f"Interseção: {intersecao}")
```

36) Elaborar um programa em Python que recebe uma lista de tuplas contendo o nome e a nota de cada aluno. Calcule a média das notas dos alunos.

```
def calcular_media_alunos(alunos):
    soma_notas = 0
    total_alunos = len(alunos)
    for aluno in alunos:
        nota = aluno[1]
        soma_notas += nota
    media = soma_notas / total_alunos
    return media

alunos=[]
while True:
    nome=input("Digite o nome do aluno ou fim para encerrar:")
    if nome=='fim':
        break
    nota=float(input("Digite a nota do aluno:"))
    aluno = (nome,nota)
    alunos.append(aluno)

media = calcular_media_alunos(alunos)
print(f"A média das notas dos alunos é: {media}")
```

37) Elaborar um programa Python para converter uma data no formato "DD/MM/AA" para o formato "DD-MM-AA".

```
def converter_data(data):
    dia, mes, ano = data.split("/")
    nova_data = dia + "-" + mes + "-" + ano
    return nova_data

data_original = input("Digite a data no formato DD/MM/AA: ")
data_convertida = converter_data(data_original)
print(f"A data convertida é: {data_convertida}")
```

38) Elaborar um programa Python para ler uma data no formato DD/MM/AAAA e escrever por extenso.

```
meses = {
    1: "janeiro",
    2: "fevereiro",
    3: "março",
    4: "abril",
    5: "maio",
    6: "junho",
    7: "julho",
    8: "agosto",
    9: "setembro",
    10: "outubro",
    11: "novembro",
    12: "dezembro"
}

def escrever_data(data):
    dia, mes, ano = map(int, data.split("/"))
    mes_extenso = meses[mes]
    data_extenso = f"{dia} de {mes_extenso} de {ano}"
    return data_extenso

data_original = input("Digite a data no formato DD/MM/AAAA: ")
data_extenso = escrever_data(data_original)
print(f"A data por extenso é: {data_extenso}")
```

39) Elaborar um programa em Python para ordenar uma lista utilizando o algoritmo Bubble Sort.

```
# Ordenar uma lista utilizando o método bubble sort
def bubble_sort(lista):
    n = len(lista)
```

```
    for i in range(n-1):
        for j in range(0, n-i-1):
            if lista[j] > lista[j+1]:
                lista[j], lista[j+1] = lista[j+1], lista[j]

# Entrada dos dados
lista=list()
i=1
while i<=10:
    elem = int(input("Digite um elemento da lista:"))
    lista.append(elem)
    i+=1

# Resultados
print("Lista original:")
print(lista)
bubble_sort(lista)
print("Lista ordenada:")
print(lista)
```

40) Elaborar em Python uma função recursiva para computar o maior elemento de uma lista.

```
# Calcular o maior elemento de uma lista
def encontrar_maior_elemento(lista):
    if len(lista) == 1:  # caso base
        return lista[0]
    else:   # etapa recursiva
        primeiro_elemento = lista[0]
        sublista = lista[1:]
        maior_elemento_sublista = encontrar_maior_elemento(sublista)
        if primeiro_elemento > maior_elemento_sublista:
            return primeiro_elemento
        else:
            return maior_elemento_sublista

# Entrada dos dados
lista=list()
i=1
while i<=10:
    elem = int(input("Digite um elemento da lista:"))
    lista.append(elem)
    i+=1
print(lista)
```

```
# Resultados
maior_elemento = encontrar_maior_elemento(lista)
print("O maior elemento da lista é:", maior_elemento)
```

41) Elaborar um programa Python para realizar a pesquisa binária em uma lista ordenada.

```
# Pesquisar um elemento em uma lista ordenada

def pesquisa_binaria(lista, elemento):
    esquerda = 0
    direita = len(lista) - 1
    while esquerda <= direita:
        meio = (esquerda + direita) // 2
        if lista[meio] == elemento:
            return meio
        elif lista[meio] < elemento:
            esquerda = meio + 1
        else:
            direita = meio - 1
    return -1

# Entrada dos dados

lista=list()
i=1
while i<=10:
    elem = int(input("Digite um elemento da lista:"))
    lista.append(elem)
    i+=1
print(lista)
lista_ordenada = sorted(lista)
print(lista_ordenada)
elemento = int(input("Digite o elemento para pesquisar:"))

# Resultados
indice = pesquisa_binaria(lista_ordenada, elemento)
if indice != -1:
    print("O elemento", elemento, "foi encontrado no índice", indice)
else:
    print("O elemento", elemento, "não foi encontrado na lista.")
```

42) Elaborar um programa Python para imprimir a posição de um ponto P (x,y) no eixo carteisano.

```
def imprimir_posicao(x, y):
    if x > 0 and y > 0:
        print("O ponto P está no primeiro quadrante.")
    elif x < 0 and y > 0:
        print("O ponto P está no segundo quadrante.")
    elif x < 0 and y < 0:
        print("O ponto P está no terceiro quadrante.")
    elif x > 0 and y < 0:
        print("O ponto P está no quarto quadrante.")
    elif x == 0 and y != 0:
        print("O ponto P está sobre o eixo y.")
    elif x != 0 and y == 0:
        print("O ponto P está sobre o eixo x.")
    else:
        print("O ponto P está na origem.")

x = float(input("Digite a coordenada x do ponto P: "))
y = float(input("Digite a coordenada y do ponto P: "))
imprimir_posicao(x, y)
```

43) Elaborar um programa Python para dobrar os elementos de uma lista.

```
# Dobrar os elementos de uma lista
def dobrar_elementos(lista):
    for i in range(len(lista)):
        lista[i] *= 2

# Entrada dos dados
lista=list()
i=1
while i<=10:
    elem = int(input("Digite um elemento da lista:"))
    lista.append(elem)
    i+=1

print(lista)
dobrar_elementos(lista)
print(lista)
```

44) Elaborar um programa Python para imprimir todos os números pares entre 50 e 100, inclusive.

```
# Imprimir os pares entre 50 e 100, inclusive.
def imprimir_numeros_pares(inicio, fim):
    for numero in range(inicio, fim + 1):
        if numero % 2 == 0:
            print(numero,end=" ")

print("#"*40)
print("Número pares entre 50 e 100, inclusive")
print("#"*40)
imprimir_numeros_pares(50, 100)
```

45) Elaborar um programa Python para somar todos os números pares entre 50 e 100, inclusive.

```
# Somar os pares entre 50 e 100, inclusive
def somar_numeros_pares(inicio, fim):
    soma = 0
    for numero in range(inicio, fim + 1):
        if numero % 2 == 0:
            soma += numero
    return soma

resultado = somar_numeros_pares(50, 100)
print("Soma:", resultado)
```

46) Elaborar um programa Python para computar o maior e o menor valor em um dicionário.

```
# Obter o maior e menor valor de uma dicionário
def obter_maior_menor_valor(dicionario):
    valores = list(dicionario.values())
    maior_valor = max(valores)
    menor_valor = min(valores)
    return maior_valor, menor_valor

# Pares nome-idade
dicionario = {'Carlos': 18, 'Rita': 15, 'Marcos': 20, 'Ana': 25}
maior, menor = obter_maior_menor_valor(dicionario)
print("Maior valor:", maior)
print("Menor valor:", menor)
```

47) Elaborar um programa Python para obter a moda em uma lista de inteiros. A moda representa o valor mais frequente em um conjunto de dados.

```
def calcular_moda(lista):
    frequencias = {}
    for elemento in lista:
        if elemento in frequencias:
            frequencias[elemento] += 1
        else:
            frequencias[elemento] = 1
    moda = []
    maior_frequencia = 0

    for elemento, frequencia in frequencias.items():
        if frequencia > maior_frequencia:
            moda = [elemento]
            maior_frequencia = frequencia
        elif frequencia == maior_frequencia:
            moda.append(elemento)

    return moda, maior_frequencia

# Entrada dos dados
lista=list()
i=1
while i<=10:
    elem = int(input("Digite um elemento da lista:"))
    lista.append(elem)
    i+=1
print(lista)

# Resultados
moda, frequencia = calcular_moda(lista)
print("Lista:", lista)
print("Moda:", moda)
print("Frequência:", frequencia)
```

48) Elaborar um programa remover os nomes repetidos de uma lista de nomes.

```
# Remover elementos repetidos de uma lista
def remover_repetidos(lista):
    lista_sem_repeticao = list(set(lista))
    return lista_sem_repeticao
# Entrada dos dados
nomes =list()
```

```
i=1
while i<=10:
   elem = input("Digite um elemento da lista:")
   nomes.append(elem)
   i+=1

# Resultados
nomes_sem_repeticao = remover_repetidos(nomes)
print(nomes_sem_repeticao)
```

49) Elaborar um programa Python para receber uma lista de inteiros e separar os números positivos e negativos da lista.

```
def separar_positivos_negativos(lista):
   positivos = []
   negativos = []

   for numero in lista:
      if numero >= 0:
         positivos.append(numero)
      else:
         negativos.append(numero)
   return positivos, negativos

# Entrada dos dados
numeros =list()
i=1
while i<=10:
   elem = int(input("Digite um elemento da lista:"))
   numeros.append(elem)
   i+=1

# Processamento dos dados
positivos, negativos = separar_positivos_negativos(numeros)
# Resultados
print("Positivos:", positivos)
print("Negativos:", negativos)
```

50) Elaborar um programa Python para gerar a tabela verdade do "ou exclusivo".

```
# Gerar tabela verdade do ou exclusivo
def tabela_verdade_xor():
   valores = [False, True]
```

```
    print(f"|  A  |  B  |  A xor B |")
    print(29*'-')
    for A in valores:
        for B in valores:
            resultado = A ^ B
            print(f"| {A:^5} | {B:^5} | {resultado:^9} |")

print('Tabela Verdade do Ou exclusivo')
tabela_verdade_xor()
```

51) Elaborar um programa em Python que dada uma frase imprima os três primeiros caracteres.

```
frase = input('Digite uma frase:')
tres_primeiros = frase[:3]
print(tres_primeiros)
```

52) Elaborar um programa em Python que dada uma frase imprima os quatro últimos caracteres.

```
frase = input('Digite uma frase:')
ultimos_quatro = frase[-4:]
print(ultimos_quatro)
```

53) Elaborar um programa Python que dada uma frase imprima os caracteres em índices pares.

```
frase = input('Digite uma frase:')
pares = frase[::2]
print(pares)
```

54) Elaborar um programa em Python que dada uma frase imprima os caracteres de indices ímpares.

```
frase = input('Digite uma frase:')
impares = frase[1::2]
print(impares)
```

55) Elaborar um programa Python que dada uma frase imprima a frase invertida.

```
frase = input('Digite uma frase:')
invertida = frase[::-1]
print(invertida)
```

56) Elaborar um programa em Python que dada uma lista de listas imprima o valor do elemento da segunda linha e terceira coluna

```
# Entrada de dados
linhas = int(input("Digite a quantidade de linhas:"))
colunas = int(input("Digite a quantidade de colunas:"))

matriz=[]
for i in range(linhas):
    linha=[]
    for j in range(colunas):
        elemento=int(input('Digite um valor inteiro:'))
        linha.append(elemento)
    matriz.append(linha)

try:
   elemento = matriz[1][2]
   print('Elemento da segunda linha e terceira coluna:',elemento)
except IndexError:
   print('Elemento não existe!')
```

57) Elaborar um programa Python que dada uma lista de listas imprima a segunda linha completa.

```
matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
segunda_linha = matriz[1]
print(segunda_linha)
```

58) Elaborar um programa Python utilizando herança para representar uma classe Pessoa com o atributo nome e uma classe Professor com os atributos nome e titulação.

```
# Definição da classe Pessoa

class Pessoa:
    def __init__(self, nome):
        self.nome = nome
    def __str__(self):
        return "Nome:"+ self.nome

# Definição da classe Professor utilizando herança simples (Professor é uma Pessoa)

class Professor(Pessoa):
    def __init__(self, nome, titulacao):
        super().__init__(nome)
        self.titulacao = titulacao
    def __str__(self):
        return "Nome:"+self.nome+" Titulação:"+self.titulacao
```

```
# Criação dos objetos
pessoa = Pessoa("Maria")
professor = Professor("Roque", "Doutor")
print(pessoa)
print(professor)
```

59) Elaborar um programa Python para desenhar um quadrado utilizando o módulo turtle.

```
import turtle

# Desenhar quadrado utilizando módulo turtle
def desenhar_quadrado(lado):
    turtle.forward(lado)
    turtle.right(90)
    turtle.forward(lado)
    turtle.right(90)
    turtle.forward(lado)
    turtle.right(90)
    turtle.forward(lado)

desenhar_quadrado(100)
turtle.done()
```

60) Elaborar um programa Python para desenhar um triângulo utilizando o módulo turtle

```
import turtle.

# Desenhar triângulo utilizando módulo turtle
def desenhar_triangulo(lado):
    turtle.forward(lado)
    turtle.left(120)
    turtle.forward(lado)
    turtle.left(120)
    turtle.forward(lado)

desenhar_triangulo(100)
turtle.done()
```

61) Elaborar um programa Python para desenhar um hexágono utilizando o módulo turtle.

```
import turtle
```

```
# Desenhar hexágono utilizando módulo turtle
def desenhar_hexagono(lado):
    for _ in range(6):
        turtle.forward(lado)
        turtle.right(60)

# Entrada dos dados
velocidade = int(input("Digite a velocidade (0-10):"))
turtle.speed(velocidade)
tamanho = int(input("Digite o tamanho da caneta:"))
turtle.pensize(tamanho)
cor = input("Digite a cor:")
turtle.color(cor)
lado = int(input("Digite o tamanho do lado:"))
desenhar_hexagono(lado)
turtle.done()
```

62) Elaborar um programa em Python para somar dois números utilizando uma janela do módulo tkinter.

```
import tkinter as tk

def somar():
    try:
     num1 = float(entry_num1.get())
     num2 = float(entry_num2.get())
     resultado = num1 + num2
     label_resultado.config(text=f"Resultado: {resultado}")
    except ValueError:
     label_resultado.config(text="Entrada inválida!!!")

// Criar janela e ajustar suas propriedades
janela = tk.Tk()
janela.title("Soma")
janela.geometry('200x200')

// Criar componentes e adicionar a janela
label_num1 = tk.Label(janela, text="Número 1:")
label_num1.pack()
entry_num1 = tk.Entry(janela)
entry_num1.pack()
```

```
label_num2 = tk.Label(janela, text="Número 2:")
label_num2.pack()
entry_num2 = tk.Entry(janela)
entry_num2.pack()

botao_somar = tk.Button(janela, text="Somar", command=somar)
botao_somar.pack()

label_resultado = tk.Label(janela, text="Resultado: ")
label_resultado.pack()

janela.mainloop()
```

63) Elaborar um programa em Python para validar um CEP utilizando expressões regulares (módulo re).

```
import re

# Validar CEP utilizando uma expressão regular
def validar_cep(cep):
    pattern = r'^\d{5}-\d{3}$'
    if re.match(pattern, cep):
        return True
    else:
        return False

cep = input("Digite o CEP no formato DDDDD-DDD: ")
if validar_cep(cep):
    print("CEP válido.")
else:
    print("CEP inválido.")
```

64) Elaborar um programa em Python para validar um CPF utilizando expressões regulares (módulo re).

```
import re

# Validar CPF utilizando uma expressão regular
def validar_cpf(cpf):
    pattern = r'^\d{3}\.\d{3}\.\d{3}-\d{2}$'
    if re.match(pattern, cpf):
        return True
    else:
```

```
        return False

cpf = input("Digite o CPF no formato DDD.DDD.DDD-DD: ")
if validar_cpf(cpf):
    print("CPF válido.")
else:
    print("CPF inválido.")
```

65) Elaborar um programa Python para criar uma prova com 5 questões objetivas e realizar a correção.

```
class Questao:
    def __init__(self, pergunta, alternativas, resposta_correta):
        self.pergunta = pergunta
        self.alternativas = alternativas
        self.resposta_correta = resposta_correta

    def verificar_resposta(self, resposta):
        return resposta == self.resposta_correta

questao1 =  Questao("Quanto é 2+2?", ["a) 4", "b) 3", "c) 5", "d) 2"],"a")
questao2 =  Questao("Quanto é 3-1?", ["a) 2", "b) 3", "c) 4", "d) 5"],"a")
questao3 = Questao("Quanto é 2*2?", ["a) 4", "b) 3", "c) 2", "d) 1"], "a")
questao4 = Questao("Quanto é 2/2?", ["a) 1", "b) 2", "c) 3", "d) 4"],  "a")
questao5 = Questao("Quanto é 2+3?", ["a) 5", "b) 4", "c) 3", "d) 2"], "a")
prova = [questao1, questao2, questao3, questao4, questao5]

acertos = 0
for i, questao in enumerate(prova):
    print(f"Questão {i+1}: {questao.pergunta}")
    for alternativa in questao.alternativas:
        print(alternativa)
    resposta = input("Digite a alternativa correta (a, b, c, d): ")
    if questao.verificar_resposta(resposta):
        acertos += 1
print(f"Você acertou {acertos} questões de um total de {len(prova)}.")
```

66) Elaborar um programa em Python para gerar o gráfico de uma função do primeiro grau utilizando o módulo matplotlib.

```
import matplotlib.pyplot as plt

def plotar_funcao(a, b):
    x = range(-10, 11)
    y = [a * xi + b for xi in x]

    plt.plot(x, y)
    plt.xlabel('x')
    plt.ylabel('y')
    plt.title('Gráfico de uma função do primeiro grau')
    plt.grid(True)
    plt.show()

def entrada():
    print("Plotar gráfico de uma função do primeiro grau")
    a = float(input("Digite o valor de a: "))
    b = float(input("Digite o valor de b: "))
    return a,b

def main():
    a,b = entrada()
    plotar_funcao(a, b)

main()
```

67) Elaborar um programa Python para gerar um gráfico de uma equação do segundo grau utilizando o módulo matplotlib.

```
import matplotlib.pyplot as plt
import numpy as np

def plot_funcao_segundo_grau(a, b, c):
    x = np.linspace(-10, 10, 400)
    y = a * x**2 + b * x + c
    plt.plot(x, y)
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.title('Gráfico da Função do Segundo Grau')
    plt.grid(True)
    plt.show()
print("Plotar gráfico de uma equação do segundo grau")
# Entrada dos dados
a = float(input("Digite o coeficiente a:"))
```

```
b = float(input("Digite o coeficiente b:"))
c = float(input("Digite o coeficiente c:"))
if a==0:
   print("Não é uma equação do segundo grau!")
else:
   plot_funcao_segundo_grau(a, b, c)
```

68) Elaborar um programa em Python para gerar um gráfico de barras utilizando a biblioteca matplotlib.

```
import matplotlib.pyplot as plt

dias = ['Segunda', 'Terça', 'Quarta', 'Quinta',
  'Sexta', 'Sábado', 'Domingo']

vendas_semana=[]

# Entrada dos Dados
for i in range(7):
   quantidade= int(input("Digite a quantidade de vendas na "+ dias[i]+":"))
   vendas_semana.append(quantidade)

# Plotar gráfico de barra
x = range(len(dias))
y = vendas_semana
plt.bar(x, y)
plt.xlabel('Dia da semana')
plt.ylabel('Vendas')
plt.title('Vendas da semana')
plt.xticks(x, dias)
plt.grid(True)
plt.show()
```

69) Elaborar um programa Python para gerar os gráficos das funções $f(x)=x$, $f(x)=x^2$, $f(x)=\log(x)$ e $f(x)=2^x$ utilizando matplotlib.

```
import numpy as np
import matplotlib.pyplot as plt

def plot_funcoes():
   x = np.linspace(0.1, 10, 100)
   # Funções
   y1 = x
   y2 = x**2
```

```
    y3 = np.log(x)
    y4 = 2**x

    # Plotar gráficos
    plt.plot(x, y1, label='f(x) = x')
    plt.plot(x, y2, label='f(x) = x^2')
    plt.plot(x, y3, label='f(x) = log(x)')
    plt.plot(x, y4, label='f(x) = 2^x')

    # Legendas
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.title('Gráfico de Funções')
    plt.legend()
    plt.grid(True)
    plt.show()

plot_funcoes()
```

70) Elaborar um programa em Python para calcular a série de Fibonacci utilizando recursão e iteração. Compare o tempo de execução e apresente os gráficos utilizando o matplotlib.

```
import time
import matplotlib.pyplot as plt

def fibonacci_recursivo(n):
    if n <= 1:
        return n
    else:
        return fibonacci_recursivo(n - 1) + fibonacci_recursivo(n - 2)

def fibonacci_iterativo(n):
    if n <= 1:
        return n
    a, b = 0, 1
    for _ in range(n - 1):
        a, b = b, a + b
    return b

def comparar_tempo_execucao(n):
    tempos_recursivo = []
    tempos_iterativo = []
```

```
    numeros = []

    for i in range(n):
        start_time = time.time()
        fibonacci_recursivo(i)
        end_time = time.time()
        tempos_recursivo.append(end_time - start_time)

        start_time = time.time()
        fibonacci_iterativo(i)
        end_time = time.time()
        tempos_iterativo.append(end_time - start_time)

        numeros.append(i)

    plt.plot(numeros, tempos_recursivo, label='Recursivo')
    plt.plot(numeros, tempos_iterativo, label='Iterativo')
    plt.xlabel('N')
    plt.ylabel('Tempo de Execução (segundos)')
    plt.title('Comparação de Tempo de Execução - Fibonacci Recursivo vs. Fibonacci Iterativo')
    plt.legend()
    plt.grid(True)
    plt.show()

n = 50
comparar_tempo_execucao(n)
```

71) Qual a saída dos programa Python abaixo?

```
s="python"
print(s[1]*s.index("n"))
```

Resposta: yyyyy

72) Qual a saída do programa Python abaixo?

```
s="linguagem python"
print(s[2]+s[-5])
```

Resposta: ny

73) Qual a saída do programa Python abaixo?

```
s="linguagem python"
```

print(s[3:8]*3)

Resposta: guageguageguage

74) Qual a saída do programa Python abaixo?

print(re.findall("\d+","rua A, numero 10, cep 34920-999"))

Resposta: ['10', '34920', '999']

75) Qual a saída do programa Python abaixo?

print ((2 << 3) + (2 << 4) - (8 >> 1))

Resposta: 44

76) Qual a saída do programa Python abaixo?

```
a = {1,2,3}
b = {3,4,5}
c = {1,7,8,9}
d = {5,6,8,9,1}
print((a-b)&(c|d))
```

Resposta: {1}

77) Qual a saída do programa Python abaixo?

print(re.findall("ab*","bbbb bab abba abab bbb"))

Resposta: ['ab', 'abb', 'a', 'ab', 'ab']

78) Qual a saída do programa Python abaixo?

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

Resposta: 6

79) Qual letra será desenhada na tela?

```
from turtle import *
```

```
t = Turtle()
t.forward(100)
t.left(180)
t.forward(50)
t.left(90)
t.forward(150)
```

Resposta: T

80) Qual a saída do programa Python?

```
def f():
    global s
    print(s)
    s = "python"
    print(s)

s = "java"
f()
print(s)
```

Resposta:
java
python
python

Referências

ALVES, Fábio Junior. *Introdução à linguagem de programação Python.* Rio de Janeiro: Ciência Moderna, 2013.

KOPEC, David. *Problemas clássicos de ciência da computação com Python.* São Paulo: Novatec, 2019.

MANZANO, José Augusto N. G. *Introdução à linguagem Python.* São Paulo: Novatec, 2018.

McKINNEY, Wes. *Python para análise de dados:* tratamento de dados com Pandas, NumPy e IPython. São Paulo: Novatec, 2021.

MENEZES, Nilo Ney Coutinho. *Introdução à programação com Python:* algoritmos e lógica de programação para iniciantes. São Paulo: Novatec, 2013.

PAYNE, Bryson. *Ensine seus filhos a programar:* um guia amigável aos pais para a programação Python. São Paulo: Novatec, 2015.

RAMALHO, Luciano. *Python fluente:* programação clara, concisa e eficaz. São Paulo: Novatec, 2015.

SHAW, Zed A. *Aprenda Python 3 do jeito certo:* uma introdução muito simples ao incrível mundo dos computadores e da codificação. Rio de Janeiro: Alta Books, 2019.