FACULDADE GRAN TIETÊ
IERT- INSTITUICOES DE ENSINO REUNIDAS TIETE LTDA
Av. 15 de Novembro, 125 – Centro
Barra Bonita - SP

PROGRAMAÇÃO PYTHON

1.º - 3.ª Engenharia da Computação/Engenharia Mecânica/Engenharia Civil

REVISÃO E TRABALHO 2

Link para envio dos códigos:
https://www.dropbox.com/request/Fwx439pN99r9564u11YT

1. O QUE É PROGRAMAÇÃO?

A programação é o processo de criação de um conjunto de instruções que um computador pode seguir para executar uma tarefa específica. É uma forma de comunicação entre humanos e máquinas, onde os seres humanos escrevem código ou programas para instruir um computador sobre como realizar uma determinada função ou resolver um problema. Vamos explorar isso com mais detalhes:

A. Instruções para Computadores:

Os computadores são máquinas extremamente poderosas, mas são essencialmente burros. Eles só podem executar tarefas para as quais são programados. Portanto, a programação é o meio pelo qual os seres humanos dão a esses computadores as instruções necessárias para realizar tarefas específicas.

B. Linguagens de Programação:

As instruções dadas aos computadores são escritas em linguagens de programação. Estas são linguagens estruturadas e formais que os computadores podem entender. Exemplos de linguagens de programação incluem Python, Java, C++, JavaScript, entre outras.

C. Algoritmos:

A programação envolve a criação de algoritmos, que são uma sequência de passos lógicos e bem definidos que descrevem como uma tarefa deve ser realizada. Algoritmos são como receitas que os computadores seguem para alcançar um resultado desejado.

D. Resolução de Problemas:

A programação é frequentemente usada para resolver problemas. Programadores identificam problemas do mundo real que podem ser resolvidos com a automação ou processamento de dados, e então escrevem programas para realizar essas tarefas.

2. POR QUE APRENDER PYTHON?

Python é uma das linguagens de programação mais populares e amplamente usadas no mundo da tecnologia da informação e do desenvolvimento de software. Existem várias razões pelas quais Python se destaca e é escolhido por muitos desenvolvedores, empresas e organizações. Aqui estão algumas das principais razões pelas quais Python é uma escolha tão popular:

A. Sintaxe Simples e Legível:

Python é conhecido por sua sintaxe limpa e legível, que se assemelha à linguagem natural. Isso torna o código Python fácil de escrever, entender e manter. É uma excelente escolha para iniciantes em programação.

B. Ampla Comunidade e Suporte:

Python possui uma comunidade de desenvolvedores ativa e expansiva. Isso significa que há uma abundância de recursos, bibliotecas e frameworks disponíveis. Se você enfrentar um problema ou tiver uma pergunta, é provável que encontre uma resposta prontamente na comunidade Python.

C. Multiplataforma:

Python é uma linguagem multiplataforma, o que significa que você pode escrever código Python em um sistema operacional e executá-lo em outro sem grandes modificações. Ele está disponível para Windows, macOS e Linux.

D. Ampla Biblioteca Padrão:

Python possui uma biblioteca padrão extensa que abrange uma ampla gama de tarefas, desde manipulação de strings até desenvolvimento web e processamento de dados. Isso economiza tempo e esforço, pois muitas funcionalidades comuns já estão disponíveis para uso imediato.

E. Facilidade de Integração:

Python é facilmente integrado com outras linguagens de programação, como C, C++, e Java. Isso é útil quando você precisa aproveitar código existente em outras linguagens ou quando deseja melhorar o desempenho de partes críticas do seu programa.

F. Suporte a Programação Orientada a Objetos:

Python suporta programação orientada a objetos (POO), o que permite a criação de código modular e reutilizável. Isso é particularmente útil para o desenvolvimento de software de grande escala.

G. Amplamente Usado em Ciência de Dados e IA:

Python é a linguagem preferida para muitos cientistas de dados e pesquisadores em aprendizado de máquina e inteligência artificial. Bibliotecas populares, como NumPy, pandas, scikit-learn e TensorFlow, tornam Python uma escolha poderosa para análise de dados e desenvolvimento de modelos de machine learning.

H. Desenvolvimento Web e Frameworks:

Python é usado no desenvolvimento web por meio de frameworks populares como Django e Flask. Esses frameworks simplificam a criação de aplicativos web robustos e escaláveis.

I. Grande Demanda no Mercado de Trabalho:

A popularidade do Python no mercado de trabalho continua a crescer, tornando-o uma habilidade valiosa para desenvolvedores. Muitas empresas estão em busca de desenvolvedores Python para uma variedade de funções.

J. Código Aberto e Gratuito:

Python é uma linguagem de código aberto, o que significa que é gratuito para uso e distribuição. Você não precisa pagar licenças para desenvolver ou distribuir software em Python.

K. Educação e Aprendizado:

Python é amplamente utilizado em ambientes educacionais e acadêmicos devido à sua facilidade de aprendizado. Muitas universidades e escolas ensinam Python como a primeira linguagem de programação para iniciantes.

3. VARIÁVEIS E TIPOS DE DADOS

Variáveis e Tipos de Dados são fundamentais em qualquer linguagem de programação, e Python não é exceção. Eles permitem que os programadores armazenem e manipulem informações de diferentes maneiras. Vamos explorar esses conceitos em detalhes:

A. Variáveis:

Variáveis são espaços de armazenamento que você usa em seu programa para guardar informações. Cada variável tem um nome único que a identifica.

Em Python, você pode criar uma variável simplesmente atribuindo um valor a ela.

B. Tipos de Dados:

Python possui diversos tipos de dados que representam diferentes tipos de informações. Alguns dos tipos de dados mais comuns incluem:

- Inteiro (int): Representa números inteiros, como -1, 0, 1, 2, 1000.
- Flutuante (float): Representa números decimais, como 3.14, -0.5, 2.0.

- String (str): Representa texto, como "Olá, mundo!" ou "Python".
- Booleano (bool): Representa valores verdadeiro (True) ou falso (False).

C. Atribuição de Variáveis:

A atribuição de variáveis é feita usando o operador de atribuição "=".

Você pode atribuir valores de diferentes tipos de dados a uma variável.

D. Tipagem Dinâmica:

Python é uma linguagem de tipagem dinâmica, o que significa que você não precisa declarar o tipo de uma variável explicitamente. O tipo é determinado automaticamente pelo valor atribuído.

E. Conversão de Tipos (Type Casting):

Você pode converter entre diferentes tipos de dados usando funções como int(), float(), str(), e bool().

F. Nomeação de Variáveis:

- Os nomes de variáveis devem seguir algumas regras:
- Devem começar com uma letra (a-z, A-Z) ou um sublinhado (_).
- Podem conter letras, dígitos (0-9) e sublinhados.
- Não podem conter espaços em branco.
- Python faz diferenciação entre maiúsculas e minúsculas (sensível a maiúsculas e minúsculas), ou seja, "nome" e "Nome" são considerados nomes de variáveis diferentes.

G. Operações com Variáveis:

Você pode realizar várias operações com variáveis, dependendo do tipo de dado. Por exemplo, você pode realizar operações aritméticas com números inteiros e flutuantes e concatenação de strings.

H. Imprimir Variáveis:

Você pode exibir o valor de uma variável na tela usando a função print(). Por exemplo:

I. Comentários:

Comentários são trechos de texto em seu código que não são executados. Eles servem para explicar o código e torná-lo mais compreensível. Em Python, os comentários começam com o símbolo "#" e vão até o final da linha.

Entender variáveis e tipos de dados é fundamental para a programação, pois permite que você manipule informações e realize cálculos em seu código. Python é uma linguagem flexível e intuitiva para trabalhar com variáveis e tipos de dados, tornando-a uma excelente escolha para iniciantes em programação.

Exercício 1: Calculadora de Idade

Crie um programa que calcula a idade de uma pessoa a partir do ano de nascimento e exiba a idade na tela.

```
# Atribua o ano de nascimento a uma variável
ano_nascimento = 1990

# Calcule a idade subtraindo o ano de nascimento do ano atual
ano_atual = 2023
idade = ano_atual - ano_nascimento

# Exiba a idade na tela
print("Você tem", idade, "anos.")
```

Exercício 2: Conversão de Temperatura

Crie um programa que converte uma temperatura de Celsius para Fahrenheit.

```
# Atribua a temperatura em graus Celsius a uma variável
temperatura_celsius = 25.0

# Converta a temperatura para Fahrenheit usando a fórmula
temperatura_fahrenheit = (temperatura_celsius * 9/5) + 32

# Exiba a temperatura em Fahrenheit na tela
print("A temperatura em Fahrenheit é:", temperatura_fahrenheit)
```

Exercício 3: Concatenação de Strings

Crie um programa que solicita o nome e sobrenome de uma pessoa, concatena-os e exibe a saída formatada.

```
# Solicite o nome e o sobrenome da pessoa
nome = input("Digite seu nome: ")
sobrenome = input("Digite seu sobrenome: ")

# Concatene os nomes e exiba a saída formatada
nome_completo = nome + " " + sobrenome
print("Seu nome completo é:", nome_completo)
```

Exercício 4: Verificação de Elegibilidade

Crie um programa que verifica se uma pessoa é elegível para votar com base na idade.

```
# Atribua a idade a uma variável
idade = 18

# Verifique se a pessoa é elegível para votar
if idade >= 18:
    print("Você é elegível para votar.")
else:
    print("Você não é elegível para votar.")
```

Exercício 5: Calculadora de Média

Crie um programa que calcula a média de três notas e exibe o resultado.

```
# Atribua as notas a três variáveis
nota1 = 7.5
nota2 = 8.0
nota3 = 6.5

# Calcule a média das notas
media = (nota1 + nota2 + nota3) / 3

# Exiba a média na tela
print("A média das notas é:", media)
```

4. OPERADORES MATEMÁTICOS

Os operadores matemáticos em Python permitem realizar operações matemáticas em valores numéricos. Aqui estão os operadores matemáticos mais comuns:

A. Adição (+): Realiza a adição de dois valores.

resultado = 5 + 3 # resultado terá o valor 8

B. Subtração (-): Realiza a subtração de dois valores.

resultado = 10 - 4 # resultado terá o valor 6

C. Multiplicação (*): Realiza a multiplicação de dois valores.

resultado = 6 * 7 # resultado terá o valor 42

D. Divisão (/): Realiza a divisão de dois valores, retornando um número de ponto flutuante (float).

resultado = 15 / 3 # resultado terá o valor 5.0

E. Divisão Inteira (//): Realiza a divisão de dois valores e retorna a parte inteira do resultado.

resultado = 17 // 5 # resultado terá o valor 3

F. Resto da Divisão (%): Retorna o resto da divisão entre dois valores.

resultado = 17 % 5 # resultado terá o valor 2

G. Potenciação (): Eleva um número à potência de outro.**

resultado = 2 ** 3 # resultado terá o valor 8

H. Atribuição com Operação (+=, -=, =, /=): Realiza uma operação e atribui o resultado à variável.

x = 5

x += 3 # x agora terá o valor 8 (x = x + 3)

I. Operador de Igualdade (==): Verifica se dois valores são iguais.

igual = (5 == 5) # igual terá o valor True

J. Operador de Diferença (!=): Verifica se dois valores são diferentes.

diferente = (5 != 3) # diferente terá o valor True

Lembre-se de que os operadores matemáticos seguem a ordem padrão das operações matemáticas, onde a multiplicação e a divisão têm precedência sobre a adição e a subtração. Você também pode usar parênteses para definir a ordem de avaliação das operações, se necessário.

K. Exemplo com parênteses:

resultado = (5 + 3) * 2 # resultado terá o valor 16

Esses operadores são essenciais para realizar cálculos em Python e são amplamente usados em programação para resolver uma variedade de problemas matemáticos e lógicos.

Exercício 1: Cálculo da Área de um Retângulo

Crie um programa que calcula a área de um retângulo com base na fórmula área = comprimento x largura. Solicite ao usuário que insira o comprimento e a largura do retângulo.

```
# Solicite o comprimento e a largura do retângulo
comprimento = float(input("Digite o comprimento do retângulo: "))
largura = float(input("Digite a largura do retângulo: "))

# Calcule a área
area = comprimento * largura

# Exiba a área na tela
print("A área do retângulo é:", area)
```

Exercício 2: Conversão de Temperatura de Celsius para Fahrenheit

Crie um programa que solicita uma temperatura em graus Celsius e a converte para Fahrenheit usando a fórmula Fahrenheit = (Celsius x 9/5) + 32.

```
# Solicite a temperatura em graus Celsius
celsius = float(input("Digite a temperatura em graus Celsius: "))

# Converta para Fahrenheit
fahrenheit = (celsius * 9/5) + 32

# Exiba a temperatura em Fahrenheit
print("A temperatura em Fahrenheit é:", fahrenheit)
```

Exercício 3: Cálculo do Volume de uma Esfera

Crie um programa que calcula o volume de uma esfera com base na fórmula volume = (4/3) * pi * raio³. Use o valor de π (pi) como 3.14159265359.

```
# Importe a constante pi
from math import pi

# Solicite o raio da esfera
raio = float(input("Digite o raio da esfera: "))

# Calcule o volume da esfera
volume = (4/3) * pi * raio**3

# Exiba o volume da esfera
print("O volume da esfera é:", volume)
```

Exercício 4: Cálculo do Perímetro de um Triângulo

Crie um programa que calcula o perímetro de um triângulo com base na soma dos lados. Solicite ao usuário que insira os comprimentos dos três lados do triângulo.

```
# Solicite os comprimentos dos lados do triângulo
lado1 = float(input("Digite o comprimento do primeiro lado: "))
lado2 = float(input("Digite o comprimento do segundo lado: "))
lado3 = float(input("Digite o comprimento do terceiro lado: "))

# Calcule o perímetro do triângulo
perimetro = lado1 + lado2 + lado3

# Exiba o perímetro do triângulo
print("O perímetro do triângulo é:", perimetro)
```

Exercício 5: Cálculo de Juros Simples

Crie um programa que calcula o montante de um investimento com base na fórmula de juros simples montante = capital + (capital * taxa * tempo). Solicite ao usuário que insira o capital, a taxa de juros anual e o tempo de investimento em anos.

```
# Solicite o capital, a taxa de juros anual e o tempo de investimento
capital = float(input("Digite o capital inicial: "))
taxa_de_juros = float(input("Digite a taxa de juros anual (em decimal): "))
tempo = float(input("Digite o tempo de investimento em anos: "))

# Calcule o montante
montante = capital + (capital * taxa_de_juros * tempo)

# Exiba o montante
print("O montante após", tempo, "anos é de:", montante)
```

5. OPERADORES RELACIONAIS E LÓGICOS

Os operadores relacionais e lógicos são essenciais na programação para realizar comparações entre valores e tomar decisões com base nessas comparações. Aqui estão os operadores relacionais e lógicos mais comuns em Python:

A. Operadores Relacionais:

Igual a (==): Verifica se dois valores são iguais.

```
5 == 5  # True
```

Diferente de (!=): Verifica se dois valores são diferentes.

```
5 != 3  # True
```

Maior que (>): Verifica se um valor é maior que outro.

```
5 > 3  # True
```

Menor que (<): Verifica se um valor é menor que outro.

3 < 5 # True

Maior ou igual a (>=): Verifica se um valor é maior ou igual a outro.

5 >= 5 # True

Menor ou igual a (<=): Verifica se um valor é menor ou igual a outro.

3 <= 5 # True

B. Operadores Lógicos:

E lógico (and): Retorna True se ambas as condições forem verdadeiras.

(5 > 3) and (4 < 6) # True

OU lógico (or): Retorna True se pelo menos uma das condições for verdadeira.

(5 > 3) or (4 < 2) # True

NÃO lógico (not): Inverte o valor de uma condição, tornando True em False e vice-versa.

not (5 > 3) # False

Exercício 1: Verificação de Login

Crie um programa que verifica se o nome de usuário e a senha inseridos correspondem aos valores armazenados em variáveis.

```python
# Defina o nome de usuário e a senha corretos
usuario_correto = "usuario123"
senha_correta = "senha456"

# Solicite o nome de usuário e a senha ao usuário
usuario = input("Digite seu nome de usuário: ")
senha = input("Digite sua senha: ")

# Verifique se o nome de usuário e a senha estão corretos
login_correto = (usuario == usuario_correto) and (senha == senha_correta)

if login_correto:
    print("Login bem-sucedido!")
else:
    print("Nome de usuário ou senha incorretos. Tente novamente.")
```

Exercício 3: Verificação de Triângulo

Crie um programa que verifica se três valores podem formar um triângulo com base na desigualdade triangular.

```python
# Solicite os comprimentos dos lados do triângulo
lado1 = float(input("Digite o comprimento do primeiro lado: "))
lado2 = float(input("Digite o comprimento do segundo lado: "))
lado3 = float(input("Digite o comprimento do terceiro lado: "))

# Verifique se os lados podem formar um triângulo
triangulo_valido = (lado1 + lado2 > lado3) and (lado1 + lado3 > lado2) and (lado2 + lado3 > lado1)

if triangulo_valido:
    print("Os lados fornecidos podem formar um triângulo.")
else:
    print("Os lados fornecidos não podem formar um triângulo.")
```

6. ESTRUTURAS CONDICIONAIS

As estruturas condicionais em Python (if, elif, else) permitem que você tome decisões no seu programa com base em condições específicas. Aqui está uma explicação dessas estruturas e alguns exemplos de como usá-las:

A. Estrutura if:

A estrutura if é usada para executar um bloco de código somente se uma condição for avaliada como verdadeira (True).

Exemplo:

```
idade = 20
if idade >= 18:
    print("Você é maior de idade.")
```

B. Estrutura elif (else if):

A estrutura elif é usada para verificar condições adicionais após o if e antes do else. Ela permite lidar com múltiplas condições.

```
if condição1:
    # Bloco de código a ser executado se condição1 for verdadeira
elif condição2:
    # Bloco de código a ser executado se condição2 for verdadeira
else:
    # Bloco de código a ser executado se nenhuma das condições anteriores for verdadeira
```

Exemplo de elif:

Exemplo:

```
if nota >= 90:
    print("Nota A")
elif nota >= 80:
    print("Nota B")
elif nota >= 70:
    print("Nota C")
else:
    print("Nota D")
```

C. Estrutura else:

A estrutura else é usada para definir um bloco de código a ser executado se nenhuma das condições anteriores for verdadeira.

```
if condição:
   # Bloco de código a ser executado se a condição for verdadeira
else:
   # Bloco de código a ser executado se a condição for falsa
```

Exemplo:

```
idade = 15
if idade >= 18:
   print("Você é maior de idade.")
else:
   print("Você é menor de idade.")
```

Exemplo de Uso Combinado:

```
idade = 30
if idade < 18:
   print("Você é menor de idade.")
elif idade >= 18 and idade < 60:
   print("Você é adulto.")
else:
   print("Você é um idoso.")
```

Essas estruturas condicionais (if, elif, else) são fundamentais para controlar o fluxo de execução do seu programa, permitindo que você execute diferentes blocos de código com base nas condições especificadas. Elas são amplamente usadas para tomar decisões em programas Python.

Exercício 1: Classificação de Triângulos

Crie um programa que recebe três valores inteiros representando os lados de um triângulo e classifica o triângulo como equilátero, isósceles ou escaleno.

```
lado1 = int(input("Digite o comprimento do primeiro lado: "))
lado2 = int(input("Digite o comprimento do segundo lado: "))
lado3 = int(input("Digite o comprimento do terceiro lado: "))

if lado1 == lado2 == lado3:
    print("É um triângulo equilátero.")
elif lado1 == lado2 or lado1 == lado3 or lado2 == lado3:
    print("É um triângulo isósceles.")
else:
    print("É um triângulo escaleno.")
```

Exercício 2: Verificação de Ano Bissexto

Desenvolva um programa que verifica se um ano inserido pelo usuário é bissexto ou não. Anos bissextos têm 366 dias (um dia a mais) e ocorrem a cada 4 anos.

```
ano = int(input("Digite um ano: "))

if (ano % 4 == 0 and ano % 100 != 0) or (ano % 400 == 0):
    print("É um ano bissexto.")
else:
    print("Não é um ano bissexto.")
```

Exercício 3: Jogo de Adivinhação

Crie um jogo simples em que o programa gera um número aleatório entre 1 e 10, e o jogador deve adivinhar o número. O programa deve fornecer dicas (maior/menor) até que o jogador acerte ou desista.

```python
import random

numero_secreto = random.randint(1, 10)

print("Bem-vindo ao Jogo de Adivinhação!")
print("Tente adivinhar o número secreto entre 1 e 10.")

while True:
    palpite = int(input("Digite seu palpite: "))

    if palpite == numero_secreto:
        print(f"Parabéns! Você acertou o número {numero_secreto}.")
        break
    elif palpite < numero_secreto:
        print("Tente um número maior.")
    else:
        print("Tente um número menor.")
```

Exercício 4: Cálculo de Desconto de Compra

Escreva um programa que calcula o desconto em uma compra com base no valor total da compra. Se o valor total for maior que R$ 100, aplique um desconto de 10%. Caso contrário, não aplique desconto algum.

```python
valor_total = float(input("Digite o valor total da compra: "))

if valor_total > 100:
    desconto = valor_total * 0.1
    valor_com_desconto = valor_total - desconto
    print(f"Você obteve um desconto de R$ {desconto:.2f}.")
else:
    valor_com_desconto = valor_total

print(f"Valor a pagar: R$ {valor_com_desconto:.2f}")
```

Exercício 5: Classificação de Notas de Alunos

Crie um programa que solicita a nota de um aluno e classifica essa nota em uma das seguintes categorias: "A" (de 9 a 10), "B" (de 7 a 8.9), "C" (de 5 a 6.9), "D" (de 3 a 4.9) ou "F" (abaixo de 3). O programa deve imprimir a categoria correspondente.

```
nota = float(input("Digite a nota do aluno: "))

if nota >= 9 and nota <= 10:
    categoria = "A"
elif nota >= 7 and nota < 9:
    categoria = "B"
elif nota >= 5 and nota < 7:
    categoria = "C"
elif nota >= 3 and nota < 5:
    categoria = "D"
else:
    categoria = "F"

print(f"A nota {nota} corresponde à categoria {categoria}.")
```

7. ESTRUTURAS DE REPETIÇÃO

Loops (laços) são estruturas de controle que permitem que você execute um bloco de código repetidamente. Em Python, os dois tipos mais comuns de loops são o loop for e o loop while. Além disso, você pode usar as palavras-chave break e continue para controlar o fluxo dentro dos loops.

A. Loop for:

O loop for é usado para iterar sobre uma sequência (como uma lista, tupla, string, etc.) ou para executar um bloco de código um número específico de vezes.

Sintaxe:

for elemento in sequência:

    # Bloco de código a ser repetido

Exemplo de Loop for:

frutas = ["maçã", "banana", "laranja"]

for fruta in frutas:

```
    print(fruta)
```

B. Loop while:

O loop while é usado para repetir um bloco de código enquanto uma condição for verdadeira.

Sintaxe:

```
while condição:
    # Bloco de código a ser repetido
```

Exemplo de Loop while:

```
contador = 0
while contador < 5:
    print(contador)
    contador += 1
```

C. break:

A palavra-chave break é usada para sair imediatamente de um loop, interrompendo a execução, mesmo se a condição do loop ainda for verdadeira.

Útil quando você deseja sair de um loop com base em alguma condição.

Exemplo de break em um Loop while:

```
contador = 0
while True:
    if contador == 3:
        break
    print(contador)
```

```
    contador += 1
```

D. continue:

A palavra-chave continue é usada para pular a iteração atual de um loop e continuar para a próxima iteração.

Útil quando você deseja pular parte do loop com base em alguma condição, mas não sair completamente do loop.

Exemplo de continue em um Loop for:

```
numeros = [1, 2, 3, 4, 5]

for numero in numeros:
    if numero % 2 == 0:
        continue
    print(numero)
```

Estas são as principais características dos loops for e while, bem como das palavras-chave break e continue, que são úteis para controlar a execução dentro dos loops em Python.

Exercício 1: Contagem Regressiva

Crie um programa que conte de 10 até 1 usando um loop while e exiba os números.

```python
contador = 10

while contador >= 1:
    print(contador)
    contador -= 1
```

Exercício 2: Soma de Números Pares

Escreva um programa que calcule a soma dos números pares de 1 a 20 usando um loop for.

```python
soma = 0

for numero in range(2, 21, 2):
    soma += numero

print("A soma dos números pares de 1 a 20 é:", soma)
```

Exercício 3: Tabuada de Multiplicação

Crie um programa que exiba a tabuada de multiplicação de um número inserido pelo usuário usando um loop while.

```python
numero = int(input("Digite um número para ver sua tabuada de multiplicação: "))
contador = 1

while contador <= 10:
    resultado = numero * contador
    print(f"{numero} x {contador} = {resultado}")
    contador += 1
```

Exercício 4: Soma de Números Ímpares

Escreva um programa que calcule a soma dos números ímpares de 1 a 30 usando um loop for.

```python
soma = 0

for numero in range(1, 31, 2):
    soma += numero

print("A soma dos números ímpares de 1 a 30 é:", soma)
```

Exercício 5: Contagem de Números Pares

Escreva um programa que conte e exiba a quantidade de números pares de 1 a 10.

```python
contador_pares = 0

for numero in range(1, 11):
    if numero % 2 == 0:
        contador_pares += 1

print("Quantidade de números pares de 1 a 10:", contador_pares)
```