


# **Trabalho Prático**

## **DGT2817 - Lógica, Algoritmos e Programação de Computadores**

 Explore a documentação deste material acadêmico com exemplos práticos e explicações detalhadas!

◆ Código-fonte, documentação disponíveis no repositório oficial.

 Acesse agora no GitHub:

[Repositório do Projeto](#)

## 1. Estruturas de Condição if e else

A estrutura if-else permite tomar decisões com base em condições específicas.

Código:

```
temperatura = 29
if temperatura < 30:
    print('A temperatura hoje está amena')
else:
    print('Hoje está fazendo calor')
```

Resultado:

- Saída: A temperatura hoje está amena

```
PS C:\Users\wesle\OneDrive\Documentos\py> & C:/Users/wesle/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/wesle/OneDrive/Documentos/py/main.py
A temperatura hoje está amena
Hoje está fazendo calor
PS C:\Users\wesle\OneDrive\Documentos\py> █
```

## 2. Estruturas de Condição elif

A estrutura elif é usada para testar múltiplas condições de forma sequencial.

Código:

```
tempoExperiencia = 5
if tempoExperiencia < 2:
    print('Nível júnior')
elif 2 <= tempoExperiencia < 5:
    print('Nível pleno')
else:
    print('Nível sênior')
```

Resultado:

- Saída: Nível sênior

```
PS C:\Users\wesle\OneDrive\Documentos\py> & C:/Users/wesle/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/wesle/OneDrive/Documentos/py/main.py
Nível de conhecimento sênior.
Nível de conhecimento júnior.
Nível de conhecimento pleno.
PS C:\Users\wesle\OneDrive\Documentos\py> █
```

### 3. Estruturas de Repetição while

O laço while executa um bloco de código enquanto uma condição for verdadeira.

Código:

```
entrada_idade = ''
while entrada_idade != '0':
    entrada_idade = input('Digite um número ou 0 para sair: ')
    print(f'Número digitado: {entrada_idade}')
```

Resultado:

- Exemplo de interação:
  - Entrada: 10 → Saída: Número digitado: 10
  - Entrada: 0 → Saída: Número digitado: 0 (loop termina)

```
PS C:\Users\wesle\OneDrive\Documentos\py> & C:/Users/wesle/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/wesle/OneDrive/Documentos/py/main.py
Digite um número qualquer ou 0 para sair: 12
Número digitado: 12
Digite um número qualquer ou 0 para sair: 12
Número digitado: 12
Digite um número qualquer ou 0 para sair: 12
Número digitado: 12
Digite um número qualquer ou 0 para sair: 12
Número digitado: 12
Digite um número qualquer ou 0 para sair: 3
Número digitado: 3
```

### 4. Estruturas de Repetição for

O laço for é ideal para iterar sobre sequências, como strings ou listas.

Código:

```
texto = 'Olá, laço for.'
for item in texto:
    print(f'Caractere: {item}')
```

Resultado:

- Saída: Caractere: O Caractere: l Caractere: á ... (e assim por diante)

### 5. Funções em Python

Funções são blocos de código reutilizáveis que organizam a lógica do programa.

Código:

```
def imprimirvariavel():
    texto = 'Olá, funções em Python'
    print(texto)
imprimirvariavel()
```

Resultado:

- Saída: Olá, funções em Python

```
PS C:\Users\wesle\OneDrive\Documentos\py> & C:/Users/wesle/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/wesle/OneDrive/Documentos/py/main.py
Olá, funções em Python
PS C:\Users\wesle\OneDrive\Documentos\py> █
```

## 6. Argumentos de Funções

Funções podem receber parâmetros para personalizar seu comportamento.

Código:

```
def loginUsuario(perfil):
    if perfil.lower() == 'admin':
        print('Bem-vindo, Administrador')
    else:
        print('Bem-vindo, Usuário')
loginUsuario('Admin')
```

Resultado:

- Saída: Bem-vindo, Administrador

```
PS C:\Users\wesle\OneDrive\Documentos\py> & C:/Users/wesle/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/wesle/OneDrive/Documentos/py/main.py
Bem-vindo, Administrador
Bem-vindo, Administrador
Bem-vindo, Usuário
Bem-vindo, Usuário
PS C:\Users\wesle\OneDrive\Documentos\py> █
```

## 7. Calculadora com Estruturas Condicionais e Funções

Uma calculadora simples que utiliza funções e condições.

Código:

```
def adicao(a, b):
    return a + b
def subtracao(a, b):
    return a - b
def multiplicacao(a, b):
    return a * b
def divisao(a, b):
    if b == 0:
        return 'Erro: Divisão por zero não é permitida'
    return a / b

print(adicao(5, 3))          # Teste de adição
print(divisao(10, 0))       # Teste de divisão por zero
```

### Resultado:

- Saída:
- 8 Erro: Divisão por zero não é permitida

```
PS C:\Users\wesle\OneDrive\Documentos\py> & C:/Users/wesle/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/wesle/ve/Documentos/py/main.py
Digite o primeiro número: 12
Digite o segundo número: 12
Digite a operação (+, -, *, / ou o nome): -
Resultado da operação: 0.0
Deseja continuar? (S/N): █
```

### Conclusão

O estudo das estruturas de controle e funções é essencial para o desenvolvimento eficiente em Python, permitindo a criação de programas robustos e bem organizados.