


# Relatório Trabalho Prático Python

Kauan Henrique de Souza (202501164846)

### Microatividade 1: Descrever a utilização das estruturas de condição if e else em Python

As estruturas de condição if e else em Python são usadas para tomada de decisões no código, ou seja, permite que certo código só seja executado caso a condição seja correspondida. Dessa forma tendo grande importância no desenvolvimento do código.



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays the file structure with 'estruturas\_condicao1.py' selected. The Editor pane shows the code for this file:

```
1 temperatura = 31
2
3 if temperatura < 30:
4     print('A temperatura hoje está amena')
5 else:
6     print('Hoje está fazendo calor')
7
```

The TERMINAL pane at the bottom shows the command prompt output:

```
PS C:\Users\K4u@M_PC\Desktop\trabalhoPraticoPython> & C:\Users\K4u@M_PC\AppData\Local\Programs\Python\Python313\python.exe c:\Users\K4u@M_PC\Desktop\trabalhoPraticoPython\estruturas_s_condicao1.py
Hoje está fazendo calor
PS C:\Users\K4u@M_PC\Desktop\trabalhoPraticoPython>
```

The status bar at the bottom indicates the current position is Line 7, Column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, and the Python interpreter is set to 3.13.3.

### Microatividade 2: Descrever a utilização da estrutura de condição else if (elif) em Python

Já o uso de “elif” em uma estrutura de condição ocorre quando queremos testar várias condições diferentes, uma após a outra, dentro de uma decisão.

The screenshot shows a Python IDE with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The code in the editor defines a function to calculate experience level based on years of experience.

```

1 tempoExperiencia = 3
2
3 if tempoExperiencia < 2:
4     print("Nível de conhecimento júnior.")
5 elif tempoExperiencia > 2 and tempoExperiencia < 5:
6     print("Nível de conhecimento pleno.")
7 else:
8     print("Nível de conhecimento sênior.")
9

```

The terminal shows the command to run the script and the output:

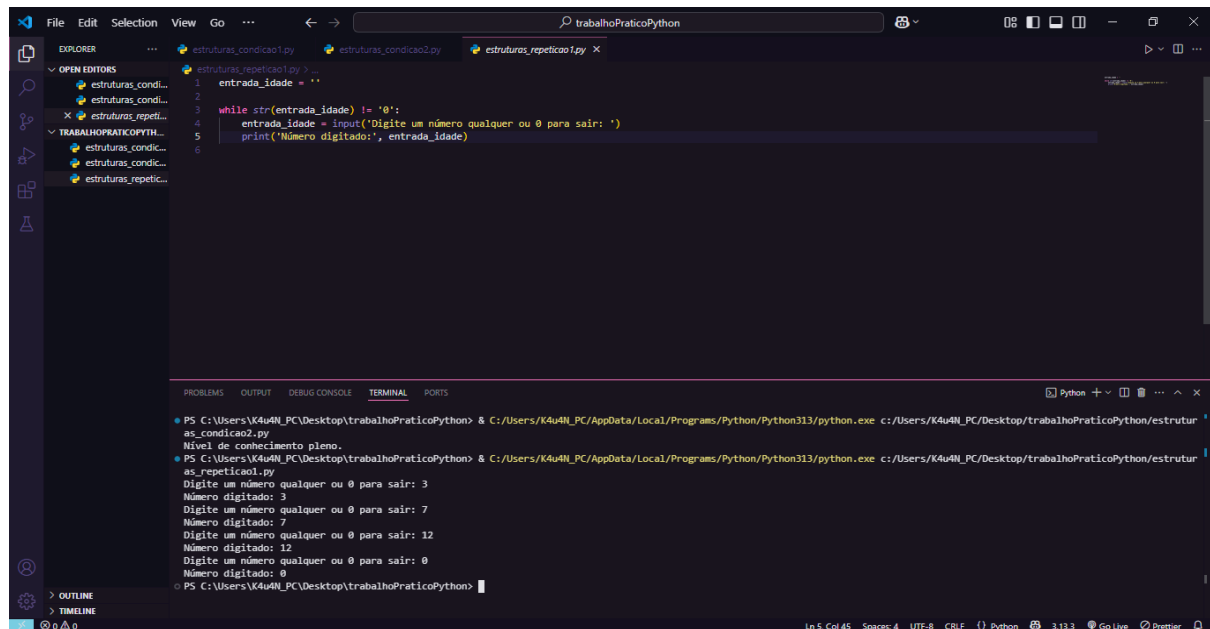
```

PS C:\Users\K4u4M_PC\Desktop\trabalhoPraticoPython> & C:\Users\K4u4M_PC\AppData\Local\Programs\Python\Python313\python.exe c:\Users\K4u4M_PC\Desktop\trabalhoPraticoPython\estrutur
as_condicao2.py
Nível de conhecimento pleno.
PS C:\Users\K4u4M_PC\Desktop\trabalhoPraticoPython>

```

### Microatividade 3: Descrever a utilização da estrutura de repetição while em Python

A estrutura de repetição while é usada para executar um bloco de código várias vezes, enquanto uma condição for verdadeira, ele é vantajoso quando não sabemos quantas vezes um determinado bloco de instruções precisa ser repetido.



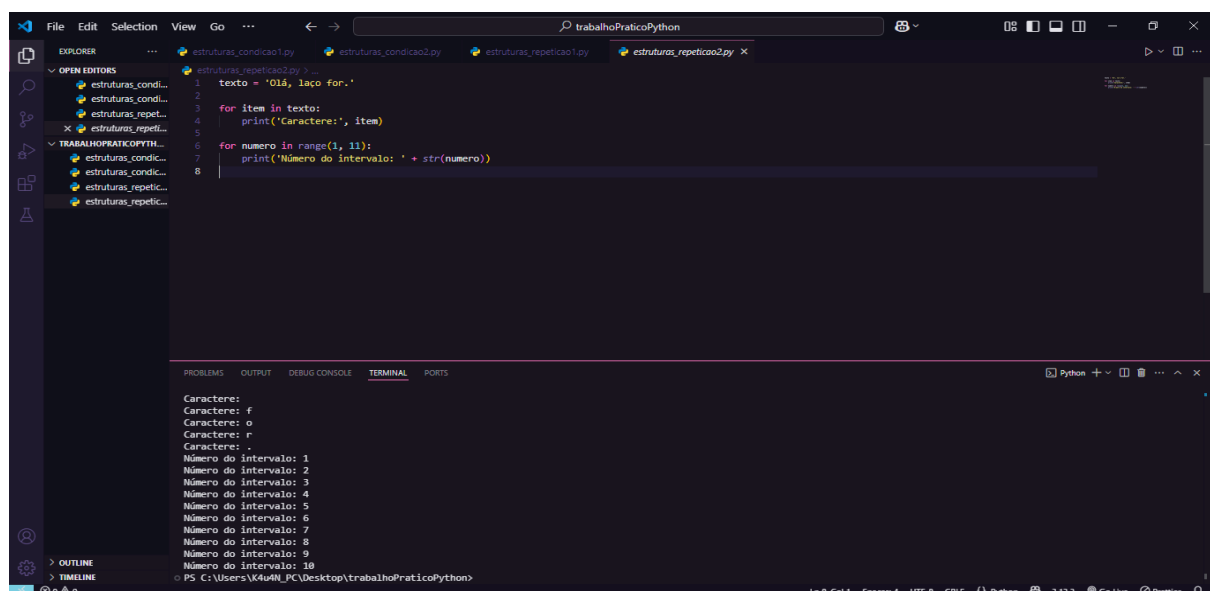
```
1 entrada_idade = ''
2
3 while str(entrada_idade) != '0':
4     entrada_idade = input('Digite um número qualquer ou 0 para sair: ')
5     print('Número digitado:', entrada_idade)
6
```

Terminal output:

```
PS C:\Users\K4u4N_PC\Desktop\trabalhoPraticoPython> & C:/Users/K4u4N_PC/AppData/Local/Programs/Python/Python313/python.exe c:/Users/K4u4N_PC/Desktop/trabalhoPraticoPython/estrutur
as_condicao2.py
Nível de conhecimento pleno.
PS C:\Users\K4u4N_PC\Desktop\trabalhoPraticoPython> & C:/Users/K4u4N_PC/AppData/Local/Programs/Python/Python313/python.exe c:/Users/K4u4N_PC/Desktop/trabalhoPraticoPython/estrutur
as_repeticao1.py
Digite um número qualquer ou 0 para sair: 3
Número digitado: 3
Digite um número qualquer ou 0 para sair: 7
Número digitado: 7
Digite um número qualquer ou 0 para sair: 12
Número digitado: 12
Digite um número qualquer ou 0 para sair: 0
Número digitado: 0
PS C:\Users\K4u4N_PC\Desktop\trabalhoPraticoPython>
```

### Microatividade 4: Descrever a utilização da estrutura de repetição for em Python

A estrutura de repetição for é usada para percorrer uma sequência de strings ou intervalos de números como no código dessa microatividade por exemplo, e executar um bloco de código para cada item dessa sequência. Ele é muito usado para repetir um código um número determinado de vezes.



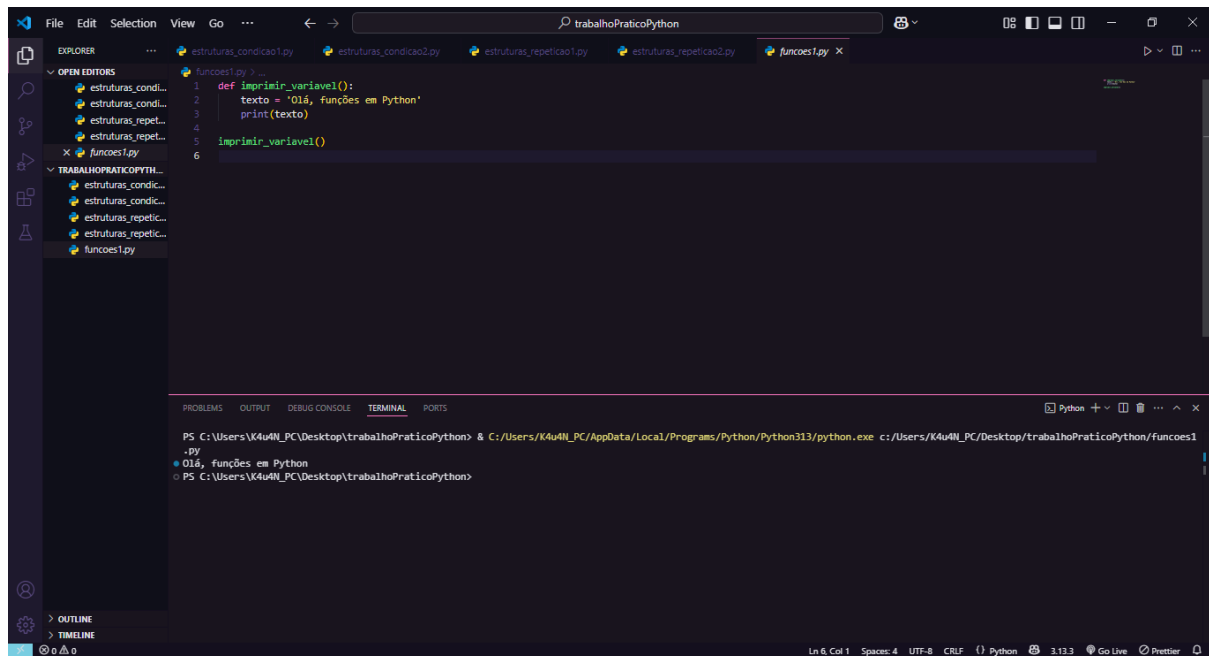
```
1 texto = 'Olá, laço for.'
2
3 for item in texto:
4     print('Caractere:', item)
5
6 for numero in range(1, 11):
7     print('Número do intervalo: ' + str(numero))
8
```

Terminal output:

```
Caractere:
Caractere: f
Caractere: a
Caractere: l
Caractere: á
Caractere: ,
Caractere: .
Número do intervalo: 1
Número do intervalo: 2
Número do intervalo: 3
Número do intervalo: 4
Número do intervalo: 5
Número do intervalo: 6
Número do intervalo: 7
Número do intervalo: 8
Número do intervalo: 9
Número do intervalo: 10
PS C:\Users\K4u4N_PC\Desktop\trabalhoPraticoPython>
```

## Microatividade 5: Descrever a utilização de funções em Python

Para criar uma função preciso usar “def” e em seguida definir o nome da função, dentro de parênteses posso adicionar um parâmetro que não é obrigatório e por isso não tem no código da microatividade. Dentro do escopo da função posso atribuir valores e fora fazer a chamada.



The screenshot shows the Visual Studio Code editor with a file named `funcoes1.py` open. The code defines a function `imprimir_variavel()` that prints the text "Olá, funções em Python". The terminal at the bottom shows the command `python c:/Users/K4u4M/Desktop/trabalhoPraticoPython/funcoes1.py` being executed, resulting in the output "Olá, funções em Python".

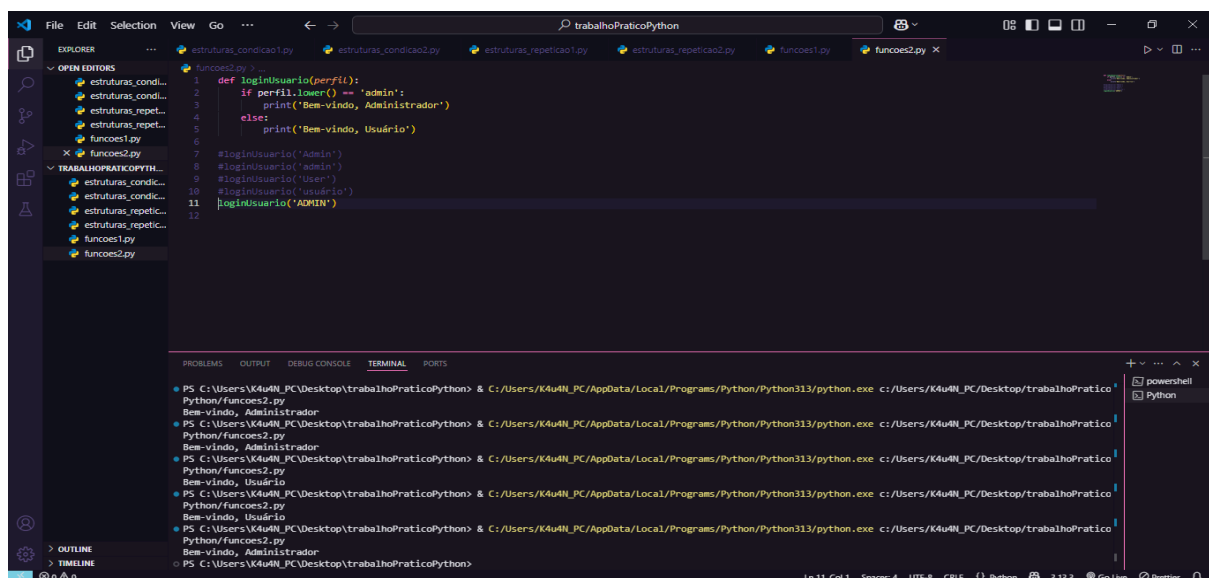
```
def imprimir_variavel():
    texto = 'Olá, funções em Python'
    print(texto)

imprimir_variavel()
```

```
PS C:\Users\K4u4M\Desktop\trabalhoPraticoPython> & C:/Users/K4u4M/AppData/Local/Programs/Python/Python313/python.exe c:/Users/K4u4M/Desktop/trabalhoPraticoPython/funcoes1.py
Olá, funções em Python
PS C:\Users\K4u4M\Desktop\trabalhoPraticoPython>
```

## Microatividade 6: Descrever a utilização argumentos de funções no Python

Os argumentos são os valores passados para uma função no momento da chamada. Eles são usados para fornecer informações que a função precisa para executar sua tarefa. Usar argumentos de forma eficiente torna as funções mais flexíveis, reutilizáveis e fáceis de entender.



The screenshot shows the Visual Studio Code editor with a file named `funcoes2.py` open. The code defines a function `login_usuario(perfil)` that checks if the user is an administrator or a regular user and prints a corresponding message. The terminal at the bottom shows the command `python c:/Users/K4u4M/Desktop/trabalhoPraticoPython/funcoes2.py` being executed, resulting in the output "Bem-vindo, Administrador".

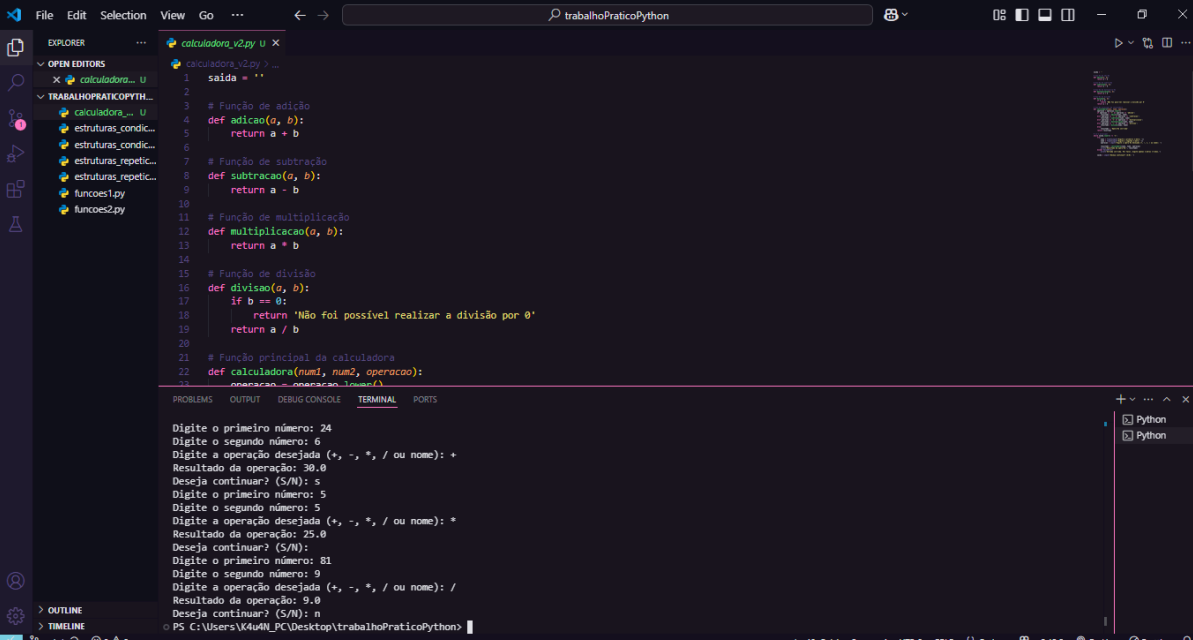
```
def login_usuario(perfil):
    if perfil.lower() == 'admin':
        print('Bem-vindo, Administrador')
    else:
        print('Bem-vindo, Usuário')

#login_usuario('Admin')
#login_usuario('admin')
#login_usuario('User')
#login_usuario('usuario')
#login_usuario('ADMIN')
```

```
PS C:\Users\K4u4M\Desktop\trabalhoPraticoPython> & C:/Users/K4u4M/AppData/Local/Programs/Python/Python313/python.exe c:/Users/K4u4M/Desktop/trabalhoPraticoPython/funcoes2.py
Bem-vindo, Administrador
PS C:\Users\K4u4M\Desktop\trabalhoPraticoPython> & C:/Users/K4u4M/AppData/Local/Programs/Python/Python313/python.exe c:/Users/K4u4M/Desktop/trabalhoPraticoPython/funcoes2.py
Bem-vindo, Administrador
PS C:\Users\K4u4M\Desktop\trabalhoPraticoPython> & C:/Users/K4u4M/AppData/Local/Programs/Python/Python313/python.exe c:/Users/K4u4M/Desktop/trabalhoPraticoPython/funcoes2.py
Bem-vindo, Usuário
PS C:\Users\K4u4M\Desktop\trabalhoPraticoPython> & C:/Users/K4u4M/AppData/Local/Programs/Python/Python313/python.exe c:/Users/K4u4M/Desktop/trabalhoPraticoPython/funcoes2.py
Bem-vindo, Administrador
PS C:\Users\K4u4M\Desktop\trabalhoPraticoPython>
```

# Trabalho Prático

O código do trabalho prático tem como objetivo realizar a função de uma calculadora fazendo operações básicas de adição, subtração, multiplicação e divisão. Para desenvolver a calculadora foi utilizado variáveis como “num1”, “num2” e “operação”, funções como as de operação “adição(a, b)” e condicionais. Sendo assim um código ótimo para a introdução de funções, condicionais, laços e tratamento de exceções em Python.



```
1  saida = ''
2
3  # Função de adição
4  def adicao(a, b):
5      return a + b
6
7  # Função de subtração
8  def subtracao(a, b):
9      return a - b
10
11 # Função de multiplicação
12 def multiplicacao(a, b):
13     return a * b
14
15 # Função de divisão
16 def divisao(a, b):
17     if b == 0:
18         return 'Não foi possível realizar a divisão por 0'
19     return a / b
20
21 # Função principal da calculadora
22 def calculadora(num1, num2, operacao):
23     operacao = operacao.lower().strip()
24
25     Digite o primeiro número: 24
26     Digite o segundo número: 6
27     Digite a operação desejada (+, -, *, / ou nome): +
28     Resultado da operação: 30.0
29     Deseja continuar? (S/N): s
30     Digite o primeiro número: 5
31     Digite o segundo número: 5
32     Digite a operação desejada (+, -, *, / ou nome): *
33     Resultado da operação: 25.0
34     Deseja continuar? (S/N):
35     Digite o primeiro número: 81
36     Digite o segundo número: 9
37     Digite a operação desejada (+, -, *, / ou nome): /
38     Resultado da operação: 9.0
39     Deseja continuar? (S/N): n
40
41     PS C:\Users\K4u4N_PC\Desktop\trabalhoPraticoPython>
```