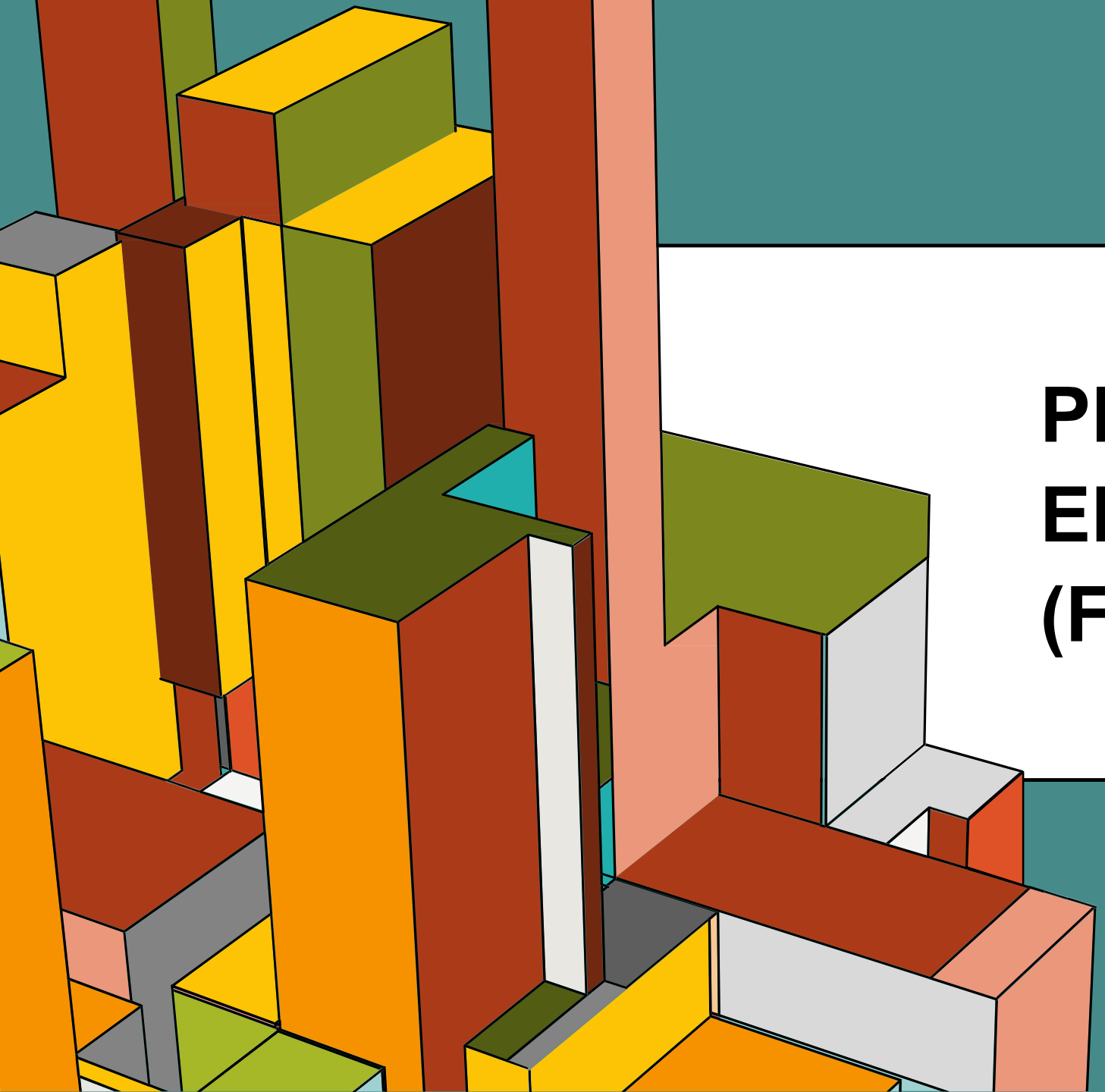


PROGRAMAÇÃO EM PYTHON (Função)

python

SENAI

FUNÇÕES EM PYTHON

Em programação, uma função é um bloco de código reutilizável que executa uma tarefa específica. As funções são usadas para dividir o código em partes mais pequenas e modulares, o que facilita a leitura, manutenção e reutilização do código.

Em Python, assim como em muitas outras linguagens de programação, as funções são definidas usando a palavra-chave def.

SINTAXE DE UMA FUNÇÃO

A sintaxe básica de uma função em Python segue o seguinte padrão:

```
def nome_da_função (parametro1, parametro2)
    # Código a ser executado
    return resultado
}
```

Parâmetros: São os valores que a função aceita como entrada. Eles são colocados entre parênteses e separados por vírgulas. Parâmetros são opcionais, e uma função pode ter zero ou mais parâmetros.

return: A palavra-chave return é usada para enviar um valor de volta quando a função é chamada. O return é opcional e pode ser omitido se a função não precisar retornar nenhum valor.

Corpo da Função: É o bloco de código indentado que contém as instruções que a função executa. Este é o local onde a tarefa específica da função é realizada.

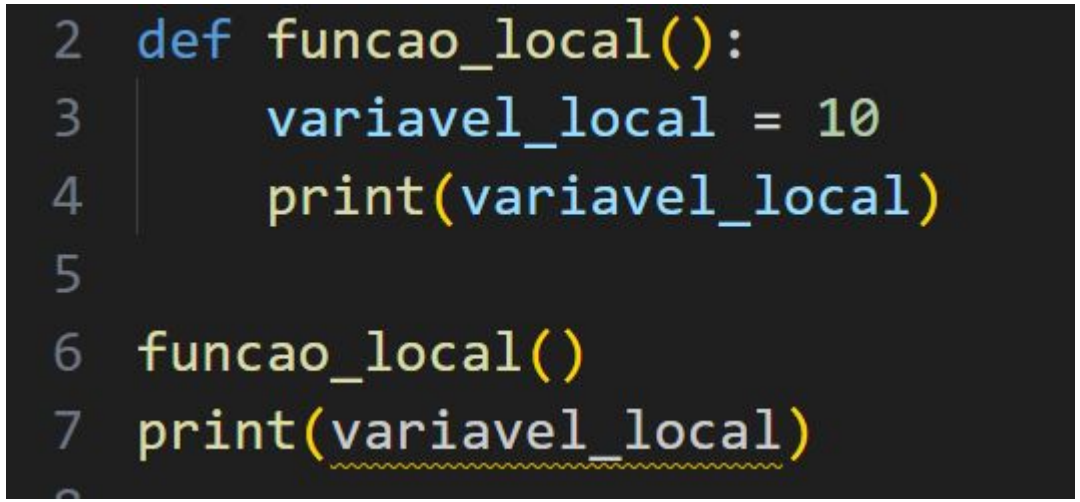
ESCOPOS DE VARIÁVEIS

O escopo de uma variável refere-se à região do código onde essa variável pode ser acessada ou modificada. Em Python, existem dois principais tipos de escopo de variáveis: escopo local e escopo global.

Escopo Local:

As variáveis definidas dentro de uma função têm escopo local.
Elas são acessíveis apenas dentro dessa função e não são visíveis fora dela.
Variáveis locais são criadas quando a função é chamada e destruídas quando a função é concluída.

Exemplo:



ESCOPOS DE VARIÁVEIS

Escopo Global:

As variáveis definidas fora de qualquer função têm escopo global.
Elas são acessíveis de qualquer lugar no código, incluindo dentro de funções.
Para modificar uma variável global dentro de uma função, você precisa usar a palavra-chave global.

Exemplo:

```
3  variavel_global = 5
4
5  def funcao_global():
6      global variavel_global
7      variavel_global += 1
8      print(variavel_global)
9
10 funcao_global()
11 print(variavel_global)
12
```

FUNÇÕES SEM PARÂMETRO

Em Python, você pode criar funções que não requerem parâmetros. Isso significa que essas funções realizam uma tarefa específica sem depender de valores passados como argumentos. Abaixo é mostrado um exemplo simples de uma função sem parâmetros:

```
2  #Criando uma Função sem parâmetro
3  def saudacao():
4      print("Olá! Bem-vindo à função sem parâmetros.")
5
6  # Chamando a função sem parâmetros
7  saudacao()
```

FUNÇÕES COM PARÂMETRO

Funções com parâmetros permitem que você passe valores específicos quando chama a função, permitindo maior flexibilidade e reutilização do código. Abaixo estão alguns exemplos de funções com parâmetros em Python:

```python
# Criando uma função com parâmetro em Python
def tabuada(numero):
    print(f"Tabuada do {numero}:")
    for i in range(1, 11):
        resultado = numero * i
        print(f"{numero} x {i} = {resultado}")

# Chamando a função com parâmetro
tabuada(7)
```

FUNÇÕES COM PARÂMETRO OPCIONAL

Funções em Python podem ter parâmetros opcionais, que são parâmetros que têm um valor padrão predefinido. Isso significa que você pode chamar a função sem fornecer um valor para esses parâmetros, e a função usará os valores padrão em vez disso. Abaixo é mostrado um exemplo de função com parâmetros opcionais:

```
4  def calcular_potencia(base, expoente=2):
5      resultado = base ** expoente
6      print(f"{base} elevado a {expoente} é igual a {resultado}")
7
8  # Chamando a função com diferentes valores para o parâmetro opcional
9  calcular_potencia(3)  # Saída: "3 elevado a 2 é igual a 9"
10 calcular_potencia(2, expoente=3)  # Saída: "2 elevado a 3 é igual a 8"
11
```

FUNÇÕES COM RETURN

Uma função em Python pode incluir a declaração return para devolver um valor específico após realizar suas operações. Isso permite que o resultado da função seja usado em outras partes do programa. Abaixo está um exemplo de função que inclue o return:

```
2  #Criando uma Função com Return
3  def somar(a, b):
4      resultado = a + b
5      return resultado
6
7  # Chamando a função com parâmetros e retorno
8  resultado_soma = somar(3, 5)
9  print("Resultado da soma:", resultado_soma)
10
```

FUNÇÕES COM EMPACOTAMENTO

O empacotamento (packing) em funções Python refere-se à capacidade de uma função aceitar um número variável de argumentos. Isso é feito usando os conceitos de argumentos posicionais, argumentos padrão e empacotamento de argumentos

```
5  #Criando uma Função com Empacotamento de Dados
6  def contador(*num ):
7      print(num)
8
9  #Chamando a Função com os parâmetros
10 contador(1,2,3)
11 contador(1,3)
12 contador(1,5,6,7)
```