Fundamentos de Programação em Python - Folha de Dicas

Pacote/Método	Descrição	Sintaxe e Exemplo de Código
AND	Retorna `True` se tanto statement1 quanto statement2 forem `True`. Caso contrário, retorna `False`.	Sintaxe: `statement1 and statement2` Exemplo: `marks = 90` `attendance_percentage = 87` `if marks >= 80 and attendance_percentage >= 85:` `print("qualify for honors")` `else:` `print("Not qualified for honors")` `# Saída = qualify for honors`
Definição de Classe	Define um modelo para criar objetos e definir seus atributos e comportamentos.	Sintaxe: `class ClassName: # Atributos e métodos da classe` Exemplo: `class Person:` `def __init__(self, name, age):` `self.name = name` `self.age = age`
Definir Função	Uma `function` é um bloco de código reutilizável que realiza uma tarefa específica ou um conjunto de tarefas quando chamado.	Sintaxe: `def function_name(parameters): # Corpo da função` Exemplo: `def greet(name): print("Hello,", name)`
Igual(==)	Verifica se dois valores são iguais.	Sintaxe: `variable1 == variable2`

		Exemplo 1: `5 == 5` retorna True Exemplo 2: `age = 25 age == 30` retorna False
Loop For	Um loop `for` executa repetidamente um bloco de código por um número especificado de iterações ou sobre uma sequência de elementos (lista, intervalo, string, etc.).	Sintaxe: `for variable in sequence: # Código a repetir` Exemplo 1: `for num in range(1, 10):` `  print(num)` Exemplo 2: `fruits = ["apple", "banana", "orange", "grape", "kiwi"]` `for fruit in fruits:` `  print(fruit)`
Chamada de Função	Uma chamada de função é o ato de executar o código dentro da função usando os argumentos fornecidos.	Sintaxe: `function_name(arguments)` Exemplo:

		`greet("Alice")`
Maior ou Igual a(>=)	Verifica se o valor de variable1 é maior ou igual a variable2.	Sintaxe: `variable1 >= variable2` Exemplo 1: `5 >= 5 and 9 >= 5` retorna True Exemplo 2: `quantity = 105` `minimum = 100` `quantity >= minimum` retorna True
Maior que(>)	Verifica se o valor de variable1 é maior que variable2.	Sintaxe: `variable1 > variable2` Exemplo 1: 9 > 6 retorna True Exemplo 2: `age = 20` `max_age = 25` `age > max_age` retorna False
Declaração If	Executa o bloco de código `se` a condição for `True`.	Sintaxe: `if condition: # bloco de código para a declaração if`

		Exemplo: `if temperature > 30:` `print("It's a hot day!")`
If-Elif-Else	Executa o primeiro bloco de código se condition1 for `True`, caso contrário, verifica condition2, e assim por diante. Se nenhuma condição for `True`, o bloco else é executado.	Sintaxe: `if condition1:` `# Código se condition1 for True` `elif condition2:` `# Código se condition2 for True` `else:` `# Código se nenhuma condição for True` Exemplo: `score = 85  # Exemplo de nota` `if score >= 90:` `print("You got an A!")` `elif score >= 80:` `print("You got a B.")` `else:` `print("You need to work harder.")` `# Saída = You got a B.`
Declaração If-Else	Executa o primeiro bloco de código se a condição for `True`, caso contrário, o segundo bloco.	Sintaxe: `if condition: # Código, se a condição for True` `else: # Código, se a condição for False` Exemplo: `if age >= 18:` `print("You're an adult.")` `else:` `print("You're not an adult yet.")`
Menor ou Igual a(<=)	Verifica se o valor de variable1 é menor ou igual a variable2.	Sintaxe: `variable1 <= variable2`

		Exemplo 1: `5 <= 5 and 3 <= 5` retorna True Exemplo 2: `size = 38` `max_size = 40` `size <= max_size` retorna True
Menor que(<)	Verifica se o valor de variable1 é menor que variable2.	Sintaxe: `variable1 < variable2` Exemplo 1: `4 < 6` retorna True Exemplo 2: `score = 60` `passing_score = 65` `score < passing_score` retorna True
Controles de Loop	`break` sai do loop prematuramente. `continue` pula o restante da iteração atual e passa para a próxima iteração.	Sintaxe: `for: # Código a repetir` `  if # declaração booleana` `  break` `for: # Código a repetir` `  if # declaração booleana` `  continue`

		Exemplo 1: `for num in range(1, 6):` `  if num == 3:` `  break` `  print(num)` Exemplo 2: `for num in range(1, 6):` `  if num == 3:` `  continue` `  print(num)`
NOT	Retorna `True` se a variável for `False`, e vice-versa.	Sintaxe: `!variable` Exemplo: `!isLocked` retorna True se a variável for False (ou seja, desbloqueada).
Não Igual(!=)	Verifica se dois valores não são iguais.	Sintaxe: `variable1 != variable2` Exemplo: `a = 10` `b = 20` `a != b` retorna True Exemplo 2:

		`` count=0 count != 0 ``` retorna False
Criação de Objeto	Criar uma instância de uma classe (objeto) usando o construtor da classe.	Sintaxe: ``` object_name = ClassName(arguments) ``` Exemplo: ``` person1 = Person("Alice", 25) ```
OR	Retorna `True` se either statement1 ou statement2 (ou ambos) forem `True`. Caso contrário, retorna `False`.	Sintaxe: ``` statement1 || statement2 ``` Exemplo: ``` "Farewell Party Invitation" Grade = 12 grade == 11 or grade == 12 ``` retorna True
range()	Gera uma sequência de números dentro de um intervalo especificado.	Sintaxe: ``` range(stop) range(start, stop) range(start, stop, step) ``` Exemplo: ``` range(5) #gera uma sequência de inteiros de 0 a 4. range(2, 10) #gera uma sequência de inteiros de 2 a 9. range(1, 11, 2) #gera inteiros ímpares de 1 a 9. ```

Declaração Return	`Return` é uma palavra-chave usada para enviar um valor de volta de uma função para seu chamador.	Sintaxe: `return value` Exemplo: `def add(a, b): return a + b` `result = add(3, 5)`
Bloco Try-Except	Tenta executar o código no bloco try. Se uma exceção do tipo especificado ocorrer, o código no bloco except é executado.	Sintaxe: `try: # Código que pode gerar uma exceção except` `ExceptionType: # Código para tratar a exceção` Exemplo: `try:` `num = int(input("Enter a number: "))` `except ValueError:` `print("Invalid input. Please enter a valid number.")`
Try-Except com Bloco Else	O código no bloco `else` é executado se nenhuma exceção ocorrer no bloco try.	Sintaxe: `try: # Código que pode gerar uma exceção except` `ExceptionType: # Código para tratar a exceção` `else: # Código a executar se nenhuma exceção ocorrer` Exemplo: `try:` `num = int(input("Enter a number: "))` `except ValueError:` `print("Invalid input. Please enter a valid number")` `else:` `print("You entered:", num)`
Try-Except com Bloco Finally	O código no bloco `finally` sempre é executado, independentemente de uma exceção ter ocorrido ou não.	Sintaxe: `try: # Código que pode gerar uma exceção except` `ExceptionType: # Código para tratar a exceção`

		finally: # Código que sempre é executado Exemplo: try: file = open("data.txt", "r") data = file.read() except FileNotFoundError: print("File not found.") finally: file.close()
Loop While	Um loop `while` executa repetidamente um bloco de código enquanto uma condição especificada permanecer `True`.	Sintaxe: while condition: # Código a repetir Exemplo: count = 0 while count < 5: print(count) count += 1

Skills Network