Introdução - Python

Python é uma linguagem de programação dinâmica e fortemente tipada. Ela utiliza tanto a tipagem de pato (duck typing) quanto a tipagem gradual por meio de dicas de tipo (type hints).

Ela é uma linguagem dinâmica pois não precisamos declarar o tipo de uma variável explicitamente, pois, o tipo de uma variável é determinado durante a execução do programa. Também dizemos ser fortemente tipada pois, embora dinâmica não significa que você possa misturar tipos de dados de maneira indiscriminada. Python impõe regras estritas sobre como os tipos de dados podem ser usados juntos, o que ajuda a evitar erros.

O duck typing é um conceito que implica que o tipo de uma variável é determinado pelo seu comportamento em vez de sua declaração explícita. Em outras palavras, se algo age como um pato (quack como um pato, anda como um pato), é considerado um pato no contexto de Python.

Variáveis e Constantes

Podemos associar variáveis a qualquer tipo de objeto usando o operador de atribuição (=), sendo: nome = valor. Uma variáveis pode ser reatribuída (ou reassociada) a diferentes valores (diferentes tipos de objetos) ao longo de sua existência.

```
>>> minha_primeira_variavel = 1
>>> minha_primeira_variavel = 2 (reatribuição)
>>> print(minha_primeira_variavel) // saída: 2
```

Constantes são nomes destinados a serem atribuídos apenas uma vez em um programa. Isso significa que seu nome e valor é imutável, não pode mudar:

```
>>> const nome = ‘Ana’
>>> nome = ‘Pedro’ --> essa reatribuição retornará erro, pois nome é uma constante
```

A palavra-chave def inicia a definição de função. Cada função pode ter zero ou mais parâmetros formais entre parênteses () seguidos por : dois pontos. As instruções para o corpo da função começam na linha seguinte após def e devem ser indentadas em um bloco.

```
def soma(x,y) :
  soma = x + y
  print(soma)
```

Chamando a função: >>> soma(3, 4)

Funções retornam explicitamente um valor ou objeto por meio da palavra-chave return. Funções que não têm uma expressão de retorno explícita retornarão implicitamente None.

```
def soma(x,y) :
soma = x + y
return soma
```