Resumo – Introdução ao Python

Modo script vs Modo interativo

Normalmente escrevemos o nosso programa em um arquivo de texto com extensão `.py`, por exemplo, `aula1.py`, e então processamos esse arquivo com o `python`. Esse é o "modo script", onde escrevemos um "roteiro" (as instruções que queremos que o `python` processe) e esse roteiro é então executado.

O outro modo é o "modo interativo", muito utilizado para rascunho e testar alguns pequenos trechos de código. Nesse modo, a cada instrução fornecida, o `python` executa essa instrução, **exibe o resultado da instrução na tela** e fica aguardando a próxima.

```
Quando você ver o símbolo abaixo, significa que está no modo interativo
Assim, o python está aguardando a próxima instrução
>>>
```

Imprimindo valores na tela

```
# Para imprimir o texto "Olá, mundo!"
print("Olá, mundo!")

# Para imprimir números ou variáveis, não utilizamos aspas
x = "Olá, mundo!"
print(x)
==> Olá, mundo!  # Funciona!

x = 10
```

```
print(x)
==> 10  # Também funciona
```

Declarando variáveis

```
# Armazenar o valor inteiro 10 em uma variável com o nome "x"
x = 10

# Armazenar o valor "Um texto" (como texto) na variável "y"
y = "Um texto"
```

Tipos de valores e variáveis

Valores e variáveis são classificados por seus **tipos**. Alguns dos principais tipos em Python são:

- `int`: categoria dos números inteiros. Ex.: `x = 10`
- `float`: categoria dos números decimais (ou não-inteiros). Ex.: `x = 12.50` (lembre que usamos `.` ao invés de `,` para denotar casas decimais).
- `string`: categoria de textos, palavras, caracteres. Ex.: `x = "Curso de Python"`.
  - É necessário utilizar aspas envolvendo o texto. Ou seja: `x = Oi tudo bem?` vai resultar em um erro.
  - Porém, quando tratamos números, não utilizamos aspas, senão estaríamos transformando esse número em uma **`string`**. Ex.: `x = "10"` — agora `x` armazena o valor "10" como **texto (string)**.
- `boolean`: valores *`True`* ou *`False`*. Denotam a veracidade ou não de uma expressão. Por exemplo: `10 == 10` vai resultar em `True`

> Podemos converter *strings* para *int* e *float* usando a função `int("10")` ou `float("10.50")`

- Podemos utilizar a função `type(valor)` para descobrir o *tipo* de um valor ou variável:

```
>>> type("teste")
<class 'str'>
>>> type(10.4)
<class 'float'>
>>> x = 10
>>> type(x)
<class 'int'>
```