PYTHON ACADEMY

TIPOS DE VARIÁVEIS NO PYTHON

Guia de tipos: int, float, str, bool, list, tuple, dict, set, None, type(), conversão de tipos (int(), str()), mutabilidade, casos práticos. Nesse ebook vamos ver os tipos de variáveis que a linguagem Python nos disponibiliza para utilizar em nossos programas.

PYTHONACADEMY.COM.BR

Salve, salve senhor ou senhora **Pythonista**!

Nesse post vamos falar sobre os diversos **Tipos de Variáveis do Python**.

Independente se você vem de outra linguagem ou se está começando agora no Python, saber quais são e como utilizá-las é muito importante para quem desenvolve código em Python.

Vamos falar de todos os tipo para que você vire um verdadeiro **Pythonista**,

Introdução

Vamos começar do começo!

Python é uma linguagem dinamicamente tipada, o que significa que não é necessário declarar o tipo de variável ou fazer *casting* (mudar o tipo de variável), pois o Interpretador se encarrega disso para nós!

Isso significa também que o tipo da variável poder variar durante a execução do programa.

Os tipos de dados padrão do Python são:

- Inteiro (int)
- Ponto Flutuante ou Decimal (float)
- Tipo Complexo (complex)
- String (str)
- Boolean (bool)
- List (list)
- Tuple

- Dictionary (dic)

Tipo Inteiro (`int`)

O tipo inteiro é um tipo composto por caracteres numéricos (algarismos) inteiros.

É um tipo usado para um número que pode ser escrito sem um componente decimal, podendo ter ou não sinal, isto é: ser positivo ou negativo.

Por exemplo, 21, 4, 0, e −2048 são números inteiros, enquanto 9.75, 1/2, 1.5 não são.

Exemplos:

```
idade = 18
ano = 2002

print(type(idade))
print(type(ano))
```

Saída:

```
<class 'int'>
<class 'int'>
```

Ponto Flutuante ou Decimal (`float`)

É um tipo composto por caracteres numéricos (algarismo) decimais.

O famoso ponto flutuante é um tipo usado para números racionais (números que podem ser representados por uma fração) informalmente conhecido como “número quebrado”.

Exemplos:

```
altura = 1.80
peso = 73.55

print(type(peso))
print(type(altura))
```

Saída:

```
<class 'float'>
<class 'float'>
```

Complexo (`complex`)

Tipo de dado usado para representar números complexos (isso mesmo, aquilo que provavelmente estudou no terceiro ano do ensino médio).

Esse tipo normalmente é usado em cálculos geométricos e científicos.

Um tipo complexo contem duas partes: a parte **real** e a parte **imaginária**, sendo que a parte imaginária contem um **j** no sufixo.

A função `complex(real[, imag])` do Python possibilita a criação de números imaginários passando como argumento: `real`, que é a parte Real do número complexo e o argumento opcional `imag`, representando a parte imaginária do número complexo.

Exemplos:

```
a = 5+2j
b = 20+6j

print(type(a))
print(type(b))
print(complex(2, 5))
```

Saída:

```
<class 'complex'>
<class 'complex'>
(2+5j)
```

String (`str`)

É um conjunto de caracteres dispostos numa determinada ordem, geralmente utilizada para representar palavras, frases ou textos.

Exemplos:

```
nome = 'Guilherme'
profissao = 'Engenheiro de Software'

print(type(profissao))
print(type(nome))
```

Saída:

```
<class 'str'>
<class 'str'>
```

Boolean (`bool`)

Tipo de dado lógico que pode assumir apenas dois valores: falso ou verdadeiro (`False` ou `True` em Python).

Na lógica computacional, podem ser considerados como 0 ou 1.

Exemplos:

```
fim_de_semana = True
feriado = False

print(type(fim_de_semana))
print(type(feriado))
```

Saída:

```
<class 'bool'>
<class 'bool'>
```

Listas (`list`)

Tipo de dado muito importante e que é **muito** utilizado no dia a dia do desenvolvedor Python!

Listas agrupam um conjunto de elementos variados, podendo conter: inteiros, floats, strings, outras listas e outros tipos.

Elas são definidas utilizando-se colchetes para delimitar a lista e vírgulas para separar os elementos, veja alguns exemplo abaixo:

```
alunos = ['Amanda', 'Ana', 'Bruno', 'João']
notas = [10, 8.5, 7.8, 8.0]

print(type(alunos))
print(type(notas))
```

Saída:

```
<class 'list'>
<class 'list'>
```

Aqui na Python Academy temos **muito** conteúdo sobre Listas para você ficar craque!

Para saber mais sobre Listas acesse nosso post sobre Listas, também nosso post sobre Manipulação de Listas, e nosso post completo sobre List Comprehensions.

Tuplas (`tuple`)

Assim como Lista, Tupla é um tipo que agrupa um conjunto de elementos.

Porém sua forma de definição é diferente: utilizamos parênteses e também separado por vírgula.

A diferença para Lista é que Tuplas são **imutáveis**, ou seja, após sua definição, Tuplas **não podem ser modificadas**.

Vamos ver alguns exemplos:

```
valores = (90, 79, 54, 32, 21)
pontos = (100, 94.05, 86.8, 62)

print(type(valores))
print(type(pontos))
```

Saída:

```
<class 'tuple'>
<class 'tuple'>
```

Caso haja uma tentativa de alterar os itens de uma tupla após sua definição, como no código a seguir:

```
tuple = (0, 1, 2, 3)
tuple[0] = 4
```

O seguinte erro do tipo `TypeError` será lançado pelo interpretador do Python:

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Dicionários (`dict`)

Dict é um tipo de dado muito flexível do Python.

Eles são utilizados para agrupar elementos através da estrutura de chave e valor, onde a chave é o primeiro elemento seguido por dois pontos e pelo valor.

Vamos ver alguns exemplos:

```
altura = {'Amanda': 1.65, 'Ana': 1.60, 'João': 1.70}
peso = {'Amanda': 60, 'Ana': 58, 'João': 68}

print(type(altura))
print(type(peso))
```

Saída:

```
<class 'dict'>
<class 'dict'>
```

Dicionários possibilitam um tipo de manipulação muito poderosa, chamada de *Dict Comprehensions*! Você pode saber tudo sobre *dict comprehensions* clicando aqui e lendo nosso post sobre o assunto!

> 💡 *Estou construindo o **DevBook**, uma plataforma que usa IA para criar ebooks técnicos — com código formatado e exportação em PDF. Te convido a conhecer!*

Como mudar o tipo de uma variável

Em determinados cenários pode ser necessário mudar o tipo de uma variável e no Python isso é muito fácil, uma das vantagens de uma linguagem dinamicamente tipada.

Abaixo veremos exemplos de como trocar o tipo de variáveis.

Decimal (`float`) para String (`str`)

```
# Antes da conversão
altura = 1.80
print(type(altura))

# Conversão do tipo
altura = str(altura)

# Depois da conversão
type(altura)
print(altura)
```

Saída:

```
<class 'float'>
<class 'str'>
'1.8'
```

Inteiro (`int`) para Decimal (`float`):

```
# Antes da conversão
idade = 18
print(type(idade))

# Conversão do tipo
idade = float(idade)

# Depois da conversão
print(type(idade))
print(idade)
```

Saída:

```
<class 'int'>
<class 'float'>
18.0
```

Booleano (`bool`) para Inteiro (`int`)

```python
# Antes da conversão
fim_de_semana = True
print(type(fim_de_semana))

# Conversão do tipo
fim_de_semana = int(fim_de_semana)

# Depois da conversão
print(type(fim_de_semana))
print(fim_de_semana)
```

Saída:

```
<class 'bool'>
<class 'int'>
1
```

Erros comuns relacionados ao tipo da variável

Entre os erros mais comuns que acontecem principalmente com quem está iniciando sua jornada com a linguagem Python estão os famosos **TypeError**!

Portanto preste sempre atenção nesse tipo de erro pois ele está relacionado a tipagem errada de uma variável ou objeto.

Além disso, pode representar uma operação incompatível, como tentar fazer uma operação matemática com variáveis de tipo incompatíveis.

Por exemplo, suponha as seguintes variáveis, do tipo inteiro e string, respectivamente:

```
peso = 65.55
altura = '1.7'
```

Se tentarmos fazer a seguinte operação:

```
imc = peso/altura**2
```

O seguinte erro será lançado:

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
```

Para fazer esse cálculo corretamente a altura não pode ser string, podendo ser `float` ou inteiro (caso não fosse um valor decimal).

Uma forma correta de realizar esse cálculo seria da seguinte maneira:

```
peso = 65.55
altura = 1.7
imc = peso/altura**2
print(imc)
```

Dessa forma, obteríamos a seguinte saída:

```
22.68166089965398
```

Sempre que o tipo `TypeError` aparecer em seu programa, preste atenção aos tipos dos dados e as operações que estão sendo executadas.

Caso seja necessário, faça a conversão das variáveis para se trabalhar com os tipos correto!

Conclusão

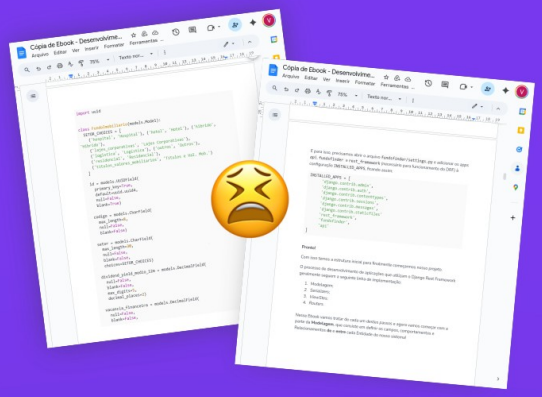
Espero que esse post tenha te ajudado a entender melhor os diversos tipos de variáveis que o Python nos disponibiliza para utilizar.

Não se esqueça de conferir!

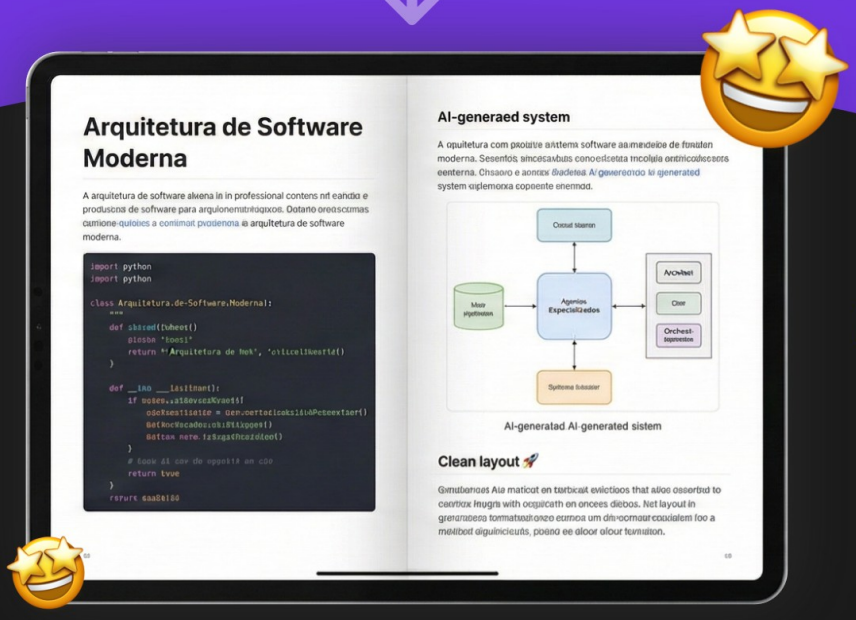
DevBook

Crie Ebooks técnicos em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código**!



Chega de formatar código no Google Docs



Syntax Highlight

Infográficos feitos por IA

Adicione Banners Promocionais

Deixe que nossa IA faça o trabalho pesado

Edite em Markdown em Tempo Real

TESTE AGORA

PRIMEIRO CAPÍTULO 100% GRÁTIS