

Fábrica de Software

Aula 03

Python



Tipos de variáveis

Em Python, você não precisa definir o tipo de variável antes de poder usá-la. Tudo o que você precisa fazer é atribuir um valor a um nome de variável para que ele seja criado automaticamente com o tipo que melhor corresponda ao valor fornecido.

Podemos dizer que a tipagem das variáveis em Python é uma do tipo dinâmica, ao contrário da tipagem estática, mais comum, como por exemplo, em C ++ ou Java. Nessas linguagens, é sempre necessário - por instruções separadas - primeiro declarar o nome e o tipo das variáveis e, então, apenas atribuir-lhes um conteúdo, que obviamente deve ser compatível com o tipo declarado.

Tipos Variáveis



TIPO INTEIRO (int)

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

Ex: 20 – 58 -11 – 36

```
#Tipo Inteiro  
type(20)
```

```
int
```

```
#Tipo inteiro
```

```
valor = 20  
type(valor)
```

```
int
```

Ponto Flutuante ou Decimal (float)

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

Ex: 32,5 – 15,8 – 65,6

```
#Tipo float  
type(25.2)
```

float

```
#Tipo float  
valor = 25.2  
type(valor)
```

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.

Ex: $2 + 5j$

```
#Tipo Complex  
type(2 + 5j)
```

complex

```
#Tipo Complex  
valor = 2 + 5j  
type(valor)
```

complex

String (str)

É um conjunto de caracteres dispostos numa determinada ordem, geralmente utilizada para representar palavras, frases ou textos. Elas devem estar entre aspas.

Ex: Todas as palavras dessa frase são strings. Essa frase inteira também é.

```
#Tipo String  
type("palavra")
```

str

```
#Tipo String  
valor = "palavra"  
type(valor)
```

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.

```
#Tipo Bool  
type(True)
```

```
bool
```

```
#Tipo Bool  
  
valor = True  
type(valor)
```

```
bool
```

Listas (list)

Uma lista é a estrutura de dados mais básica do Python e armazena os dados em sequência, onde cada elemento possui sua posição na lista, denominada de índice. O primeiro elemento é sempre o índice zero e a cada elemento inserido na lista esse valor é incrementado. No Python, uma lista pode armazenar qualquer tipo de dado primitivo (string, inteiro, float, etc).

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

Ex:

```
# Tipo Lista  
  
exemplo = [20,54,'verde']  
type(exemplo)
```

```
list
```

```
# Tipo Lista  
  
exemplo2 = ['casa',2568,True]  
type(exemplo2)
```

```
list
```

Tuplas (tuple)



Uma tupla é uma estrutura bastante similar a uma lista, com apenas uma diferença: os elementos inseridos em uma tupla não podem ser alterados, diferente de uma lista onde podem ser alterados livremente. 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.

```
# Tipo Tupla
exemplo = (22, 'parte', 65.8, False)
type(exemplo)
tuple
```

Dicionários (dict)

No Python, os dicionários são coleções de itens desordenados com uma diferença bem grande quando comparados às outras coleções (lists, sets, tuples, etc): um elemento dentro de um dicionário possui uma chave atrelada a ele, uma espécie de identificador. Sendo assim, é muito utilizado quando queremos armazenar dados de forma organizada e que possuem identificação única (como acontece em bancos de dados). 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.

```
# Tipo Dicionário
exemplo = {'Nome': 'Maria', 'Idade': 22, 'Nacionalidade': 'brasileira'}
type(exemplo)
dict
```

Funções

Função é uma sequência nomeada de instruções que executa uma operação de computação. Ao definir uma função você especifica o nome e a sequência de instruções. Depois, você pode “chamar” a função pelo nome.

Ex: type(50)

Nesse caso, chamamos a função type(). A expressão dentro do parênteses chama-se argumento. A função type nos devolve o tipo do argumento (int, float, bool, etc). Dizemos que a função recebe um argumento e retorna um resultado.

Temos também funções que convertem valores de um tipo em outro. Por exemplo, a função int recebe um valor e o transforma em inteiro.

```
#Transformando float em int  
int(33.5)
```

33

Mudando 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)

```
# Decimal para string  
str(25.8)
```

'25.8'

Na função input da aula anterior, colocamos o int para nos certificarmos que receberíamos um valor inteiro. Lembra?

```
var1 = int( input("Digite um inteiro: ") )  
var2 = int( input("Digite outro inteiro: ") )  
  
soma = var1 + var2  
  
print(soma)
```

```
Digite um inteiro: 10  
Digite outro inteiro: 20  
30
```

Entrada de dados com a Função input

Vamos começar falando sobre **entrada do usuário**. Esse é um artifício muito comum em programação, quando precisamos que o usuário passe ao programa algum tipo de dado.

Em Python, fazemos isso utilizando a função `input()`, que é literalmente ‘entrada’ em inglês. A função `input()` recebe como parâmetro uma string que será mostrada como auxílio ao usuário, geralmente o informando que tipo de dado o programa está aguardando receber. Já utilizamos a função input, no exemplo anterior. Note que no exemplo anterior, “forçamos” a entrada como sendo um número inteiro, isso é bastante útil.

Saída de dados com a Função print

A função para imprimir dados em Python é a função `print()`. Ela é responsável por mostrar valores em seu terminal. Podemos inserir valores em qualquer posição do `print`, usando as chaves e o comando `format()`.

```
n1 = int(input("Insira um número qualquer: "))
n2 = int(input("Insira um segundo número qualquer: "))
soma = n1 + n2
print("A soma dos número {}, e {} é {}".format(n1,n2,soma))
```

```
Insira um número qualquer: 35
Insira um segundo número qualquer: 42
A soma dos número 35, e 42 é 77
```

Funções Built-in

Funções built-in são funções internas e nativas, ou seja, que já vem incorporadas na linguagem e estão sempre disponíveis para utilização. Assim não é necessário a importação. Basta utilizá-las diretamente no seu código quando desejar. O Python possui diversas funções built-in.

Funções Built-in				
<code>abs()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>all()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>any()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>ascii()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bin()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>bool()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	
<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>	

Link: <https://docs.python.org/pt-br/3.6/library/functions.html>