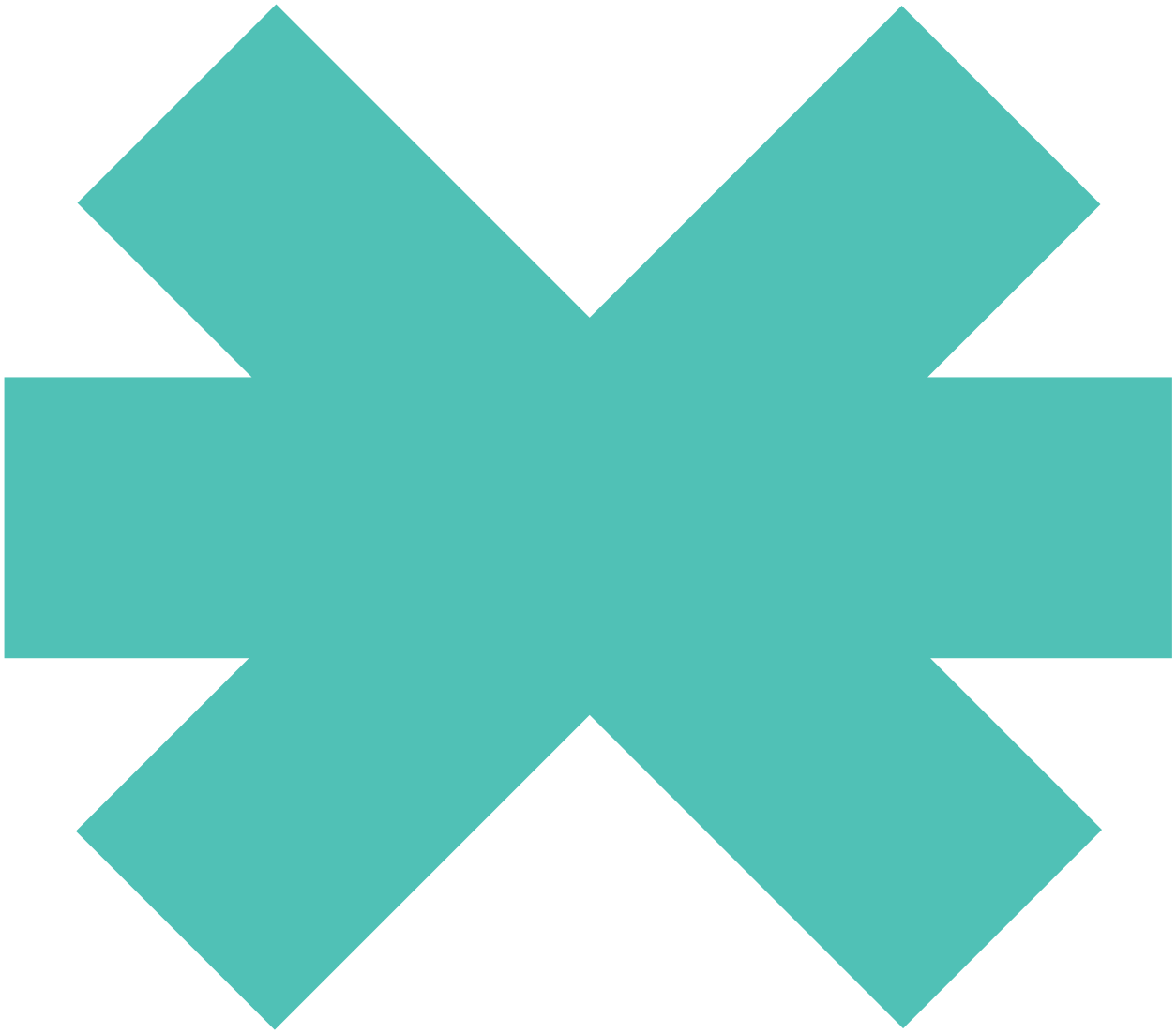
MÓDULO 2
Características básicas da linguagem
Programação Python

1

Tipos de dados básicos

New Technology School

Tokio.



1 Tipos de dados básicos

Sumário

1.1 Palavras reservadas

Em Python existem palavras com um significado especial para o interpretador, que apenas podem ser utilizadas para a finalidade para a qual foram criadas. Apresentamos a lista das 34 palavras reservadas, do interpretador do *Python* 3:

and	*as*	*assert*	*async*	*await*	*break*	*class*	*continue*	*def*	*del*	*elif*	*else*
except	*False*	*finally*	*for*	*from*	*global*	*if*	*import*	*is*	*lambda*	*None*	*nonlocal*
not	*or*	*pass*	*raise*	*return*	*True*	*try*	*while*	*with*	*yield*		

1.2 Comentários

Os **comentários** em Python são linhas dentro do código que o interpretador ignora, no momento de executar o programa. Servem tanto para facilitar a leitura do código, a pessoas que não o tenham desenvolvido, como para, futuramente, relembrar o que faz o nosso programa e como o faz.

Escrever comentários, ainda que requeira um esforço, é uma boa prática.

Existem duas formas de escrever comentários em Python:

- **Notação *Inline*:** Através do símbolo cardinal (#) conseguiremos que o interpretador do Python ignore tudo o que estiver atrás dele, até ao final da linha.
- **Notação *Multiline*:** Delimitando o comentário entre os caracteres de três aspas duplas (“””) ou simples (‘’’), conseguiremos que o interpretador do Python ignore tudo o que estiver entre esses delimitadores.

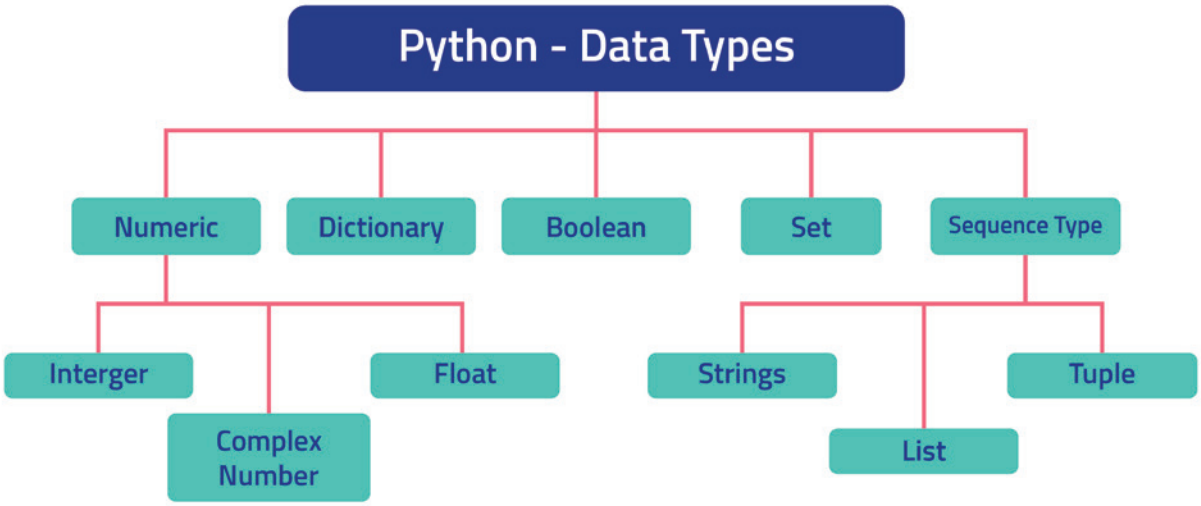
```
In [ ]: # Isto é um comentário de uma linha em Python

In [ ]: '''Isto é um comentário
multilinha'''

In [ ]: """Isto é outro comentário
multilinha"""
```

Um dado a ter em conta é que os comentários *multiline* em *Jupyter Notebook* não são aceites, pelo que apenas poderemos utilizar comentários *inline*.

```
In [1]: # Divisão
3 / 2

Out[1]: 1.5

In [2]: # Módulo
3 % 2

Out[2]: 1

In [3]: # Potência
3 ** 2
```

1.3 Tipos de dados básicos

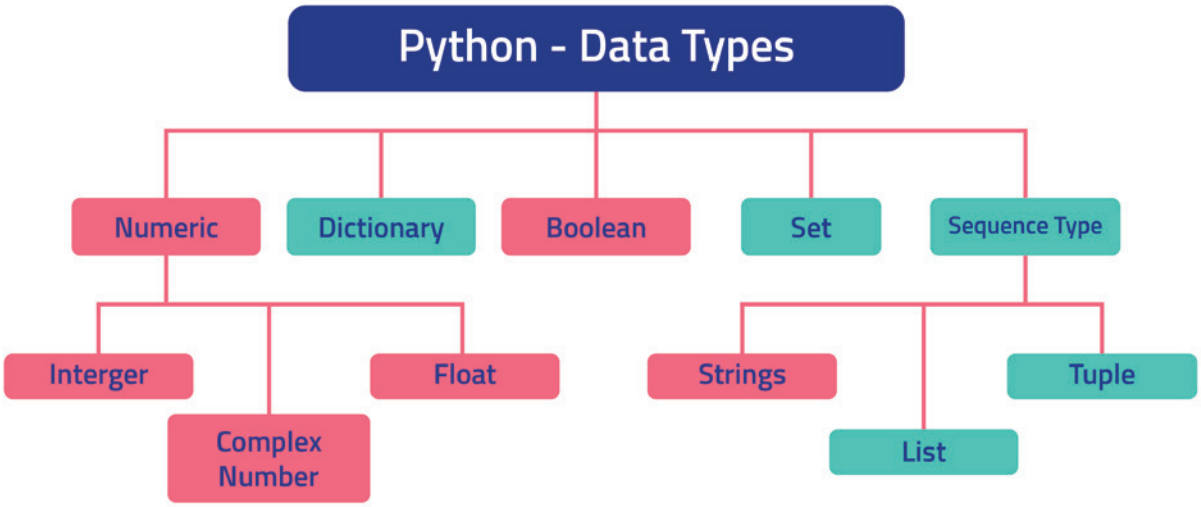
Um **tipo de dados** é um atributo dos dados que indica ao sistema e ao programador, a classe de dados que vão armazenar. Isto inclui impor restrições aos dados, como os valores que podem ter e que operações podem realizar. Por exemplo, um tipo de dados numérico, apenas pode armazenar números, e um tipo de dados *booleano*, apenas pode armazenar dois estados, verdadeiro ou falso. Segundo o tipo de dados utilizado para definir uma variável, ocupa mais ou menos espaço, na memória.



Os tipos de dados podem ser divididos em básicos, que são os nativos da linguagem; e em avançados, que são uma extensão dos nativos.

Agora veremos os tipos de dados básicos (numéricos, cadeias de caracteres e lógicos), os avançados (lista, tupla, conjunto e dicionário), veremos posteriormente.

Os **tipos de dados básicos** em Python são:



NUMÉRICOS

Este tipo de dados é utilizado para representar números. Dentro destes dados diferenciamos três tipos:

- Inteiros (*Integer*): Este tipo de dados compreende o conjunto de todos os números inteiros, aqueles que não têm decimais, mas como esse conjunto é infinito, o Python está limitado pela capacidade de memória disponível, não há um limite de representação imposto pela linguagem.

```
In [7]: 1
Out[7]: 1
```

- Decimais ou flutuantes (*Float*): Este tipo de dados compreende o conjunto de todos os números reais, aqueles que têm uma parte inteira e uma parte decimal. Tal como os números inteiros, não existe um limite de representação imposto pela linguagem.

```
In [8]: 323239829389.238273283
Out[8]: 323239829389.2383
```

- Complexos (*Complex*): Este tipo de dados compreende os números compostos de uma parte real e uma parte imaginária, ambas as partes são representadas como tipo flutuante, pelo que um número complexo constará, por sua vez, de duas partes diferenciadas de tipo *float*: uma parte real e outra imaginária.

```
In [1]: 2 + 4j
Out[1]: (2+4j)
```

CADEIAS DE CARACTERES (*STRINGS*)

As cadeias de caracteres são sequências imutáveis que contêm caracteres entre aspas duplas (“”) ou simples (‘’).

```
In [4]: 'Olá Mundo'
Out[4]: 'Ola Mundo'

In [5]: "Olá Mundo"
Out[5]: 'Olá Mundo'
```

Em Python existe uma função *print*() que nos permite mostrar corretamente o valor de uma cadeia ou outros valores ou variáveis por ecrã. Utiliza-se da forma seguinte:

```
In [6]: print("Uma cadeia")
        print("Outra cadeia")
        print("Outra cadeia mais, cada um numa linha")

        Uma cadeia
        Outra cadeia
        Outra cadeia mais, cada um numa linha
```

LÓGICOS OU BOOLEANOS (*BOOLEAN*)

Representa a mínima expressão racional, os valores verdadeiro (*True* ou 1) ou falso (*False* ou 0). Os tipos de dados *booleanos* utilizam-se habitualmente para guardar, a qualquer momento, o estado de uma propriedade ou característica:

```
In [7]: EstaaChover = False
        FazSol = True
        print(EstaaChover)
        print(FazSol)

        False
        True
```

1.4 Variáveis

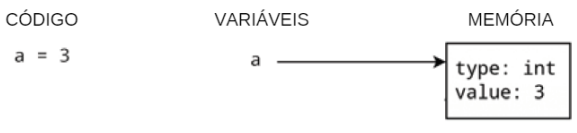
Este é um conceito fundamental da programação, no qual se define um identificador e se atribui um valor e, posteriormente, poderemos utilizar as variáveis como se de um valor literal se tratasse, podemos inclusivamente operá-las entre outras variáveis e voltar a atribuir-lhes um valor a qualquer momento.

Na realidade, uma **variável** é um lugar, na memória, onde armazenar um dado, cujo valor poderemos variar durante a execução do programa. Tem um nome, um valor e é de um tipo. Como vimos anteriormente, segundo o tipo de dados, que seja utilizado para definir uma variável, ocupa mais ou menos espaço na memória.

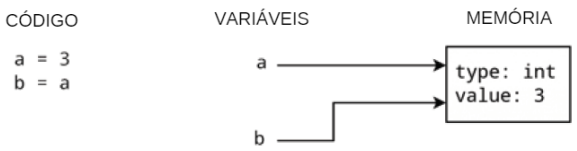
Tipo	Classe	Notas	Exemplo
str	Corrente	Imutável	"Hoje"
unicode	Corrente	Versão Unicode de str	u"Hoje"
list	Sequência	Mutável, contém objetos de diversos tipos	[4, "Hoje", 3.14]
tuple	Sequência	Imutável, contém objetos de diversos tipos	(4, "Hoje", 3.14)
set	Conjunto	Mutável, sem ordem e sem duplicatas	Set([4, "Hola", 3.14])
frozenset	Conjunto	Imutável, sem ordem e sem duplicatas	Frozenset([4, "Hola", 3.14])
dict	Dicionário	Pares de chaves: valor	("chave1":4, "chave2": "Hoje")
int	Inteiro	Precisão fixa, converte para longo se necessário	32
Long	Inteiro	Precisão arbitrária	32L ó 1298918298398923L
float	Decimal	Ponto flutuante de dupla precisão	3.141592
complex	Complexo	Parte real e imaginária	(4.5 + 3j)
bool	Booleano	Valores verdadeiro ou falso	True ou False

O conteúdo de uma variável é o seu valor armazenado na memória. Qualquer operação realizada sobre uma variável é realizada sobre o seu valor. O que contém não é mais do que a associação de um nome chamado identificador, e um endereço de memória, que aponta para o conteúdo, ou seja, o valor associado a esse nome.

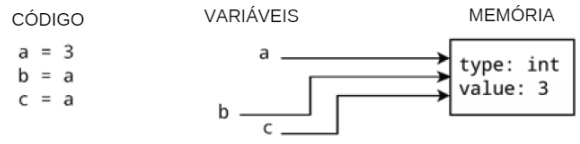
1



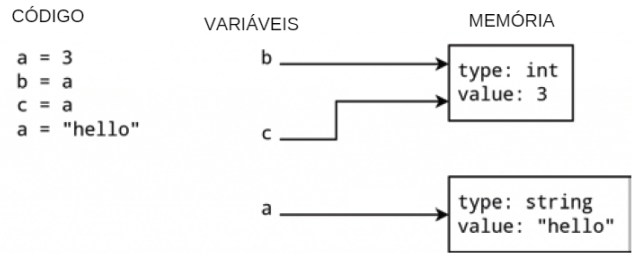
2



3



4



As regras que constituem o identificador de uma variável são:

- As palavras reservadas não podem ser utilizadas como identificadores.
- Não são permitidos espaços em branco.
- Não podem ser incluídos caracteres especiais como: !, @, #, $, %.
- Não é permitido utilizar um dígito como primeiro carácter.

Uma boa prática na utilização de variáveis implica:

- Utilizar apenas letras minúsculas, maiúsculas, dígitos ou o *underscore* (_).
- Começar sempre com uma letra minúscula.
- Utilizar nomes descritivos, ainda que sejam longos e compostos por várias palavras.
- Existem métodos distintos para separar o conjunto de palavras que forma uma variável e torná-la mais legível. Um, o mais habitual, consiste em capitalizar cada palavra agrupada (*camelCase*) e outro, consiste em separar cada palavra com um *underscore* (*snake_case*). A distância recomendável é de 2 a 4 palavras ou entre 8 e 20 caracteres.

A atribuição de um valor a uma variável é realizada com o operador de atribuição =, que veremos posteriormente. É importante mencionar que em Python a atribuição não imprime o resultado no ecrã, contrariamente ao que acontece em *MATLAB* e *Octave*, exceto se incluirmos o ponto e vírgula no final. A melhor forma de visualizar a variável que acabamos de atribuir é esta (atenção, porque apenas é valida em *Jupyter Notebook*):

```
In [8]: # Atribuição de um valor a uma variável
        n = 3
        n
Out[8]: 3
```

1.5 Constantes

Uma **constante** é um lugar, na memória, onde armazenar um dado, cujo valor não pode ser alterado durante a execução do programa. Tem um nome, um valor e é de um tipo. Em Python não existem constantes, e para resolver este problema, a solução mais comum e prática é através da utilização de variáveis.

As regras que constituem o identificador, ou nome de uma constante, são as mesmas que para as variáveis:

As boas práticas na utilização de variáveis implicam:

- Utilizar apenas letras maiúsculas, dígitos ou o *underscore* (_).
- Utilizar nomes descritivos, ainda que sejam longos e compostos por várias palavras.
- Neste caso, o conjunto de palavras que constitui uma constante não se separa (*TUDOUNIDO*) ou separa-se cada palavra com um *underscore* (*TUDO_UNIDO*). Como nas variáveis, a distância recomendável é de 2 a 4 palavras ou entre 8 e 20 caracteres.

```
In [ ]: CONSTANTE = 125
```