Ficha 1 - Variáveis e tipos em Python resolvida

Jorge Gustavo Rocha

May 30, 2020

1 Variáveis e tipos em Python

Jorge Gustavo Rocha
Departamento de Informática, Universidade do Minho
27 de abril de 2020

O Python é uma linguagem tipada, isto é, as operações só fazem sentido para um determinado tipo de operando. Se o tipo do operando não é o adequado, o Python interrompe a execução.

Nestes exercícios vamos introduzir as variáveis, os tipos e os objectos. São alguns dos conceitos fundamentais para se programar em Python.

1.1 Variáveis

As variáveis são usadas para guardar dados, para posterior utilização. No exemplo seguinte definem-se duas variáveis `g` e `nome`que posteriormente são usadas na função `print()`.

[1]:
```
g = 9.8
nome = 'Galileu Galilei'
print( "A aceleração da gravidade é {} m/s^2 e já era do conhecimento do {}".
 ↪format( g, nome ))
```

```
A aceleração da gravidade é 9.8 m/s^2 e já era do conhecimento do Galileu
Galilei
```

1.1.1 Nomes das variáveis

Como em qualquer linguagem, o Python tem regras muito bem definidas para os identificadores.

1. As variáveis têm que começar com uma letra ou `_`. Não podem começar com algarismos.
2. As variáveis só podem conter letras, algarismos e `_`. Boa notícia para os portugueses: as letras acentuadas são aceites.
3. As letras maiúsculas são diferentes das minúsculas. Por isso, as variáveis `nome`, `Nome` e `NOME` são todas diferentes.

Exemplos de identificadores para variáveis:

Válido	Inválido
`data`	`2vezes`
`data2`	`%perc`
`ângulo`	`o ângulo`

Válido	Inválido
`vel_inicial`	`vel-inicial`

Exercício Complete o código seguinte, de forma a provar que as duas variáveis são diferentes (por diferirem em maiúsculas e minúsculas).

[2]:
```
nome = 'Manuel'
Nome = 'Maria'
print(nome)
# imprime Manuel, porque a instrução:
# Nome = 'Maria'
# não altera em nada o conteúdo da variável nome
```

```
Manuel
```

1.1.2 Atribuição

A atribuição de uma valor a uma variável, como já se viu, na sua forma mais simples é:

```
g = 9.8
```

Como usamos o símbolo `=` para atribuir valores a variáveis. Para comparar dois valores, temos que usar um símbolo diferente, que é o `==`. Ou seja, as seguintes expressões são completamente diferentes:

```
g = 9.8
g == 9.8
```

A expressão `g == 9.8` dá sempre um resultado booleano (`True` ou `False`).

Em Python, pode-se fazer várias atribuições numa só linha. Por exemplo:

[3]:
```
r, g, b = "Vermelho", "Verde", "Azul"
print("Pode-se formar qualquer cor juntando: {}, {} e {}".format( r, g, b) )
```

```
Pode-se formar qualquer cor juntando: Vermelho, Verde e Azul
```

Pode-se, também, atribuir o mesmo valor a várias variáveis.

[4]:
```
i = k = j = 0
print(i, j, k)
```

```
0 0 0
```

Exercício Use a função predefinida divmod para calcular o quociente e o resto da divisão de 17 por 5.

Nota: A função `divmod()` retorna um `tuple` (um par).

[5]:
```
quociente, resto = divmod(17, 5)
# confirmar o resultado
```

```
print(5*quociente+resto)
```

```
17
```

1.2 Tipos

Os principais tipos predefinidos do Python são:

Categoria	Tipo
Numéricos	`int`, `float`, `complex`
Texto	`str`
Coleções	`list`, `tuple`, `range`
Dicionário	`dict`
Conjuntos	`set`, `frozenset`
Booleano	`bool`
Binários	`bytes`, `bytearray`, `memoryview`
Funções	`builtin_function_or_method`

Como o Python é tipado (como já se disse), temos que ter cuidado com as operações e o tipo dos operandos. Por exemplo, tente correr o seguinte código:

[ ]:
```
from datetime import date
ano_atual = date.today().year

ano_nascimento = input()
print("Você tem {} anos.".format(ano_atual - ano_nascimento) )
```

Porque é que o exemplo anterior corre mal? Porque a variável `ano_nascimento` que fica com o resultado da função `input()` é do tipo `str`. Por isso, o Python não consegue fazer a subtração: `ano_atual - ano_nascimento` entre um `int` e uma `str`.

No código seguinte, assegura-se que a variável `ano_nascimento` é convertida em `int` com `int(ano_nascimento)`.

[ ]:
```
from datetime import date
ano_atual = date.today().year

ano_nascimento = input()
print("Você tem {} anos.".format(ano_atual - int(ano_nascimento)) )
```

1.2.1 Tipo de uma variável

Para investigar o tipo de uma variável, usa-se a função `type()`.

[ ]:
```
ano_nascimento = input()
type(ano_nascimento)
```

```
[ ]: planetas = [ 'Mercúrio', 'Vénus', 'Terra', 'Marte', 'Júpiter', 'Saturno',␣
      ↪'Urano', 'Neptuno' ]
     type(planetas)
```

```
[ ]: planetas[0]
```

```
[ ]: type(planetas[0])
```

Uma variável em Python não tem que ter sempre o mesmo tipo. A qualquer momento pode-se atribuir um valor de outro tipo à mesma variável.

```
[ ]: x = 3.14159
     x = 'Salazar Slytherin'
     type(x)
```

Caso seja preciso confirmar se uma variável é de um determinado tipo, pode-se usar a função `isinstance()`.

```
[ ]: x = 3.14159
     isinstance(x,int)
```

Exercício Prove que `'Amor de perdição'` e `"Amor de perdição"`são ambas do tipo `str` e que são iguais, sendo uma escrita com `'` e outra com `"`.

```
[ ]: print( type('Amor de perdição'), type("Amor de perdição"), 'Amor de perdição'␣
      ↪== "Amor de perdição" )
```