Programação I / Introdução à Programação

Capítulo 2, "Variables, expressions and statements"

João Pedro Pedroso

2024/2025

Última aula:

- Linguagens naturais e formais
- Tipos de erros em programação
- Debugging
- Python: "Hello, world!"

Hoje:

- Variáveis, Expressões, Instruções

Valores e tipos de dados

- Valor → elemento fundamental utilizado em programas
  - letras, números, ...
- Valores são classificados em classes ou tipos de dados
  - 4 → inteiro
  - "Hello, world!" → sequência de carateres / *string*
    - entre aspas "" ou apóstrofos ''
  - Podemos saber qual a classe de um valor com a função `type`:

```
>>> type("Hello, World!")
<class 'str'>
>>> type(17)
<class 'int'>
>>> type(3.2)
<class 'float'>
```

Strings: diferentes representações

```
>>> type('This is a string.')
<class 'str'>
>>> type("And so is this.")
<class 'str'>
>>> type("""and this.""")
<class 'str'>
>>> type('''and even this...''')
<class 'str'>
>>> print('''"Oh no", she exclaimed, "Ben's bike is broken!"''')
"Oh no", she exclaimed, "Ben's bike is broken!"
>>>
```

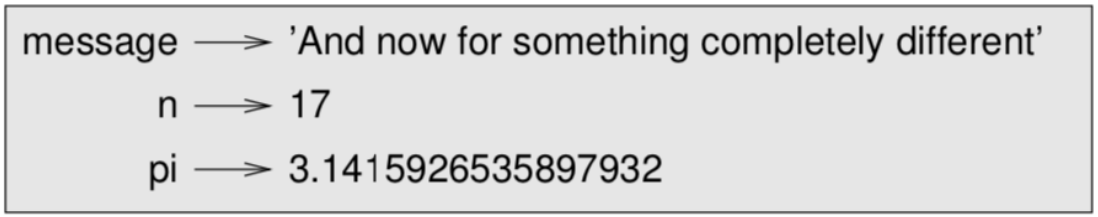
Variáveis

- Variável → nome que se refere a um *valor*
- Atribuição → instrução que dá um valor a uma variável

```
>>> message =  'And now for something completely different'
>>> n = 17
>>> pi = 3.14159265358979 32
```

  - sinal = → *token* de atribuição
  - vincula o nome do lado esquerdo ao valor do lado direito
  - ler, e.g., *n fica com o valor 17*

- Diagrama de estado




```
>>> message
'And now for something completely different'
>>>
```

Variáveis

Reatribuição:

```
>>> day = "Thursday"
>>> day
'Thursday'
>>> day = 21
>>> day
21
```

Nomes de variáveis / palavras reservadas

- Começam com uma letra, seguida de letras, números ou sublinhado[1]
- [Podem ter letras com acentos]
- Não podem ter espaços ou tabulações
- Não podem ser palavras reservadas de Python:

and	def	exec	if	not	return
assert	del	finally	import	or	try
break	elif	for	in	pass	while
class	else	from	is	print	yield
continue	except	global	lambda	raise	

[1] mais tarde veremos que isto não é verdade

Instruções

Instrução → em inglês: *statement*

- Sequência de *tokens* que o Python pode executar
  - atribuições

```
>>> day = "Thursday"
```

  - veremos na próxima aula:
    - `while`
    - `for`
    - `if`
    - `import`
    - `...`

Avaliação de expressões

- Expressão: sequência se valores, variáveis, operadores e chamadas a funções
- Se se escrever no prompt, o Python avalia-a e imprime o resultado

```
>>> 1 + 1
2
>>> len("hello")
5
```

- Avaliação de uma expressão → produz um valor
  - expressões podem aparecer no lado direito de uma atribuição
  - um valor é por si só uma expressão
  - uma variável é por si só uma expressão

```
>>> x = 1 + 1
>>> x
2
```

Operadores e operandos

Exemplos de expressões com *operadores e operandos*:

```
20+32     hour-1     hour*60+minute     minute/60     5**2     (5+9)*(15-7)
```

Ordem das operações:

- P parêntesis → ()
- E exponenciação → **
- MD multiplicação, divisão → * /
- AS adição, subtração → + -

Variáveis, Expressões, Instruções

```
x = 3
y = x**2
print(y)
```

Funções para converter tipos

Em Python pode-se facilmente converter valores `int`, `float` e `str`:

```
>>> int(3.14)
3
>>> int(3.9999)     # This doesn't round to the closest int!
3
>>> int(-3.999)     # Note that the result is closer to zero
-3
>>> int(minutes / 60)
10
>>> int("2345")     # Parse a string to produce an int
2345
>>> int(17)         # It even works if arg is already an int
17
>>> int("23 bottles")
Traceback (most recent call last):
File "<interactive input>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '23 bottles'
>>> float(17)
17.0
>>> float("123.45")
123.45
>>> str(17)
'17'
```

Operações com sequências de carateres (*strings*)

```
>>> message = "hello"
>>> message - 1          # Error
>>> "Hello" / 123       # Error
>>> message * "Hello"   # Error
>>> "15" + 2            # Error
```

- Concatenação: operador +

```
1  fruit = "banana"
2  baked_good = " nut bread"
3  print(fruit + baked_good)
```

- Repetição: operador *

```
"Fun"*3                      -->  "FunFunFun"
"Fun" + "Fun" + "Fun"        -->  "FunFunFun"
```

Input

Função prédefinida no Python para ler valores do teclado:

```
name = input("Please enter your name: ")
```

- devolve sempre uma string
- valores numéricos: necessário converter a partir de string

```
age = input("Please enter your age: ")
age = int(age)
```

Composição

- Podemos combinar elementos de um programa:
  variáveis, expressões, instruções, chamadas a funções
- Podemos compor um bloco maior a partir de blocos pequenos
- Exemplo: determinar
  Area $= \pi R^2$

```
1  response = input("What is your radius? ")
2  r = float(response)
3  area = 3.14159 * r**2
4  print("The area is ", area)
```

- Compondo:

```
1  r = float(input("What is your radius? "))
2  print("The area is ", 3.14159 * r**2)
```

O operador módulo

- Símbolo % (percentagem)
- Trabalha com valores inteiros
- Devolve o resto da divisão do primeiro valor pelo segundo
- Tem precedência igual à da multiplicação

```
>>> q = 7 // 3      # This is integer division operator
>>> print(q)
2
>>> r = 7%3
>>> print(r)
1
```

Funções

```
def f(x):
    return x*x

print("f(3) = ", f(3))
```

- funções matemáticas: `import math`

```
1  import math
2  def f(x):
3      return math.log(x*x)
4
5  print("f(3) = ", f(3))
```

- funções com vários parâmetros:

```
1  def soma(x,y):
2      z = x + y
3      return z
```

Noções estudadas

assignment statement
assignment token
composition
concatenate
data type
evaluate
expression
float
floor division
int
keyword
modulus operator
operand
operator
rules of precedence
state snapshot
statement
str
value
variable
variable name

Próxima aula

- "Fluxo num programa"