Programação I / Introdução à Programação

Capítulo 3, "Program Flow"

João Pedro Pedroso

2024/2025

Última aula:

- Módulo `turtle`
- Ciclos `for`

```
1  import math
2  for x in [0,1,2,3,4,5]:
3      print(x, math.sqrt(x))
```

  - `for` → permite *percorrer* uma lista

Última aula:

- Módulo `turtle`
- Ciclos `for`

```
1  import math
2  for x in [0,1,2,3,4,5]:
3      print(x, math.sqrt(x))
```

  - `for` → permite *percorrer* uma lista

- Função `range`

```
1  import math
2  for x in range(6):
3      print(x, math.sqrt(x))
```

Última aula:

- Execução condicional

```
1  for x in range(5):
2      if x%2 == 0:
3          print(x, "é par")
4      else:
5          print(x, "é ímpar")
```

- Podemos criar *variáveis com valores lógicos*:

```
1  for i in range(1,21):
2      is_even = i % 2 == 0
3      is_mult_3 = i % 3 == 0
4      if is_even and is_mult_3:
5          print(i, "is even and multiple of 3")
```

  - por vezes, tornam o código mais legível
  - podemos usar o resultado de uma expressão lógica em várias condições

Hoje:

- Fluxo num programa (continuação)

Função range

```
for x in range(5):   # 0, 1, 2, 3, 4
    print(x)

for x in range(10): # 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
    print(x)

for x in range(3,10): # 3, 4, 5, 6, 7, 8, 9
    print(x)

for x in range(3,10,2): # 3, 5, 7, 9
    print(x)
```

- `range(n)` valores inteiros de 0 até $n-1$ inclusive
- `range(i,n)` $\rightarrow$ valores inteiros de i até $n-1$ inclusive
- `range(i,n,d)` valores inteiros i, $i+d$, $i+2d$, ... inferiores a n

Ajuda em Python

```
>>> help(range)
```

ou

`https://docs.python.org/3/library/stdtypes.html#typesseq-range`

meta-notação descreve a sintaxe do Python, sem fazer parte dela

- elementos facultativos entre parêntesis retos
- palavras reservadas em *bold*
- itálico: e.g., *variable* → pode ser substituído por qualquer nome válido para variáveis
- ...

Atribuição e *atualização de variáveis*

- Inicialização: primeiro valor que se atribuí a uma variável

```
i = 0
```

- Incrementar: aumentar o valor em uma unidade

```
i = i + 1
i += 1      # forma mais curta (e mais habitual)
```

- Decrementar: diminuir o valor em uma unidade

```
i -= 1
```

- Outros operadores:

```
v = v + k    ↔   v += k
v = v - k    ↔   v -= k
v = v * k    ↔   v *= k
v = v / k    ↔   v /= k
v = v // k   ↔   v //= k
v = v ** k   ↔   v **= k
```

Instrução `for`

- Permite *percorrer* os valores de uma lista
- Podemos utilizar para calcular a soma de valores numa lista
  - exemplo: `[5, 6, 32, 21, 9]`

Instrução `for`

- Permite *percorrer* os valores de uma lista
- Podemos utilizar para calcular a soma de valores numa lista
  - exemplo: `[5, 6, 32, 21, 9]`

```
1  numbers = [5, 6, 32, 21, 9]
2  running_total = 0
3  for number in numbers:
4      running_total += number
5  print(running_total)
```

- Verifique o que acontece usando o Python Tutor: `http://pythontutor.com`

Instrução `while`

Permite iterar sem saber *a priori* quantas vezes se vai executar o ciclo

```
while <CONDITION>:
    <STATEMENTS>
```

Exemplo:

```
n=6
current_sum = 0
i=0
while i <= n:
    current_sum += i
    i += 1
print(current_sum)
```

A sequência de Collatz

Nem sempre é fácil concluir que um ciclo while não termina.
Exemplo:

- Calcular a Sequência de Collatz para um dado $n > 0$:
  Enquanto n não for igual a 1, gerar o próximo elemento da sequência usando a função:

$$f(n) = \begin{cases} n/2 & \text{se } n \text{ é par} \\ 3n+1 & \text{se } n \text{ é ímpar} \end{cases}$$

```
def sequencia(n):
    print(n)
    while n != 1:
        if n%2 == 0:   # n é par
            n = n//2
        else:          # n é ímpar
            n = 3*n+1
        print(n)
```

`sequencia(1)`	1
`sequencia(2)`	2, 1
`sequencia(3)`	3, 10, 5, 16, 8, 4, 2, 1
`sequencia(4)`	4, 2, 1
`sequencia(5)`	5, 16, 8, 4, 2, 1
`sequencia(6)`	6, 3, 10, 5, 16, 8, 4, 2, 1
...	
`sequencia(27)`	27, 82, 41, 124, 62, 31, 94, 47, 142, 71, 214, 107, 322, 161, 484, 242, ...
	$\rightarrow$ *são necessárias 111 iterações para chegar a 1*

- Não sabemos se esta função termina para todo $n > 0$.
- `http://en.wikipedia.org/wiki/Collatz_conjecture`

Seguir o fluxo de um programa

`sequencia(3)`

n	output até ao momento
3	3,
10	3, 10,
5	3, 10, 5,
16	3, 10, 5, 16,
8	3, 10, 5, 16, 8,
4	3, 10, 5, 16, 8, 4,
2	3, 10, 5, 16, 8, 4, 2,
1	3, 10, 5, 16, 8, 4, 2, 1.

Ciclo `for` ou `while`?

Iteração definida, ciclos *for* para

- iterar sobre sequências aritméticas
- iterar sobre listas ou tuplos

Iteração indefinida, ciclos *while* nos outros casos

Saída e continuação num ciclo

`break` sai a meio de um ciclo

`continue` passa ao início da próxima iteração

- se a condição de paragem se verificar, termina

```
for i in [12, 16, 17, 24, 29]:
    if i % 2 == 1: # se é ímpar
       break       #  ... termina o ciclo
    print(i)
print("fim")
```

Saída e continuação num ciclo

`break` sai a meio de um ciclo

`continue` passa ao início da próxima iteração

- se a condição de paragem se verificar, termina

```
for i in [12, 16, 17, 24, 29]:
    if i % 2 == 1: # se é ímpar
        break      #  ... termina o ciclo
    print(i)
print("fim")
```

```
12
16
fim
```

```
for i in [12, 16, 17, 24, 29, 30]:
    if i % 2 == 1:     # se é impar
        continue       # ...passa ao próximo
    print(i)
print("fim")
```

Saída e continuação num ciclo

`break` sai a meio de um ciclo

`continue` passa ao início da próxima iteração

- se a condição de paragem se verificar, termina

```
for i in [12, 16, 17, 24, 29]:
    if i % 2 == 1: # se é ímpar
       break       #  ... termina o ciclo
    print(i)
print("fim")
```

```
12
16
fim
```

```
for i in [12, 16, 17, 24, 29, 30]:
    if i % 2 == 1:     # se é impar
       continue        # ...passa ao próximo
    print(i)
print("fim")
```

```
12
16
24
30
fim
```

Variantes de ciclos: paragem a meio

```
import random
rng = random.Random() # "random number generator"
number = rng.randrange(1, 1000) # Get random number between [1 and 1000)
guesses = 0
message = ""
while True:
    guess = int(input(message + "\nGuess my number between 1 and 1000: "))
    guesses += 1
    if guess > number:
        message += str(guess) + " is too high.\n"
    elif guess < number:
        message += str(guess) + " is too low.\n"
    else:
        break
input("\n\nGreat, you got it in "+str(guesses)+" guesses!\n\n")
```

Noções estudadas

Muitas noções importantes → ler com atenção final do capítulo!

atributo estado, ou valor, que pertence a um objeto determinado (e.g., a cor);

biblioteca *standard* coleção de módulos que fazem parte da distribuição normal do Python;

ciclo `for` instrução Python˜para repetir as instruções que escrevermos no *corpo/bloco* do ciclo;

condição de paragem condição que, quando ocorre, termina a execução de um ciclo; a mais habitual, e a de não haver mais elementos a atribuir à variável do ciclo.

controlo do fluxo manipulação do fluxo de execução;

corpo de um ciclo — conjunto de instruções dentro do ciclo (indentados relativamente à definição do ciclo);

determinístico propriedade de um processo que é repetível e previsível.

estado conjunto de valores para os atributos de um objeto;

gerador de números (pseudo-)aleatórios função de fornece a possibilidade de criar números com propriedades semelhantes às dos aleatórios;

instância objeto de uma classe (`alex` e `tess` instâncias da classe `Turtle`);

invocar ou *chamar* – executar um método de um objeto (`wn.exitonclick()` chama o método exitonclick do objeto `wn`);

iteração elemento de programação que permite repetir alguns passos de um programa;

método função associada a um objeto; quando é invocada, causa algum efeito no objeto;

módulo ficheiro contendo definições e instruções Python, destinadas a serem usadas por outros programadores, através de `import`;

número aleatório número gerado por um processo que torna impossível a sua previsão com 100% de certeza (processo estocástico);

número pseudo-aleatório número que, sendo em rigor gerado por um processo determinístico, possuí propriedades semelhantes à de um aleatório;

objeto elemento informação relativo a dados de um determinado tipo, referido por uma variável;

`range` função incorporada no Python para gerar sequências de inteiros;

sequencial comportamento de um programa passo a passo, da primeira até à última instrução;

tela *(canvas)* superfície dentro de uma janela, onde se desenha;

variável do ciclo variável à qual é atribuído um valor diferente a cada iteração do ciclo, que é usada como critério de paragem;

Próxima aula

- Funções