37. Programmare in Python

Corso di Informatica

Corso di Laurea in Matematica (D.M. 270/04) - A.A. 2020/2021

Angelo Cardellicchio
angelo.cardellicchio@uniba.it

11/01/2021

Outline

- Python 101
- Programmazione strutturata
- Funzioni

Python 101

- Le parentesi:
  - ***tonde*** *indicano le chiamate ad una funzione, oltre che la precedenza delle operazioni;*
  - ***quadre*** *sono usate per l'accesso agli elementi di una lista;*
  - ***graffe*** *sono usate per la creazione dei dizionari.*
- Non si utilizza il punto e virgola al termine di un'istruzione
- L'ambito è determinato mediante la tabulazione
  - *Codice allo stesso livello di tabulazione, o indentatura, è nello stesso ambito*
  - *La parentesi graffa che in C e C++ determinava la partenza di un ambito è sostituita dai due punti; quella di chiusura non c'è*

Programmazione strutturata (1)

- Il costrutto **`if`**:

```
>>> a = 5
>>> if a < 5:
...     print('a è minore di 5')
... elif a == 5:
...     print('a è uguale a 5')
... else:
...     print('a è maggiore di 5')
...
```

- Il costrutto **`switch`** non esiste in Python
  - *Esistono però numerose vie alternative, non necessariamente gli* **`if/elif/else`**

Programmazione strutturata (2)

- Il costrutto **for**:

```
>>> vals = [0,1,2,3,4]
>>> for i in vals:
...     print(i)

>>> string = "Python"
>>> for char in string:
...     print(char)
```

- La differenza fondamentale rispetto all'implementazione del **for** negli altri linguaggi di programmazione è che Python itera su una *sequenza* come una lista o una stringa

Programmazione strutturata (3)

- Il costrutto **`while`**:

```
>>> i = True
>>> while i:
...     if randint(-5, 5) > 0:
...         print("Continuo!")
...     else:
...         print("Esco!")
...             i = False
```

- Una piccola nota sui valori booleani in Python: questi, infatti, iniziano con la lettera maiuscola, quindi **non si tratta di un refuso**.

Programmazione strutturata (4)

- La funzione **`range()`** ci permette di evitare di definire manualmente la sequenza per iterare su un **`for`**:

```
range(i, j, s)
```

- **`i`** è l'estremo inferiore, **`j`** l'estremo superiore, **`s`** il passo di incremento
  - *Nota: per stampare a schermo i valori presenti nel range, dovremo convertirlo in lista mediante cast*

```
>>> r = range(0, 5, 1)
>>> print(list(r))
[0, 1, 2, 3, 4]
```

Programmazione strutturata (5)

- **`i`** è di default pari a 0, **`s`** ad 1

```
>>> r = range(5)
>>> print(list(r))
[0, 1, 2, 3, 4]
```

- Possiamo anche specificare un range discendente, scegliendo i valori in modo che **`i > j`**, **`s < 0`**

```
>>> r = range(5, 1, -1)
>>> print(list(r))
[5, 4, 3, 2]
```

- In accoppiata con la funzione **`len()`**, **`range()`** può essere usata per iterare su una lista

```
>>> l = ['Pippo', 'Pluto', 5, 'Paperino']
>>> for i in range(len(l)):
...     print(l[i])
```

Programmazione strutturata (6)

- Le istruzioni **`break`** e **`continue`** sono analoghe alle controparti C/C++

```
>>> while (True):
...     if randint(-5, 5) > 0:
...         print("Continuo!")
...         continue
...     else:
...         print("Esco!")
...         break
```

Programmazione strutturata (7)

- È possibile usare l'istruzione **`else`** in accoppiata con l'istruzione **`break`**
- Questa particolare forma fa sì che **l'`else` venga chiamata solo quando la `break` non viene chiamata**

```
>>> for i in range(2, 10):
...     for j in range(2, 5):
...         if i % j == 0:
...             break
...     else:
...         print('{} non è divisibile per {}'.format(i, j))
...
5 non è divisibile per 4
7 non è divisibile per 4
```

- ***Domanda: come funziona questo programma?***

Funzioni (1)

- La sintassi per definire una funzione è:

```
def nome_funzione(parametri):
    # istruzioni
    return valore_ritorno
```

- Notiamo che:
  - *non è necessario un tipo di ritorno, e non esiste il void;*
  - *non è necessario definire il tipo di ciascuno dei parametri passati;*
  - *sono consentiti i parametri di default*

Funzioni (2)

- Ecco alcuni esempi:

```
>>> def calcola_area_quadrato(lato):
...     return lato * lato
...
>>> calcola_area_quadrato(5)
25
```

```
>>> l = [1,2]
>>> def raddoppia_lista(valore):
...     for i in range(len(valore)):
...             l.append(l[i] * 2)
...
>>> raddoppia_lista(l)
>>> l
[1, 2, 2, 4]
```

```
>>> def genera_lista_casuale(lunghezza=5):
...     l = []
...     for i in range(lunghezza):
...             l.append(randint(0, 10))
...     return l
...
>>> genera_lista_casuale()
[3, 1, 2, 0, 6]
>>> genera_lista_casuale(10)
[7, 9, 1, 10, 2, 4, 9, 1, 4, 8]
```

Funzioni (3)

- Il passaggio dei parametri **avviene esclusivamente per valore**
- Per cui, potremmo trovarci davanti ad errori 'inspiegabili':

```
>>> def raddoppia(valore):
...     valore = valore * 2
...
>>> val = 1
>>> raddoppia(val)
>>> val
1
```

- Come ha fatto a funzionare quindi la funzione **`raddoppia_lista`**?
  - *Semplice: abbiamo usato dei metodi su un tipo mutabile!*

Funzioni (4)

- L'istruzione **pass**, in ultimo, non fa assolutamente niente

```
>>> def passa():
...     pass
...
>>> passa()
```

- Può essere utile come placeholder o nella definizione dei metodi delle classi astratte

Domande?

42