Programmazione in Python

Funzioni

Dario Pescini - Mirko Cesarini [1]

Università degli Studi di Milano-Bicocca

Dipartimento di Statistica e Metodi Quantitativi

dario.pescini@unimib.it

Funzioni: definizione e chiamata

```
def nomeDellaFunzione( lista_parametri ):
    parte esecutiva
    return valore
```

lista_parametri: definizione

- La lista dei parametri fornita nell'intestazione serve all'interprete per definire i parametri che verranno passati alla funzione nella chiamata. Lista parametri formali.
- É una lista di elementi separati da virgola.
- Tramite `nomeParametro` la funzione accederà al valore ad esso associato.
- L'associazione `nomeParametro` valore avviene solo in fase di chiamata.

```
nomeDellaFunzione( lista_valori )
```

lista_parametri: chiamata

- `lista_valori` è la lista di valori da passare alla funzione separati da virgola. lista parametri attuali
- L'associazione tra parametri formali e attuali avviene in maniera posizionale.
- parametri formali e attuali devono essere in egual numero.

Funzioni: approfondimento parametri

Nella definizione di una funzione si possono avere parametri

- senza valore di default
  ```
  def funzione(par1, par2, par3):
  ```
- con valore di default
  ```
  def funzione(par1='pippo', par2=7,
  par3=[5]):
  ```
- di tipo variabile
  ```
  def funzione(*args, **keywords):
  ```

Funzioni: approfondimento parametri

Nella chiamata della funzione posso passare argomenti in maniera

- puramente posizionale
  `funzione('pippo', 7, [5])`
- con denominazione (keyword)
  `funzione(par1='pippo', par2=7, par3=[5])`
- lista argomenti di cardinalità variabile

Funzioni: approfondimento parametri

- il valore di default per un parametro lo rende opzionale
- tutti i parametri senza default devono precedere quelli con default
- tutti i parametri senza default devono essere specificati in fase di chiamata
- un argomento non può essere specificato più di una volta
- la specifica degli argomenti tramite keyword è indipendente dalla posizione
- è possibile la specifica degli argomenti mista: posizionale e tramite keywords (prima i posizionali)

Funzioni: approfondimento parametri

Il valore di default per un parametro lo rende opzionale

```
def funzione(par1, par2, par3=7.0):
    print "par1: ", par1, "par2: ", par2, "par3: ", par3

funzione('pippo', 5)
funzione(par1='pippo', par2=5)
funzione('pippo', 5, 'paperino')
```

Funzioni: approfondimento parametri

Tutti i parametri senza default devono precedere quelli con default

```
def funzione(par1, par2, par3=7):
    print "par1: ", par1, " par2: ", par2, " par3: ", par3

funzione('pippo', 5.0)
funzione('pluto', 5.0, 'paperino')
```

```
def funzione(par1='pippo', par2, par3):
    print "par1: ", par1, " par2: ", par2, " par3: ", par3
```

Genera un errore di sintassi.

Funzioni: approfondimento parametri

Tutti i parametri senza default devono essere specificati in fase di chiamata

```
def funzione(par1, par2, par3=7):
    print "par1: ", par1, " par2: ", par2, " par3: ", par3

funzione(par3='paperino')
```

Genera un errore di sintassi.

Funzioni: approfondimento parametri

Un argomento non può essere specificato più di una volta

```
def funzione(par1, par2, par3=7):
    print "par1: ", par1, " par2: ", par2, " par3: ", par3

funzione('pluto', 5.0, par1='paperino')
```

Genera un errore.

Funzioni: approfondimento parametri

La specifica degli argomenti tramite keyword è indipendente dalla posizione

```
def funzione(par1, par2, par3=7):
    print "par1: ", par1, " par2: ", par2, " par3: ", par3

funzione(par3='pluto', par1='paperino', par2=5.0)
```

Funzioni: approfondimento parametri

É possibile la specifica mista posizionale e tramite keywords (prima i posizionali)

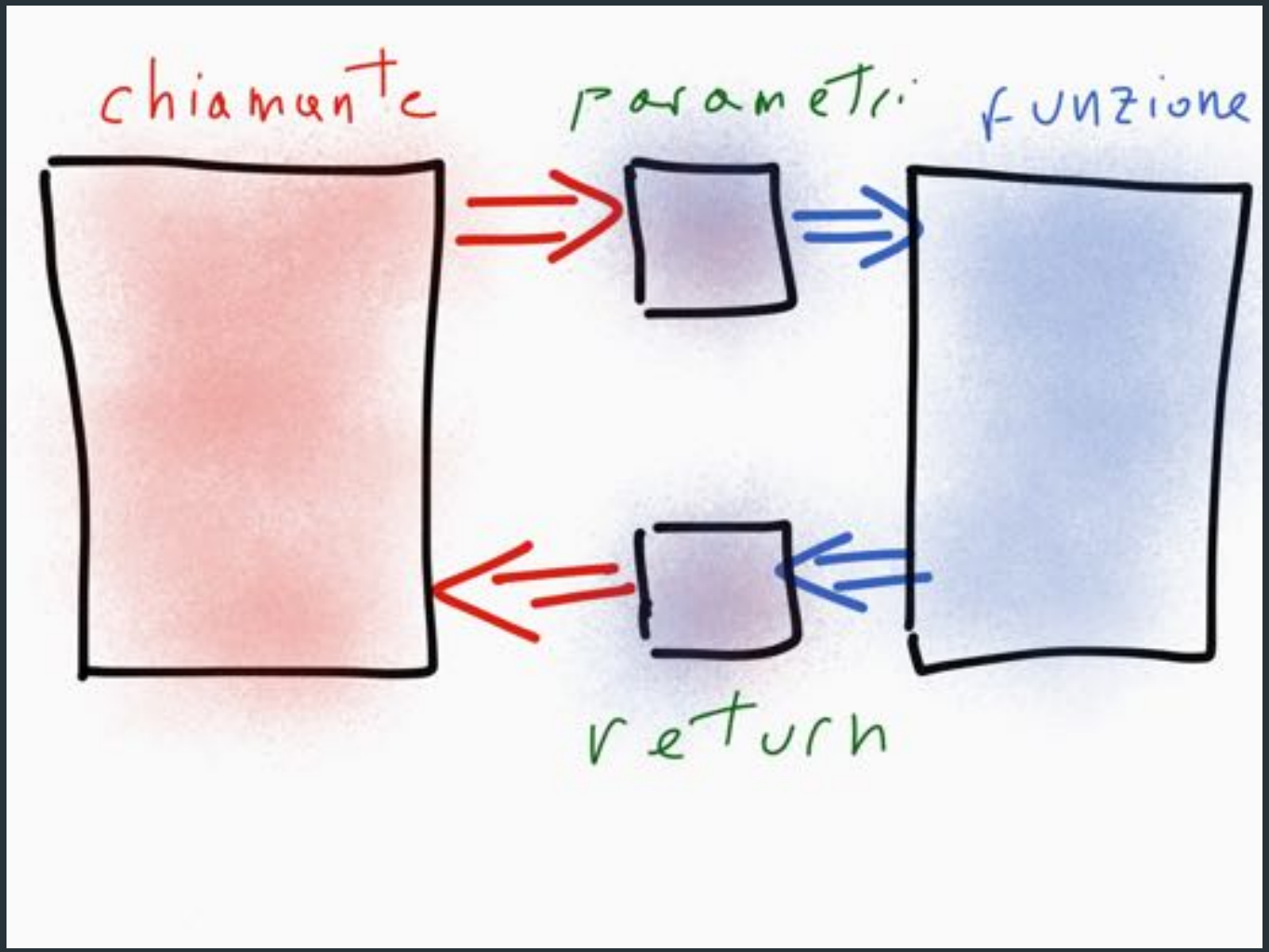
```
def funzione(par1, par2, par3=7):
    print "par1: ", par1, " par2: ", par2, " par3: ", par3

funzione('pluto', par3=5.0, par2='paperino')
```

Funzioni: approfondimento parametri

Vantaggi uso keyword=valore:

- maggior chiarezza nella chiamata
- passaggio solo dei valori senza default
- permette l'estensione del numero dei parametri della funzione garantendo backward compatibility

Funzioni: approfondimento parametri



Visibilità parametri

Funzioni: approfondimento return

In Python è possibile che una funzione restituisca più valori separati da , (una t-upla):

```
def funzione(numeratore=1, denominatore=1):
    if denominatore != 0:
        quoziente = numeratore/denominatore
        resto = numeratore - quoziente * denominatore
    else:
        quoziente = None
        resto = None
    return quoziente, resto

a, b = funzione(5, 2)
print a, b

print funzione(5, 2)
```

Funzioni: composizione

In Python è possibile comporre più funzioni:

```
def restoF(num, den, quoz):
    return num - quoz * den

def funzione(numeratore=1, denominatore=1):
    if denominatore != 0:
        quoziente = numeratore/denominatore
        resto = restoF(numeratore, denominatore, quoziente)
    else:
        quoziente = None
        resto = None
    return quoziente, resto

a, b = funzione(5, 2)
```

Funzioni: composizione

- il passaggio dei parametri avviene dall'esterno all'interno
- la valutazione della composizione dall'interno all'esterno