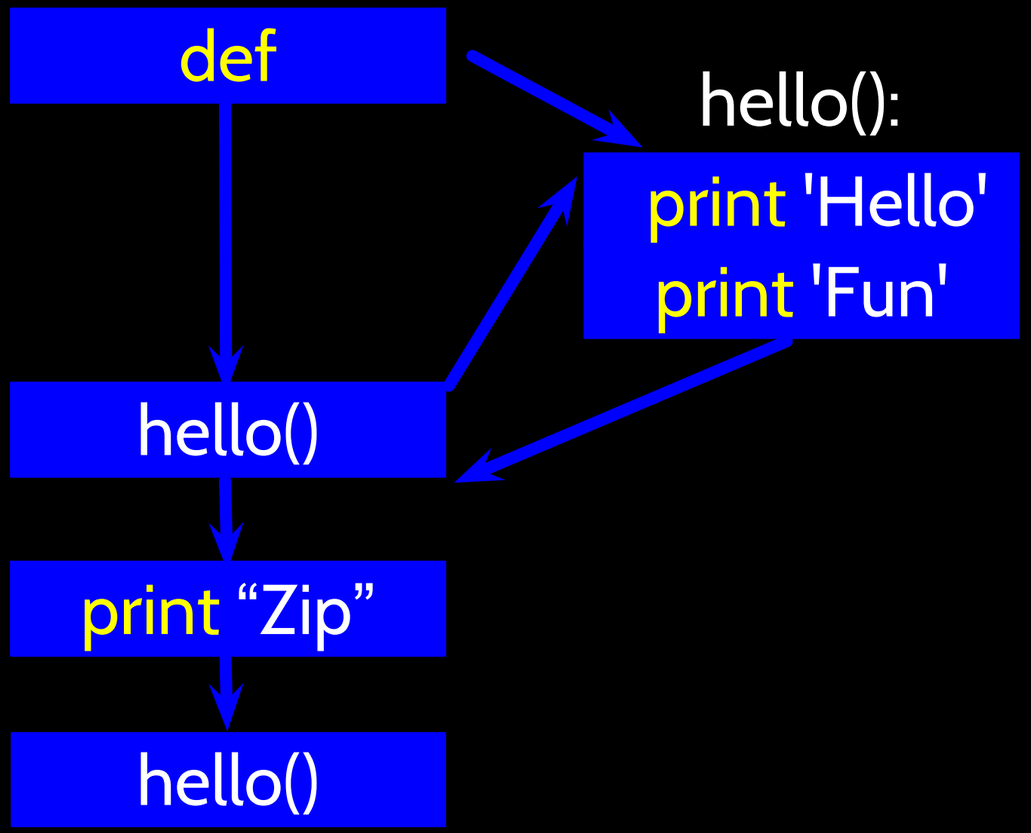
Funzioni

Capitolo 4

Python for Informatics: Exploring Information
www.pythonlearn.com

Passi memorizzati (e riusati)



Programma:

```
def thing():
    print 'Hello'
    print 'Fun'

thing()
print 'Zip'
thing()
```

Output:

Hello
Fun
Zip
Hello
Fun

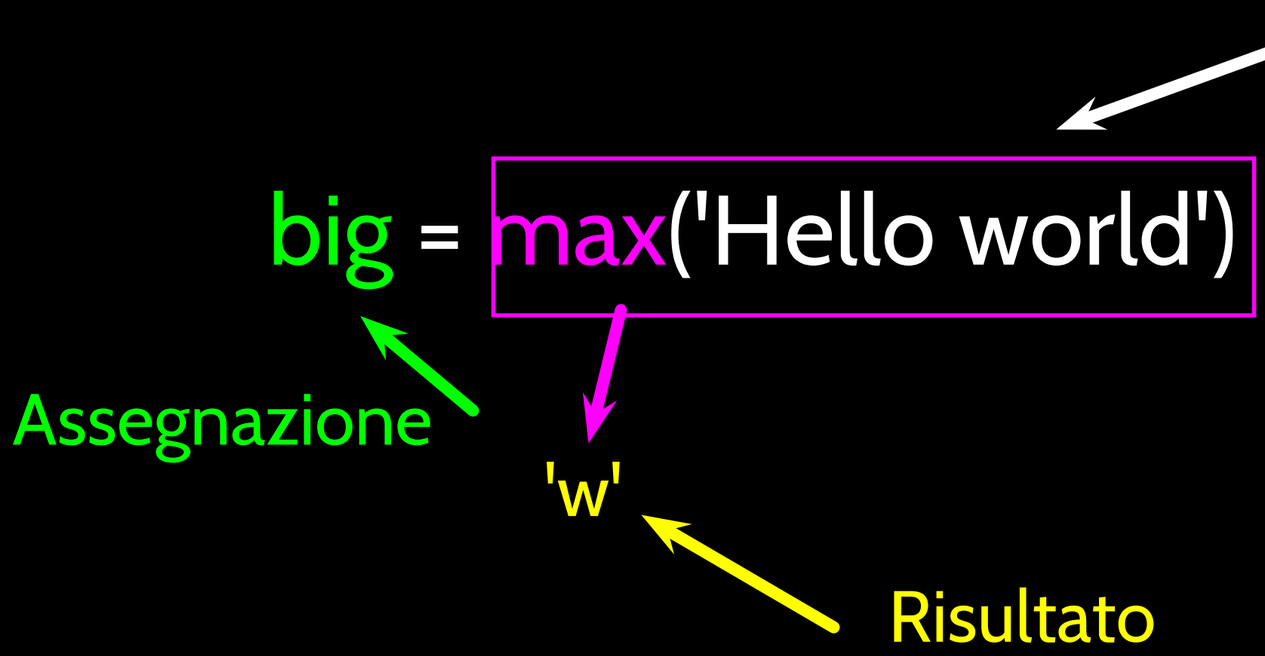
Chiamiamo questi pezzi di codice “funzioni”

Funzioni Python

- Ci sono due tipi di funzioni in Python.
  - funzioni built-in (incorporate) che sono disponibili come parti di Python - raw_input(), type(), float(), int() ...
  - Funzioni che possiamo definire e poi utilizzare
- Consideriamo i nomi delle funzioni built-in come "nuove" parole riservate (es. evitiamo di usarle come nomi di variabili)

Definizione di funzione

- In Python una funzione è un elemento di codice riutilizzabile che prende alcuni argomenti in input, esegue delle operazioni, e restituisce uno o più risultati.
- Per definire una funzione usiamo la parola riservata def
- Per chiamare/invocare la funzione usiamo il suo nomeby, parentesi e argomenti in una espressione.



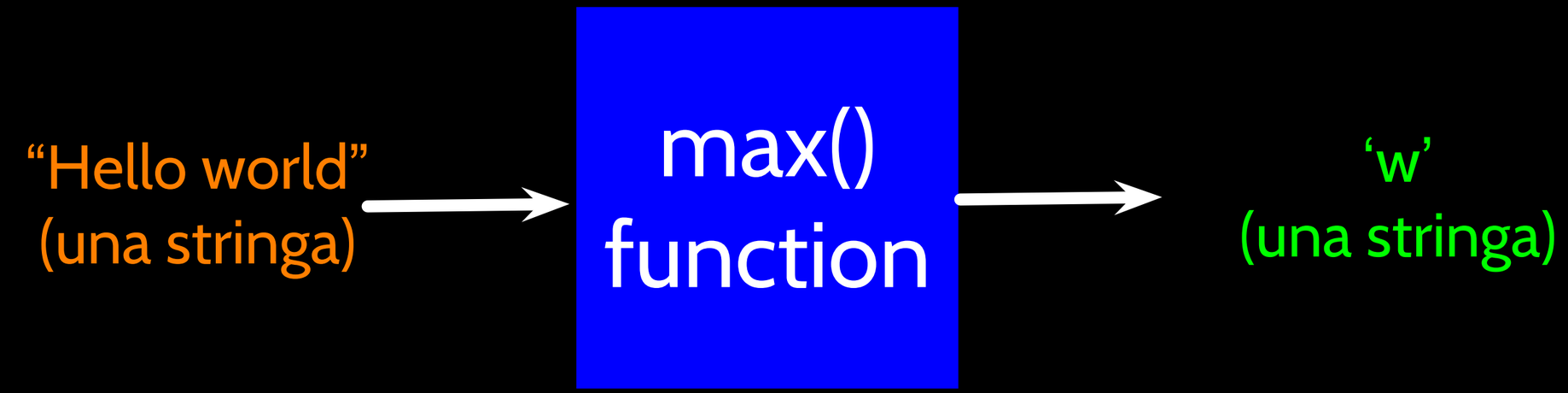

```
>>> big = max('Hello world')
>>> print big
w
>>> tiny = min('Hello world')
>>> print tiny

>>>
```

Funzione Max

```
>>> big = max('Hello world')
>>> print big
w
```

Una funzione è una porzione di codice. Una funzione accetta degli input e produce degli output.

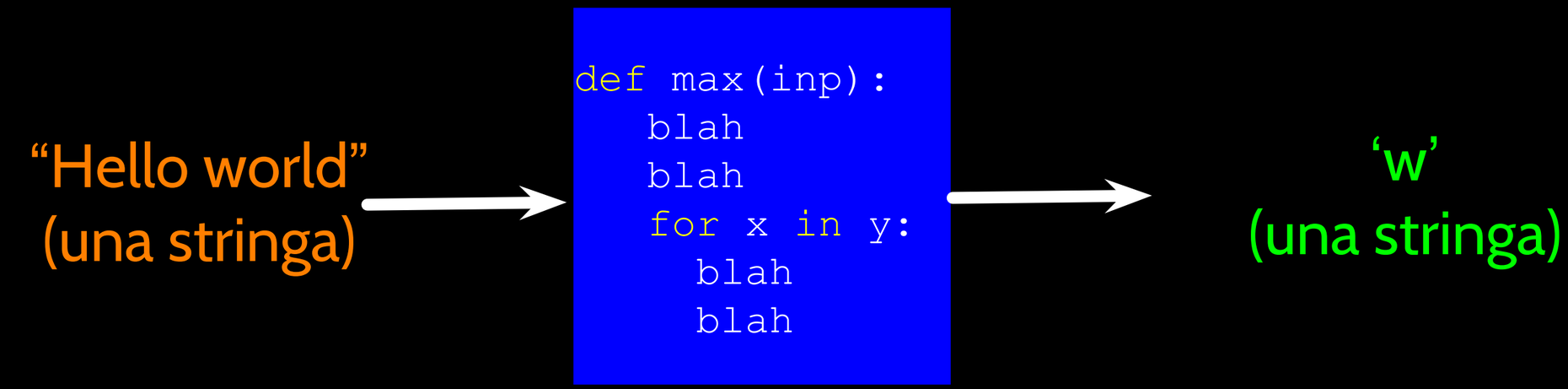


Guido ha scritto questo codice

Funzione Max

```
>>> big = max('Hello world')
>>> print big
w
```

Una funzione è una porzione di codice. Una funzione accetta degli input e produce degli output.



Guido ha scritto questo codice

Conversione tra tipi (Type)

- Utilizzando un intero e un numero con virgola mobile (float) in una espressione, l'intero è implicitamente convertito in virgola mobile
- La conversione è gestibile con le funzioni int() e float()

```
>>> print float(99) / 100
0.99
>>> i = 42
>>> type(i)
<type 'int'>
>>> f = float(i)
>>> print f
42.0
>>> type(f)
<type 'float'>
>>> print 1 + 2 * float(3) / 4 - 5
-2.5
>>>
```

Conversione di Stringhe

- E' possibile utilizzare int() e float() per convertire stringhe e numeri
- Si ottiene un errore se la stringa non contiene caratteri numerici

```
>>> sval = '123'
>>> type(sval)
<type 'str'>
>>> print sval + 1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str'
and 'int'
>>> ival = int(sval)
>>> type(ival)
<type 'int'>
>>> print ival + 1
124
>>> nsv = 'hello bob'
>>> niv = int(nsv)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int()
```

Costruirsi le proprie funzioni

- Possiamo creare una nuova funzione usando la keyword def seguita da parametri opzionali tra parentesi.
- Il corpo (body) della funzione va indentato.
- Questo definisce la funzione ma *NON* ne esegue il corpo (body)

```
def print_lyrics():
    print "I'm a lumberjack, and I'm okay.”
    print 'I sleep all night and I work all day.'
```

print_lyrics():

print "I'm a lumberjack, and I'm okay."
print 'I sleep all night and I work all day.'

```
x = 5
print 'Hello'

def print_lyrics():
    print "I'm a lumberjack, and I'm okay.”
    print 'I sleep all night and I work all day.'

print 'Yo'
x = x + 2
print x
```

Hello
Yo
7

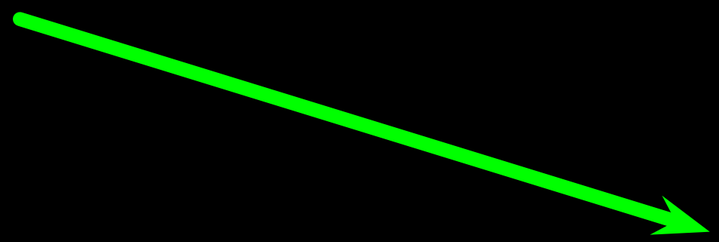
Definizioni e Utilizzo

- Una volte definita una funzione, possiamo chiamarla (call) (o invocarla - invoke) tutte le volte che serve
- Questo è il modello store and reuse

```
x = 5
print 'Hello'

def print_lyrics():
    print "I'm a lumberjack, and I'm okay.”
    print 'I sleep all night and I work all day.'

print 'Yo'
print_lyrics()
x = x + 2
print x
```



Hello
Yo
I'm a lumberjack, and I'm okay.
I sleep all night and I work all day.
7

Argomenti

- Un argomento è una valore che passiamo alla funzione come suo input nel momento in cui la chiamiamo.
- Usiamo gli argomenti per far eseguire alla funzione computazioni divese quando viene chiamata diverse volte
- Mettiamo gli argomenti tra parentesi dopo il nome della funzione

```
big = max('Hello world')
```

Argomento

Parametri

Un parametro è una variabile che usiamo nella definizione di una funzione. E' un "handle" che permette al codice nella funzione di accedere agli argomenti per una particolare chiamata ad una funzione.

```
>>> def greet(lang):
...     if lang == 'es':
...         print 'Hola'
...     elif lang == 'fr':
...         print 'Bonjour'
...     else:
...         print 'Hello'
...
>>> greet('en')Hello
>>> greet('es')Hola
>>> greet('fr')Bonjour
>>>
```

Ritorno dei Valori

Spesso una funzione prende gli argomenti, eregue delle operazioni, e ritorna un valore che verrò utilizzato come valore della funzione stessa nell' espressione di chiamata. La keyword return è utilizzata per questo.

```
def greet():
    return "Hello”

print greet(), "Glenn”
print greet(), "Sally"
```

```
Hello Glenn
Hello Sally
```

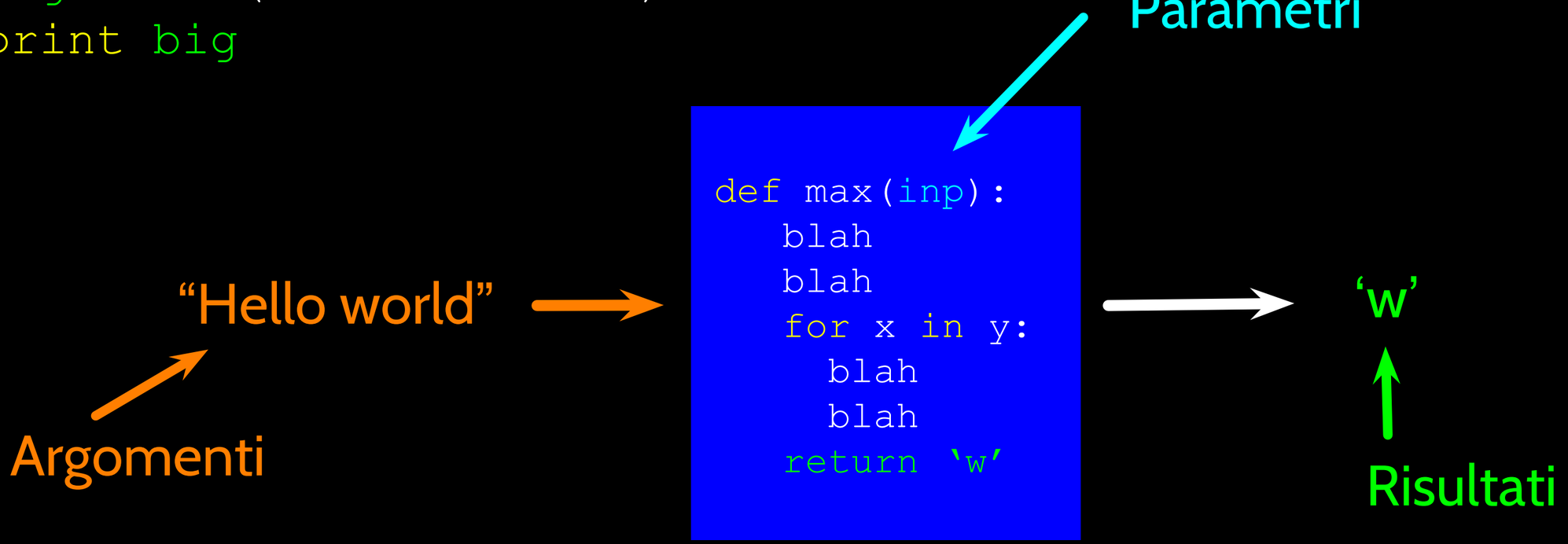
Ritorno dei Valori

- Una funzione “fruttuosa” è quella che produce un risultato (o ritorna un valore - return value)
- L’istruzione return chiude l’ esecuzione della funzione e “rispedisce indietro” il risultato della funzione

```
>>> def greet(lang):
...         if lang == 'es':
...             return 'Hola'
...         elif lang == 'fr':
...             return
'Bonjour'
...         else:
...             return 'Hello'
... >>> print greet
('en'),'Glenn'
Hello Glenn
>>> print greet('es'),'Sally'
Hola Sally
>>> print greet
('fr'),'Michael'
Bonjour Michael
>>>
```

Argomenti, Parametri, e Risultati

```
>>> big = max('Hello world')
>>> print big
w
```



Parametri Multipli / Argomenti

- Possiamo definire più di un parametro nella definizione della funzione
- Aggiungiamo semplicemente più argomenti quando chiamiamo la funzione
- Il numero di argomenti e parametri deve coincidere.

```
def addtwo(a, b):
    added = a + b
    return added

x = addtwo(3, 5)
print x
```

Void Funzioni (non-fruttuose)

- Quando una funzione non ritorna un valore, viene chiamata ''void''
- Funzioni che ritornano valori sono ''fruttuose''
- Funzioni Void sono ''non fruttuose''

Funzioni o non funzioni...

- Organizza il codice in “paragrafi” - cattura un pensiero completo e “nominalo”
- Non ripeterti - scrivilo una volta e riutilizzalo
- Se qualcosa diventa tropo lungo e complesso, dividilo in porzioni logiche e inseriscile in funzioni
- Crea una libreria di operazioni comuni che esegui più volte - magari condividila con amici...

Esercizio

Riscrivi il tuo calcolo della paga con l'ingremento di una volta e mezza per le ore eccedenti le 40 e crea una funzione chiamata computepay che usa due parametri (hours e rate).

Enter Hours: 45
Enter Rate: 10
Pay: 475.0

475 = 40 * 10 + 5 * 15

Summario

- Funzioni
- Funzioni incorporate (Built-In)
  - Conversioni (int, float)
- Argomenti
- Parametri

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School Of Information

… Insert new Contributors here

...