Python funzioni ed I/O

Loriano Storchi

loriano@storchi.org

http::/www.storchi.org/

Le funzioni

- Una funzione riceve in generale dei parametri che in generale sono indirizzi di memoria o valori che una funzione riceve in input per poter poi eseguire una certa sequenza di operazioni e ritornare un qualche risultato
- I parametri (parametri formali) sono le variabili che trovo nella definizione della funzione. Ad esempio in python **def funzione (a, b, c)**
- Mentre gli argomenti (o parametri attuali) sono le variabili passate in input al momento della chiamata alla funzione. Ad esempio in python **funzione(x, y, z)**

Le funzioni

- **Passaggio di parametri per valore**, implica che i **parametri attuali** vengono copiati nei **parametri formali**, e quindi la funzione lavora su una copia dei valori quando essi vengono modificati questo non si riflette si parametri effettivi
- **Passaggio per riferimento** in questo caso viene passato “un puntatore” quindi la funzione potra' modificare ad esempio il **parametro formale** e questo avra' affetto sul **parametro attuale**

Per valore o per riferimento

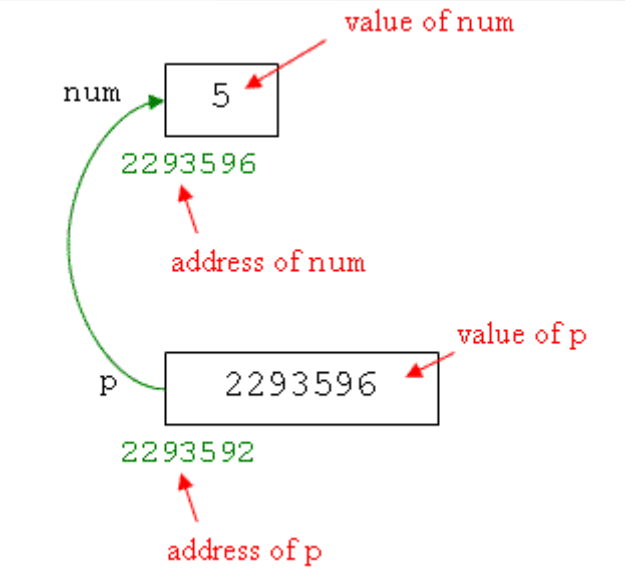
```
#include <iostream>

void funcval (int a, int b)
{
  a = 1;
  b = 1;
}

void funcref (int & a, int & b)
{
  a = 1;
  b = 1;
}

int main (int argc, char ** argv)
{
  int a, b;

  a = 0;
  b = 0;

  funcval (a, b);
  std::cout << "a: " << a << " b: " << b << std::endl;

  funcref (a, b);
  std::cout << "a: " << a << " b: " << b << std::endl;

  return 0;
}
```

```
a: 0 b: 0
a: 1 b: 1
```

Breve digressione: i puntatori

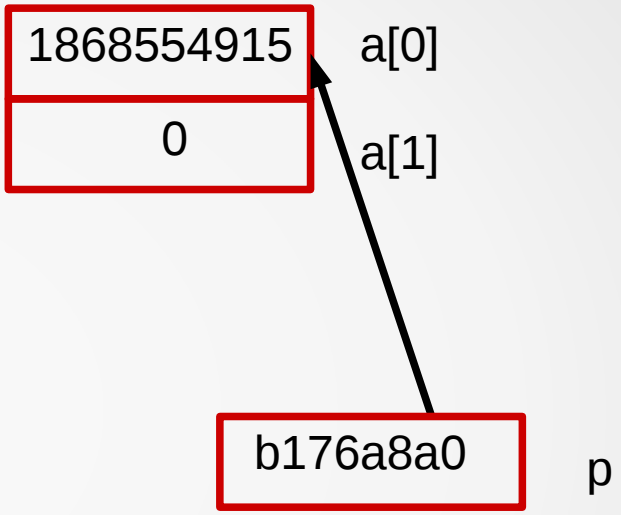
- Un puntatore e' una variabile “speciale” che contiene l'indirizzo ad una zona della memoria

```
#include <stdio.h>

void func (int * a, int * b)
{
  *a = 1;
  *b = 1;
}

int main (int argc, char ** argv)
{
  int a, b;

  a = 0;
  b = 0;

  func (&a, &b);
  fprintf (stdout, "a: %d b: %d \n", a, b);

  return 0;
}
[redo@banquo functionsc (master)]$ ./pointer
a: 1 b: 1
```



Breve digressione 2 per i piu' curiosi

```c
#include <stdio.h>

int main (int argc, char ** argv)
{
  int a[2];
  char * p;

  a[0] = 1868654915;
  a[1] = 0;

  p = (char *) a;
  fprintf (stdout, "%d %d %d %x\n", sizeof(int),
      sizeof(double), sizeof(p), a);
  fprintf (stdout, "%x %s \n", p, p);

  return 0;
}
```




```
[redo@banquo functionsc (master)]$ ./poi
4 8 8 b176a8a0
b176a8a0 Ciao
```

esadecimale

Funzioni in python

- Per definire una funzione in python si usa la parola chiave def
- Facciamo un semplice esempio una funzione che calcola il valore medio data in input una lista di numeri

```
def calcola (vals):
  sum = 0.0
  for v in vals:
    sum += v

  return sum/len(vals)

valori = [1.1, 4.0, 5.0, 9.5, 5.6]
m = calcola(valori)
print "valore medio: ", m
[redo@buchner functions]$ python meanfunc.py
valore medio:  5.04
```

Funzioni in python

- Un funzione in python puo' tonare piu' di un valore

Scope: n esiste solo all'interno della funzione

```
import math

def calcola (vals):
  n = float(len(vals))

  m = 0.0
  for v in vals:
    m += v
  m = m / n

  s = 0.0
  for v in vals:
    s = (v-m)**2
  s = s / n
  s = math.sqrt(s)

  return m, s

valori = [1.1, 4.0, 5.0, 9.5, 5.6]
m, s = calcola(valori)
print "valore medio: ", m , " stdev: ", s
```

Funzioni in python

- Una funzione puo' avere parametri di default. Ad esempio una funzione che divide tutti gli elementi della lista per un dato numero

```
def divide (vals, d = 2.0):
  res = []
  for v in vals:
    res.append(v / d)

  return res

vals = [1.0, 3.5, 5.6, 7.8]
print vals
print divide(vals)
print divide(vals, 3.0)
[redo@buchner functions]$ python divide.py
[1.0, 3.5, 5.6, 7.8]
[0.5, 1.75, 2.8, 3.9]
[0.3333333333333333, 1.1666666666666667, 1.8666666666666665, 2.6]
```

Esercizio

- Scrivere un programma che calcoli la soluzione di una equazione di secondo grado usando una funzione (partendo ad esempio da solv.py)

```
[redo@buchner functions]$ python secondord.py
insert a:1
insert b:2
insert c:3
a =  1.0  b =  2.0  c =  3.0
non ci sono soluzioni reali
[redo@buchner functions]$ python secondord.py
insert a:1
insert b:3
insert c:-4
a =  1.0  b =  3.0  c =  -4.0
soluzioni:  1.0 -4.0
```

Esercizio

- Scrivere una funzione che dati in input una lista di numeri, uno scalare ed un carattere esegua a seconda dal valore del carattere una delle quattro operazioni fodamentali +, -*, / (partite ad esempio da divide.py)

```
[redo@buchner functions]$ python operaz.py
[1.0, 3.5, 5.6, 7.8]
[0.5, 1.75, 2.8, 3.9]
[0.3333333333333333, 1.1666666666666667, 1.8666666666666665, 2.6]
[3.0, 5.5, 7.6, 9.8]
```

```
[redo@buchner functions]$ python operaz2.py
[1.0, 3.5, 5.6, 7.8]
[0.5, 1.75, 2.8, 3.9]
[0.3333333333333333, 1.1666666666666667, 1.8666666666666665, 2.6]
[3.0, 5.5, 7.6, 9.8]
```

```
def opera (vals, d = 2.0, c = "/"):
  res = []
  for v in vals:
    res.append(operazioni[c] (v, d))

  return res
```

Un dizionario
un pò speciale

Mutable or not

- I parametri in python si puo’ dire che sono passati **by assignment**
- Ricordatevi cosa succede quando assegniamo un valore ad una variabile, in realta’ stiamo semplicemente facendo puntare quella variabile ad una data locazione di memoria.
- Se l’oggetto che passo alla funzione e’ mutable potro’ quindi modificarlo
- Se l’oggetto non e’ mutable non potro’ modificarlo

Mutable le liste

```
def editlista (lista):
  lista.append("tre")

def assignlista (lista):
  lista = ["nuova", "lista"]

lista = ["uno", "due"]
editlista(lista)
print lista
assignlista(lista)
print lista


Lorianos-Air:functions redo$ python listemutable.py
['uno', 'due', 'tre']
['uno', 'due', 'tre']
```

Immutable

- Le stringhe in python non sono mutable, quindi cosa succede quando passo una stringa come parametro ad una funzione ?

```
def editstring (valin):
  valin = "output"
  print "modificata in: ", valin

val = "input"
editstring (val)
print "dopo la call: ", val

Lorianos-Air:functions redo$ python stringimmutable.py
modificata in:  output
dopo la call:  input
```

Quando all'interno della funzione scrivo valin = "output" sto creando un nuovo oggetto stringa che contiene il valore "output" e sto facendo puntare valin a questa nuovo indirizzo di memoria

Immutable 2

- Quindi come posso se mi serve agire per cambiare il valore delle stringa passata in input, semplice :

```
def editstring (valin):
  valin = "output"
  return valin

val = "input"
val = editstring (val)
print "dopo la call: ", val

Lorianos-Air:functions redo$ python stringimmutable2.py
dopo la call:  output
```

Analizziamo cosa succede ….

Input ed output da file

- L’istruzione open apre un file e ritorna un file object:

  f = open (“/path/nomefile”, ”r”)

  r = solo lettura

  w = file di testo per scrittura

  a = append mode

  rb= apre un file binario per la lettura

  wb = apre un file binario in modalita’ scrittura

  Posso poi usare il metodo close per chiudere il file: f.close()

I/O da file

```
Lorianos-MacBook-Air:iofiles redo$ cat iofileread.py
import sys

filename = ""

if len(sys.argv) != 2:
  print "usage: ", sys.argv[0], " filein"
  exit(1)
else:
  filename = sys.argv[1]

f = open (filename, "r")

l = f.readline()
print l
lines = f.readlines()
print lines

f.seek(0, 0) # offset primo parametro, whence 0 posizione assoluta,
             # 1 posizione relativa rispetto all poszione attuale
             # 2 poszione relativa alla fine del file

for l in f:
  print l

f.close()

Lorianos-MacBook-Air:iofiles redo$ python iofileread.py  filein
line1

['line2\n', 'line3\n', 'line4\n', 'line5\n', 'line6\n', '\n']
line1

line2
```

I/O file

```
import sys

filename = ""
if len(sys.argv) != 2:
  print "usage: ", sys.argv[0], " filein"
  exit(1)
else:
  filename = sys.argv[1]

f = open (filename, "w")

lista = ["linea1", "linea2", "linea3"]
for l in lista:
  f.write(l+"\n")

f.writelines(lista)
f.close()

Lorianos-MacBook-Air:iofiles redo$ python iofilewrite.py test ; cat test
linea1
linea2
linea3
linea1linea2linea3Lorianos-MacBook-Air:iofiles redo$
```

Esercizio

- Leggi n numeri da file calcola media e deviazione standard (suggerimento potrebbe servirvi il metodo split). Calcolo della deviazione standard vedi metodo Welford (anche in Donald Knuth’s Art of Computer Programming) e vedi numpy