

# Programmazione in Python

Funzioni: metodi e strutture dati

Dario Pescini - Mirko Cesarini <sup>1</sup>

Università degli Studi di Milano-Bicocca

Dipartimento di Statistica e Metodi Quantitativi

[dario.pescini@unimib.it](mailto:dario.pescini@unimib.it)

## Esercizio

Scrivere una funzione che conti tutte le occorrenze di una lettera in una stringa inserita dall'utente.

```
parola = raw_input('inserire una parola ')
# con la sintassi sotto la parola va racchiusa tra apici
# parola = input('inserire una parola racchiusa tra apici')

print parola

lettereCarattere = []
lettereConteggio = []
i = 0
while i < len(parola):
    if parola[i] in lettereCarattere:
        j = 0
        while j < len(lettereCarattere):
            if parola[i] == lettereCarattere[j]:
                lettereConteggio[j] += 1
            j += 1
    else:
        lettereCarattere.append(parola[i])
        lettereConteggio.append(1)
    i += 1

i = 0
while i < len(lettereCarattere):
    print lettereCarattere[i], ', ', lettereConteggio[i]
    i += 1
```

## Esercizio

Scrivere una funzione che conti tutte le occorrenze di una lettera in una stringa inserita dall'utente, sapendo che è possibile aggiungere un elemento al dizionario tramite l'istruzione:

```
dizionario[chiave] = valore
```

## Esercizio

Scrivere una funzione che conti tutte le occorrenze di una lettera in una stringa inserita dall'utente.

```
parola = raw_input('inserire una parola ')
# con la sintassi sotto la parola va racchiusa tra apici
# parola = input('inserire una parola racchiusa tra apici')

print parola

lettere = {}
i = 0
while i < len(parola):
    if parola[i] in lettere:
        lettere[parola[i]] += 1
    else:
        lettere[parola[i]] = 1
    i += 1

for lettera in lettere:
    print lettera, lettere[lettera]
```

## Esercizio - variante con metodo

```
parola = raw_input('inserire una parola ')
# con la sintassi sotto la parola va racchiusa tra apici
# parola = input('inserire una parola racchiusa tra apici')

print parola

lettere = {}

i = 0
while i < len(parola):
    if parola[i] not in lettere:
        lettere[parola[i]] = parola.count(parola[i])

i = 0
for carattere in lettere:
    print carattere, " ", lettere[carattere]
```

## Esercizio

Scrivere un programma che individui tutte le occorrenze di una lettera all'interno di una stringa entrambe inserite dall'utente

## Esercizio

```
def trova_lettera(word, substring):
    indice = word.find(substring)
    return indice

parola = raw_input('inserire una parola ')
lettera = raw_input('inserire la lettera da cercare ')

posizione = 0
indici = []
while posizione < len(parola):
    nuovo_indice = trova_lettera(parola[posizione:], lettera)
    if nuovo_indice != -1:
        indici += [nuovo_indice + posizione]
        posizione += nuovo_indice + len(lettera)
    else:
        posizione = len(parola)

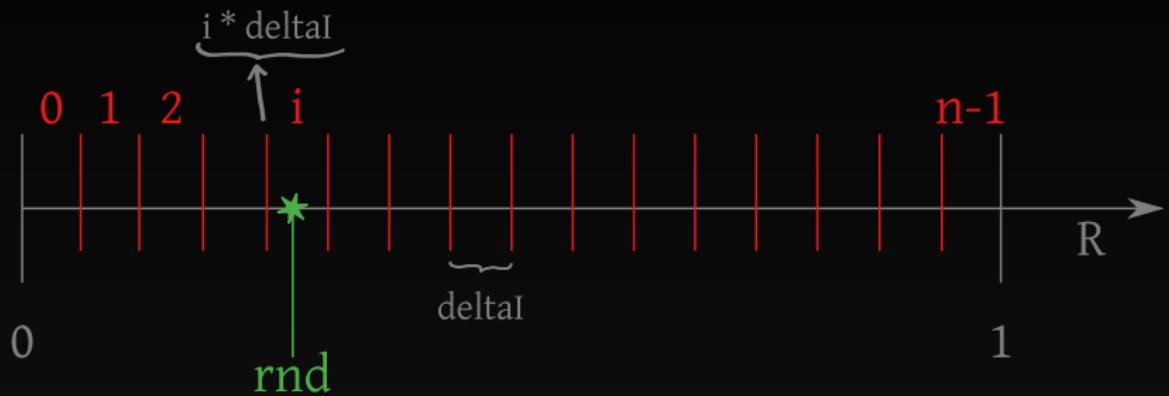
if indici != []:
    for indice in indici:
        print indice
else:
    print "lettera non trovata"
```

## Esercizio

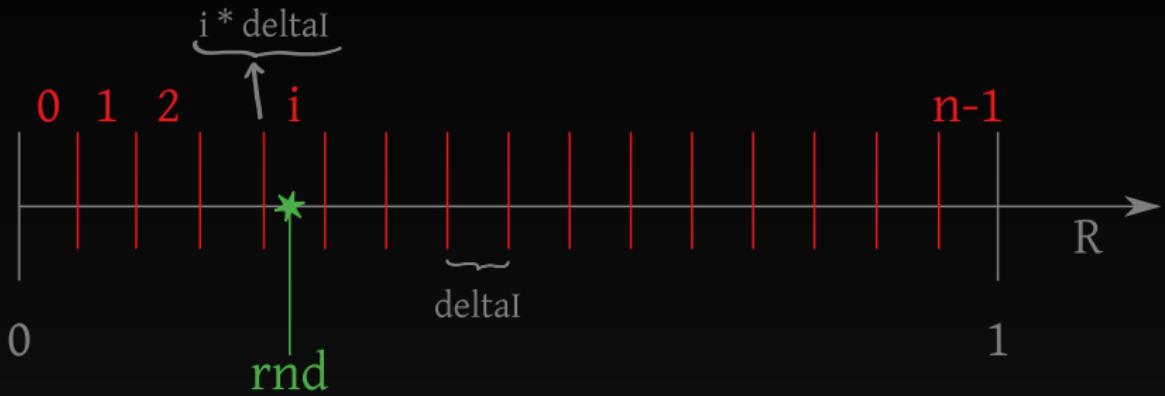
Scrivere un programma che

- riceva in input il numero di `lanci` da effettuare ed il numero di intervalli `numIntervalli` in cui dividere l'intervallo  $[0.0, 1.0]$
- effettui l'estrazione di `lanci` numeri casuali in  $[0.0, 1.0]$
- crei la distribuzione dei conteggi per i `numIntervalli`
- calcoli la relativa distribuzione di frequenza

## Esercizio

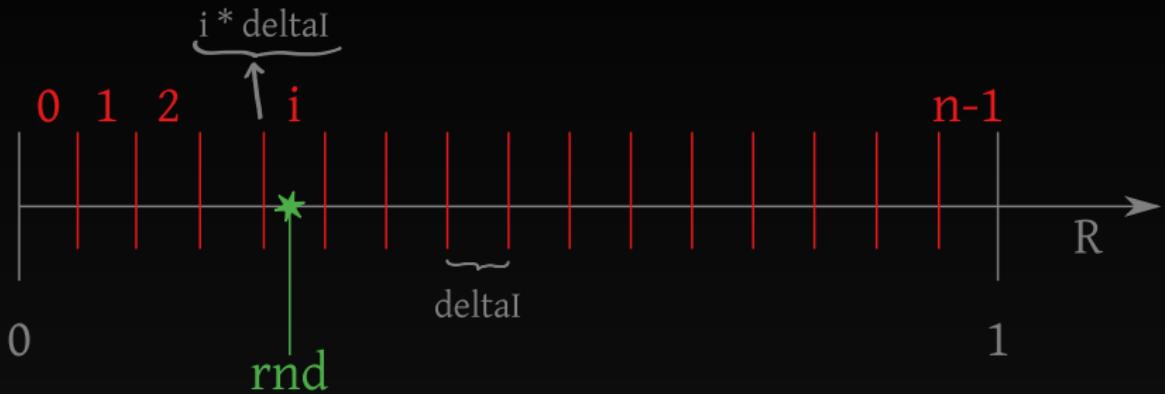


## Esercizio



$$\delta I = \frac{1}{n}$$

## Esercizio



$$\delta I = \frac{1}{n}$$

$i = \text{floor}(rnd / \delta I)$

## Esercizio - schema

```
carico input

intervalli = [] # preparo la lista per immagazzinare i conteggi

determino deltaI larghezza dell'intervallo

i = 0
while i < lanci:
    lancio il dado
    assegno il valore all'intervallo
    i += 1

frequenze = [] # preparo una lista per immagazzinare le
                frequenze

i = 0
while i < numIntervalli:
    calcolo le frequenze
    i += 1

stampo i risultati
```

## Esercizio

```
import random as rnd
import math as m

lanci = input('inserire il numero di lanci ')
numIntervalli = input('inserire il numero di intervalli in [0.0, 1.0] ')

serie = []
intervalli = []

deltaIntervallo = 1.0 / float(numIntervalli)
print "larghezza intervallo", deltaIntervallo

i = 0
while i < numIntervalli:
    intervalli.append(0.0)
    i += 1
```

## Esercizio - continua

```
i = 0
while i < lanci:
    numero = rnd.random()
    indiceIntervallo = int(m.floor(numero / deltaIntervallo))
    intervalli[indiceIntervallo] += 1
    i += 1

frequenze = list(intervalli)

i = 0
while i < numIntervalli:
    frequenze[i] /= lanci
    i += 1

# print serie
print "\nconteggi"
for intervallo in intervalli:
    print intervallo

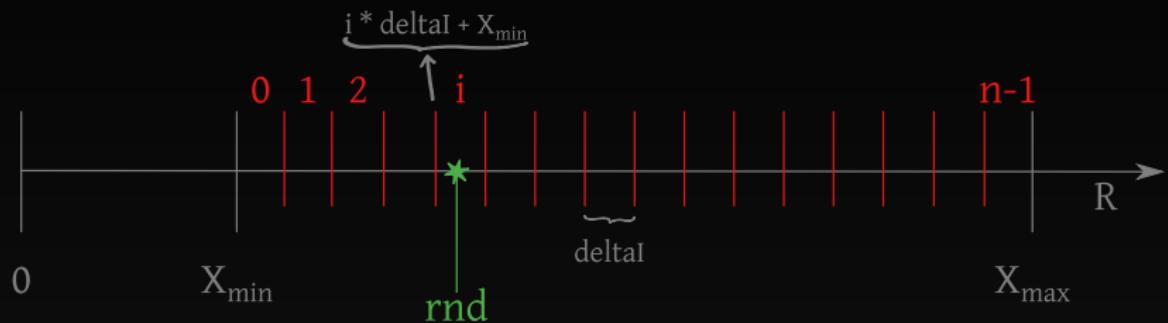
print "\nfrequenze"
for frequenza in frequenze:
    print frequenza
```

## Esercizio

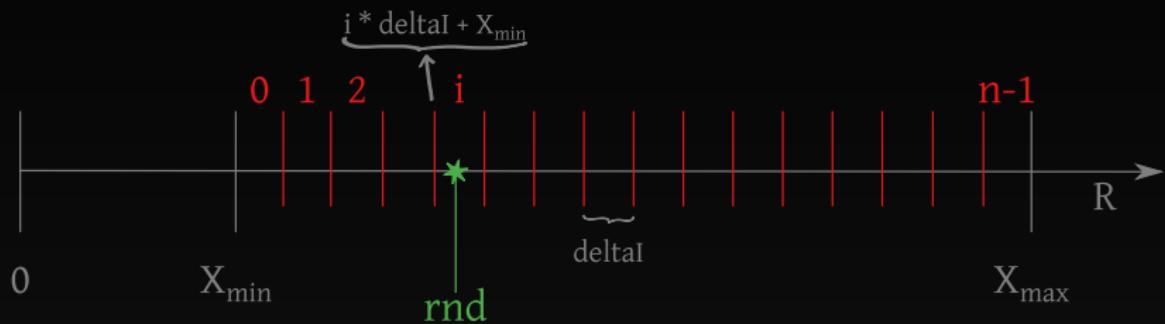
Scrivere un programma che

- riceva in input il numero di `lanci` da effettuare ed il numero di intervalli `numIntervalli` in cui dividere l'intervallo  $[x_{min}, x_{max}]$
- effettui l'estrazione di `lanci` numeri casuali in  $N(5,1)$
- crei la distribuzione dei conteggi per i `numIntervalli`
- calcoli la relativa distribuzione di frequenza

## Esercizio

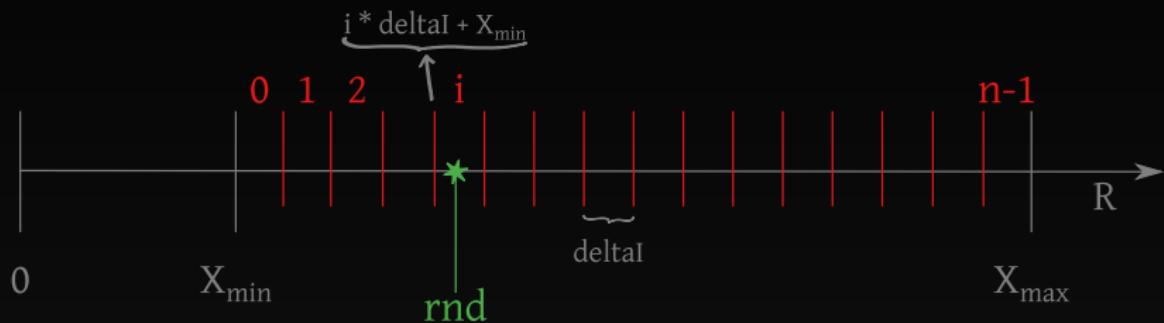


## Esercizio



$$\text{deltaI} = \frac{xMax - xMin}{n}$$

## Esercizio



$$\text{deltaI} = \frac{xMax - xMin}{n}$$

$i = \text{floor}((rnd - xMin) / \text{deltaI})$

## Esercizio - scheletro

- genera la lista dei lanci
- trova  $x_{\text{Min}}$  e  $x_{\text{Max}}$
- calcola  $\Delta \text{Intervallo}$
- prepara la lista dei conteggi
- riempi istogramma
- calcola frequenze

## Esercizio - genera la lista dei lanci

```
i = 0
while i < nLanci:
    numero = rnd.gauss(5, 1)
    serie.append(numero)
    i += 1
```

## Esercizio - trova xMin, xMax

```
min = float('inf')
max = -float('inf')
for el in lista:
    if (el < min):
        min = el
    if (el > max):
        max = el
```

## Esercizio - riempi istogramma

```
for el in lista:  
    indiceIntervallo = int(m.floor((el - numeroMinMax[0]) /  
        deltaIntervallo))  
    conteggi[indiceIntervallo] += 1
```

## Esercizio - soluzione

```
import random as rnd
import math as m

def estrazione(lista, nLanci):
    i = 0
    while i < nLanci:
        numero = rnd.gauss(5, 1)
        serie.append(numero)
        i += 1
    return lista

def minMax(lista):
    min = float('inf')
    max = -float('inf')
    for el in lista:
        if (el < min):
            min = el
        if (el > max):
            max = el
    return [min, max]
```

## Esercizio - soluzione

```
def istogramma(lista, conteggi, min, deltaInt):
    for el in lista:
        indiceIntervallo = int(m.floor((el - numeroMinMax[0]) / deltaIntervallo))
        conteggi[indiceIntervallo] += 1

lanci = input('inserire il numero di lanci ')
numIntervalli = input('inserire il numero di intervalli ')

conteggioIntervallo = []
i = 0
while i <= numIntervalli: # un intervallo in più per contenere MAX
    conteggioIntervallo.append(0.0)
    i += 1

serie = []
estrazione(serie, lanci)
```

## Esercizio - soluzione

```
numeroMinMax = minMax(serie)
print "valore minimo =", numeroMinMax[0], "valore massimo =", 
      numeroMinMax[1]
deltaIntervallo = (numeroMinMax[1] - numeroMinMax[0]) /
      float(numIntervalli)
print "larghezza intervallo", deltaIntervallo

istogramma(serie, conteggioIntervallo, numeroMinMax[0],
            deltaIntervallo)
del conteggioIntervallo[-1] # per eliminare il numero massimo

xIntervallo = []
frequenze = list(conteggioIntervallo)
i = 0
while i < numIntervalli:
    frequenze[i] /= lanci
    xIntervallo.append(numeroMinMax[0] + i * deltaIntervallo)
    i += 1
```

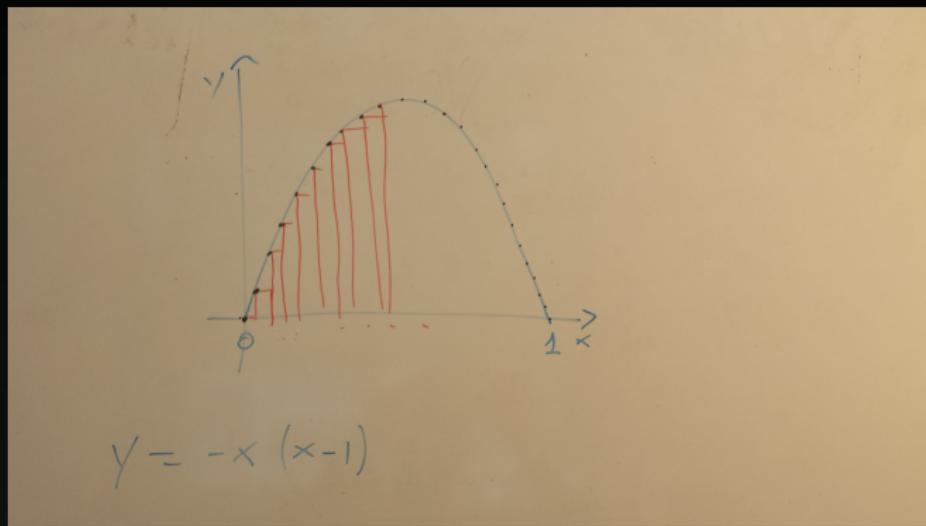
## Esercizio - soluzione

```
print "\nconteggi"
for intervallo in conteggioIntervallo:
    print intervallo

print "\nfrequenze"
for frequenza in frequenze:
    print frequenza

print "\nxIntervalli"
for x in xIntervallo:
    print x
```

## Esercizio



$$y = -x(x-1)$$

Si calcoli l'area sottesa dalla parabola in figura.

## Esercizio: decomporre il problema in problemi più semplici

- L'area sottesa può essere approssimata come la somma delle aree di rettangoli contigui aventi per base l'asse delle ascisse e per altezza il punto di contatto con la parabola
- costruire i rettangoli
- determinare base e altezza di un rettangolo
- calcolare area di un rettangolo
- sommare i contributi

## Esercizio: decomporre il problema in problemi più semplici

- L'area sottesa può essere approssimata come la somma delle aree di rettangoli contigui aventi per base l'asse delle ascisse e per altezza il punto di contatto con la parabola
- costruire i rettangoli
- determinare base e altezza di un rettangolo
- calcolare area di un rettangolo
- sommare i contributi

## Esercizio: decomporre il problema in problemi più semplici

- per costruire la base dei rettangoli si può suddividere l'intervallo  $I = [0, 1]$  in un numero finito,  $M$ , di intervalli contigui  $I_i$ .
- in questo caso ciascun intervallo  $I_i$  avrà lunghezza pari a  $\frac{1}{M}$  (base del rettangolo)
- ciascun rettangolo ( $i$ ) avrà per base il segmento  $[\frac{i}{M}, \frac{i+1}{M}]$
- l'altezza potrebbe essere  $y_i = -x_i(x_i - 1) = -\frac{i}{M}(\frac{i}{M} - 1)$

## Esercizio: soluzione

```
def effe(x):
    y = -x * (x - 1.0)
    return y

numIntervalli = input('inserire il numero di intervalli in [0.0, 1.0] ')

deltaIntervallo = 1.0 / float(numIntervalli)
print "larghezza intervallo", deltaIntervallo

xIntervalli = []
yIntervalli = []
i = 0
while i < numIntervalli:
    xIntervallo = i*deltaIntervallo
    xIntervalli.append(xIntervallo)
    yIntervalli.append(effe(xIntervallo))
    i += 1

areaSottesa = 0.0
for altezza in yIntervalli:
    areaSottesa += altezza * deltaIntervallo

print "l'area sottesa dalla curva vale ", areaSottesa
```

## Esercizio: Bisezione

Sapendo che:

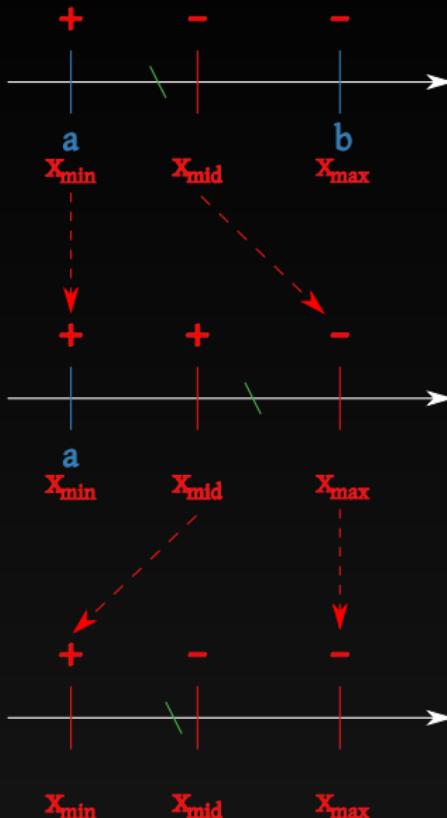
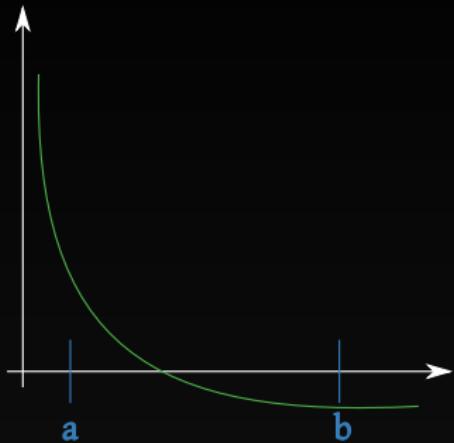
- la funzione  $f: \mathbb{R} \rightarrow \mathbb{R}$  è continua nell'intervallo  $I = [a, b]$
- $f(a), f(b)$  hanno segno opposto

Si può mostrare che la funzione  $f$  ha almeno una radice in  $I$ ,  
i.e. esiste almeno una  $x_0$  tale che  $f(x_0) = 0$ .

Numericamente, si può stimare il valore di  $x_0$  tramite il metodo di bisezione.

## Esercizio: Bisezione

$$x_{mid} = \frac{x_{min} + x_{max}}{2}$$



## Esercizio: Bisezione

Si scriva un programma che implementi il metodo della bisezione per trovare la radice della funzione:

$$f(x) = x^3 - 13x^2 + 19x + 33$$

nell'intervallo  $I = [0, 9]$

Si suggerisce di fissare in anticipo il numero di iterazioni per l'arresto del programma.

## Esercizio: Bisezione

```
ITERAZIONI = 100

def effe(x):
    y = x**3 - 13 * x**2 + 19 * x + 33
    return y

# listaPunti[0] = Xmin
# listaPunti[1] = Xmid
# listaPunti[2] = Xmax
def intervallo(listaPunti):
    if(effe(listaPunti[0]) * effe(listaPunti[1]) <= 0):
        listaPunti[2] = listaPunti[1] # Xmax = Xmid
    else:
        listaPunti[0] = listaPunti[1] # Xmin = Xmid
    listaPunti[1] = 0.5 * (listaPunti[0] + listaPunti[2])
```

## Esercizio: Bisezione

```
a = 0.0
b = 9.0

listaX = [a, 0.5 * (a + b), b]
print listaX
i = 0
while (i < ITERAZIONI):
    intervallo(listaX)
    print listaX
    i = i + 1
```

## Esercizio - Percorso

Data una sequenza di città,  
calcolare la lunghezza del  
percorso che le unisce.

**Input** Coordinate delle città

**Output** lunghezza percorso

- leggere da terminale le coordinate
- calcolare le distanze tra le città
- sommare le distanze
- stampa a video la lunghezza del percorso

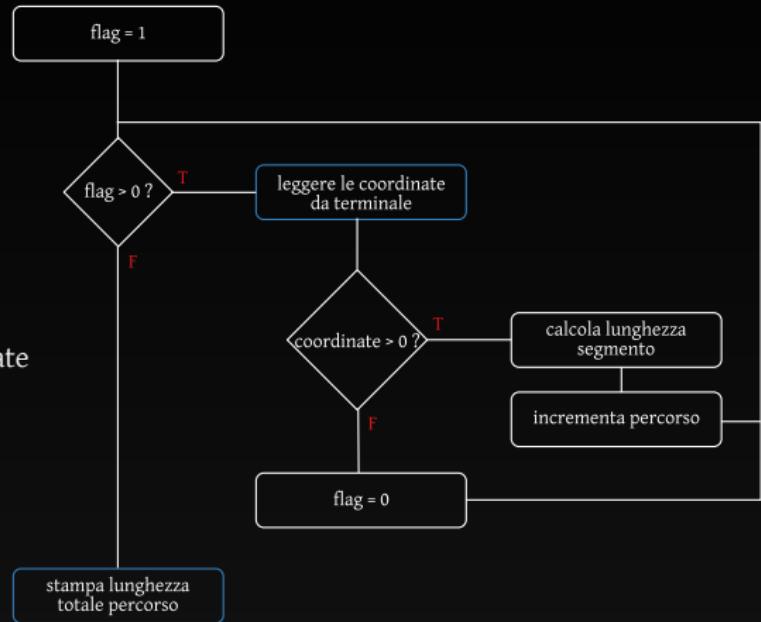
# Esercizio - Percorso

Data una sequenza di città,  
calcolare la lunghezza del  
percorso che le unisce.

**Input** Coordinate delle città

**Output** lunghezza percorso

- leggere da terminale le coordinate
- calcolare le distanze tra le città
- sommare le distanze
- stampa a video la lunghezza del percorso



## Esercizio - Percorso

```
import math

haltFlag = 1
lunghezza = 0
print "inserire le coordinate delle citta':"
print "un valore negativo termina l'inserimento"
xOld = input("ascissa ")
yOld = input("ordinata ")
while (haltFlag > 0):
    x = input("ascissa ")
    y = input("ordinata ")
    if(x < 0 or y < 0):
        haltFlag = 0
    else :
        deltaX = x - xOld
        deltaY = y - yOld
        xOld = x
        yOld = y
        lunghezza += (deltaX**2 + deltaY**2)
lunghezza = math.sqrt(lunghezza)
print "la lunghezza del percorso e' ", lunghezza
```

## Esercizio - Percorso

```
import math

haltFlag = 1
lunghezza = 0
print "inserire le coordinate delle citta':"
print "un valore negativo termina l'inserimento"
xOld = input("ascissa ")
yOld = input("ordinata ")
while (haltFlag > 0):
    x = input("ascissa ")
    y = input("ordinata ")
    if(x < 0 or y < 0):
        haltFlag = 0
    else :
        deltaX = x - xOld
        deltaY = y - yOld
        xOld = x
        yOld = y
        lunghezza += (deltaX**2 + deltaY**2)
lunghezza = math.sqrt(lunghezza)
print "la lunghezza del percorso e' ", lunghezza
```