Python (strutture dati)

Loriano Storchi

loriano@storchi.org

http::/www.storchi.org/

I numeri complessi

- I numeri complessi sono intrinsecamente definiti ed usabili in python

```
a = 2 + 3j
b = 4 - 1j

print a
print a.real, " ", a.imag
print b
print b.real, " ", b.imag

c = a * b
print type(c), " valore ", c

if type(c) == complex:
    print "c e\' un numero complesso"

[redo@banquo datastr (master)]$ python complex.py
(2+3j)
2.0   3.0
(4-1j)
4.0   -1.0
<type 'complex'>  valore  (11+10j)
c e' un numero complesso
[redo@banquo datastr (master)]$
```

Strutture dati

- Le strutture dati permettono di organizzare i dati in modo da rendere il loro uso e la loro manipolazione piu' efficiente
- Vedremo in particolare le stringhe, le liste, le tuple, i dizionari ed i set.
- Noi vedremo solo le basi minime di uso utili allo svolgimento di esercizi base

Le stringhe

- Una stringa e' una sequenza di caratteri, in python ci sono diversi metodi/operazioni utili alla manipolazione delle stringhe, una stringa in generale puo' essere vista come un array di caratteri

```
str1 = "hello"
str2 = "world"

print str1 + " " + str2
print 3*(str1+" ")
print str1[0]
print str1[0:3]
print str1[-2:]
[redo@banquo datastr (master)]$ python strin1.py
hello world
hello hello hello
h
hel
lo
[redo@banquo datastr (master)]$
```

Le stringhe

- Come detto la classe in questione ha diversi metodi vediamo solo alcuni

```
[redo@banquo datastr (master)]$ cat strin2.py
str = "Hello, World!"
index = str.find("ll")
if index >= 0:
  print "a ", index, " trovato ", str[index]

res = str.split(" ")
idx = 0
for r in res:
    idx += 1
    print idx , " - ", r

for i in range(0,len(str)):
    print str[i]
[redo@banquo datastr (master)]$ python strin2.py
a  2  trovato  l
1  -  Hello,
2  -  World!
H
e
l
```

Liste

- Le liste sono sequenze ordinate di oggetti, molte operazioni di base sono in comune con le stringhe

```
a = [1, "pippo", 4.5, "pluto"]
print a[0]
for i in range(0,len(a)):
    print a[i]

for l in a:
    print l
[redo@banquo datastr (master)]$ python liste1.py
1
1
pippo
4.5
pluto
1
pippo
4.5
pluto
[redo@banquo datastr (master)]$
```

Liste

- La lista ha molti metodi utili vediamone alcuni

```
[redo@banquo datastr (master)]$ cat liste2.py
a = [1, 3.5, -6.0, 5]
a.append(46)
print a
print "rimuove l\'ultimo elemento: ", a.pop()
a.append(46)
i = len(a) - 1
print "rimuove l\'elemento ", i, " ", a.pop(i)
a.sort()
print a
[redo@banquo datastr (master)]$ python liste2.py
[1, 3.5, -6.0, 5, 46]
rimuove l'ultimo elemento:  46
rimuove l'elemento  4   46
[-6.0, 1, 3.5, 5]
[redo@banquo datastr (master)]$
```

Tuples

- Le tuples in python sono molti simili alle liste solo che la loro manipolazione e' piu' rapida visto che sono “immutabili”

```
t = (1,3.5,8,10.0)
for i in range(len(t)):
    print t[i]

for val in t:
    print val

# ma posso modificare un valore ?
t[1] = 0
[redo@banquo datastr (master)]$ python tuples.py
1
3.5
8
10.0
1
3.5
8
10.0
Traceback (most recent call last):
  File "tuples.py", line 9, in <module>
    t[1] = 0
TypeError: 'tuple' object does not support item assignment
[redo@banquo datastr (master)]$
```

Piccola precisazione

```
[redo@banquo datastr (master)]$ cat tuples_pres.py
t = (1,3.5,8,10.0,[10,5])

for val in t:
    print val

# ma posso modificare un valore se mutabile
t[4][1] = 0
print t
[redo@banquo datastr (master)]$ python tuples_pres.py
1
3.5
8
10.0
[10, 5]
(1, 3.5, 8, 10.0, [10, 0])
[redo@banquo datastr (master)]$
```

Dizionari

- Un dizionario e' una sequenza di elementi , ogni elemento a' una coppia chiave, valore. Le chiavi sono uniche ed i dizionari si creano usando le parentesi graffe

```
d = {"k1":1, "k2":"valore", 3:"val3"}
print d[3], d["k1"]
d["quattro"] = 4
print d
if d.has_key("quattro"):
    print "chiave presente"
if not d.has_key(4):
    print "chiave non presente"
[redo@banquo datastr (master)]$ python diz1.py
val3 1
{'k2': 'valore', 'k1': 1, 3: 'val3', 'quattro': 4}
chiave presente
chiave non presente
[redo@banquo datastr (master)]$
```

Dizionari

```
d = {"k1":1, "k2":"valore", 3:"val3", "quattro":4}
for x in d.itervalues():
    print x
print " "
for x in d.iterkeys():
    print x
print " "
for x in d.iteritems():
    print x
    print x[0], x[1]
[redo@banquo datastr (master)]$ python diz2.py
valore
1
val3
4

k2
k1
3
quattro

('k2', 'valore')
k2 valore
('k1', 1)
k1 1
(3, 'val3')
```

Dizionari

```
d = {"k1":1, "k2":"valore", 3:"val3", "quattro":4}
del d["k1"]
for x in d.iteritems():
    print x
d.clear()
print len(d)
for x in d.iteritems():
    print x
[redo@banquo datastr (master)]$ python diz3.py
('k2', 'valore')
(3, 'val3')
('quattro', 4)
0
```

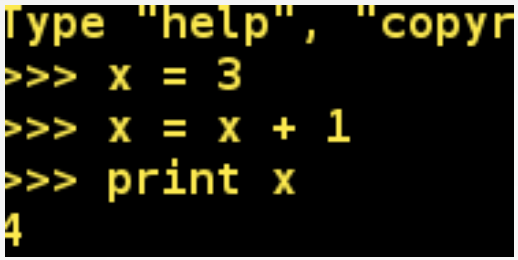
Reference

- Quando dichiaro una variabile sto chiedendo una certa quantita' di spazio in memoria
- In python l'operazione di assegnamento manipola i riferimenti , quindi x = y non crea una copia dei dati contenuti in y in x, ma crea un riferimento a y, x punta ad y

```
a = [1,2,3,4]
b = a
a.append(5)
print b
x = 3
y = x
x = 4
print y
[redo@banquo datastr (master)]$ python refer.py
[1, 2, 3, 4, 5]
3
```

Quando scrivo x = 4 viene allocato lo spazio in memoria necessario a contenere l'intero 4 e poi viene “memorizzato l'indirizzo della” (creato il riferimento alla) locazione di memoria in x

Reference

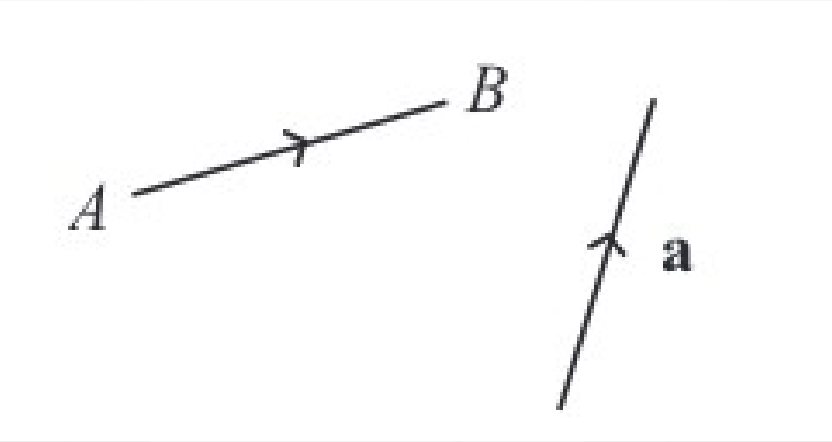


- Cosa succede veramente quando incremento x ?
  - L'interprete recupera il valore contenuto nell'indirizzo di memoria a cui x fa riferimento
  - Viene calcolato il risultato dell'operazione 3 + 1 ed il risultato viene memorizzato in una nuova locazione di memoria
  - Si cambia il riferimento in x, x adesso fara' riferimento al nuovo indirizzo in memoria dove e' memorizzato il valore 4
  - Python ha un garbage collector che elimina libera tutta la memoria allocata quando non ci sono piu' nomi che fanno riferimento alle zone di memoria in questione

VETTORI

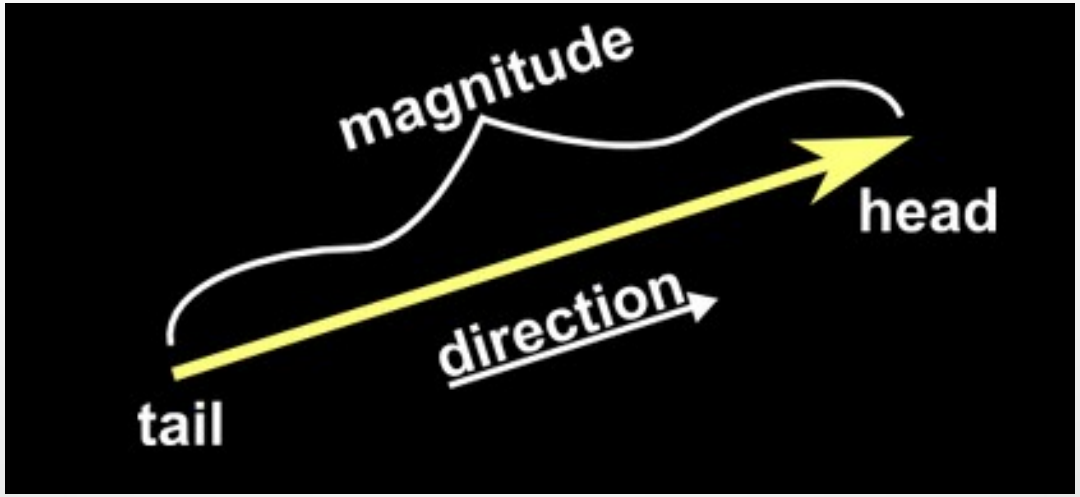
Vettori

- I vettori sono estremamente utili in fisica. La caratteristica principale di un vettore è che ha sia una grandezza (o una dimensione) che una direzione.
- Un esempio di una grandezza vettoriale è velocità. Questa è la velocità, in una particolare direzione.



Vettori

- I vettori sono estremamente utili in fisica. La caratteristica principale di un vettore è che ha sia una grandezza (o una dimensione) che una direzione.
- Un esempio di una grandezza vettoriale è velocità. Questa è la velocità, in una particolare direzione.



Vettori

- I vettori sono estremamente utili in fisica. La caratteristica principale di un vettore è che ha sia una grandezza (o una dimensione) che una direzione.
- Un esempio di una grandezza vettoriale è velocità. Questa è la velocità, in una particolare direzione.

Vettori

- TODO come rappresento un vettore ? Vedi una lista e poi magari rappresenta anche con matplotlib poi somman sottrazione prodotto scalare e vettoriale
-

MATRICI

Una matrice

- Posso usare una lista di liste per memorizzare una matrice in modo semplice

```
import random
import math

A =  [[0.0, 0.0, 0.0],
      [0.0, 0.0, 0.0],
      [0.0, 0.0, 0.0]]

for i in range(len(A)):
  for j in range(len(A[0])):
    A[i][j] = random.uniform(0.0, 1.0)

print "Matrix A"
print A
[redo@banquo mtxmtx (master)]$ python storemtx.py
Matrix A
[[0.1975653801959704, 0.4483683937014895, 0.278
 0.9080131010804726, 0.3679889526650569], [0.471
.5738920118128268]]
```

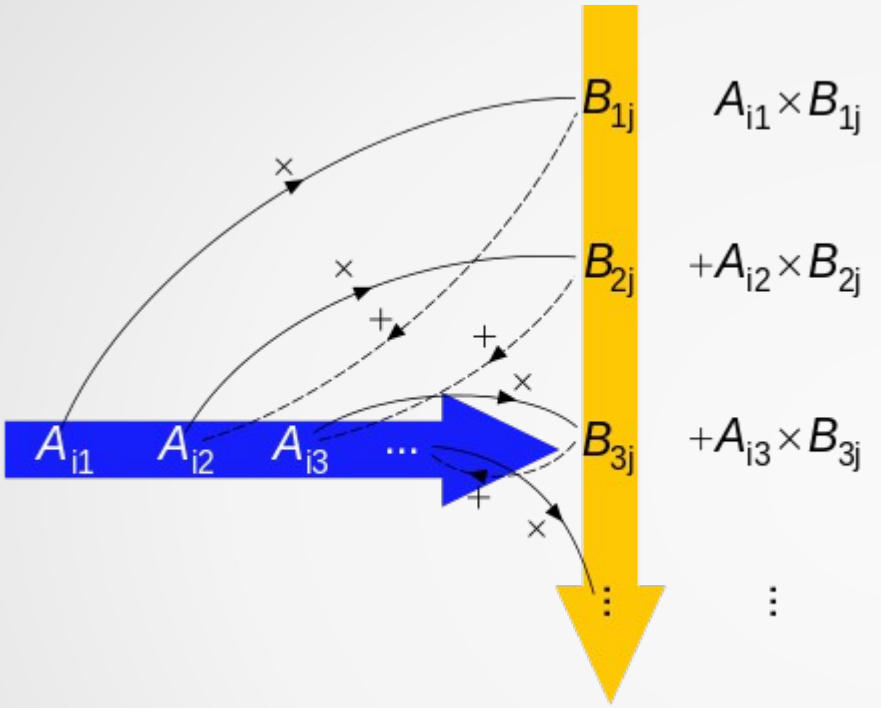
Moltiplicazione matrice matrice

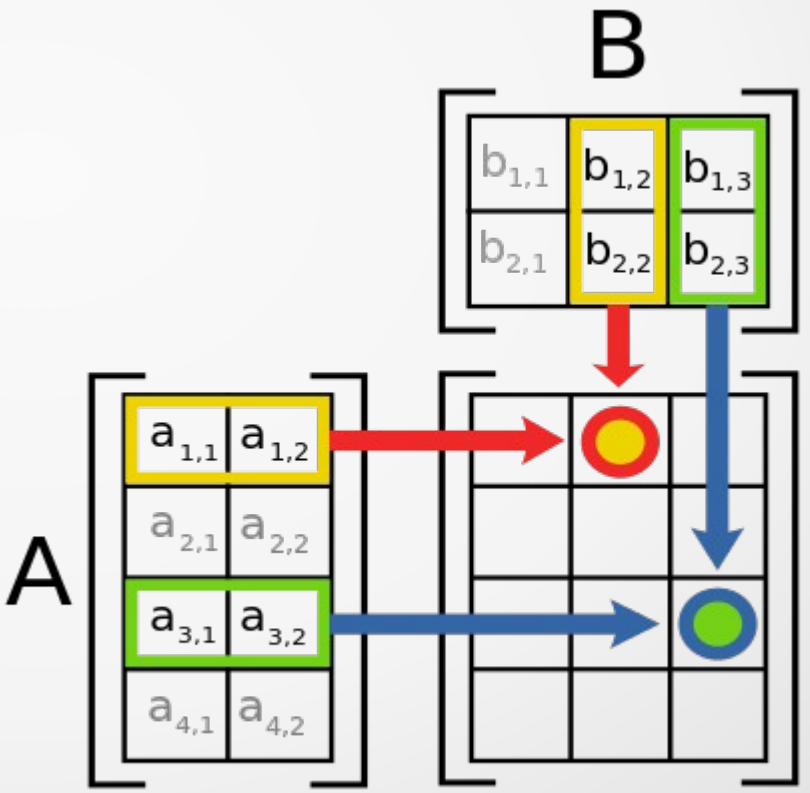
$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1p} \\ B_{21} & B_{22} & \cdots & B_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m1} & B_{m2} & \cdots & B_{mp} \end{pmatrix}$$

$$(\mathbf{AB})_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj} \ .$$

Il prodotto e' definito solo per matrici con dimensioni compatibili

Moltiplicazione matrice matrice





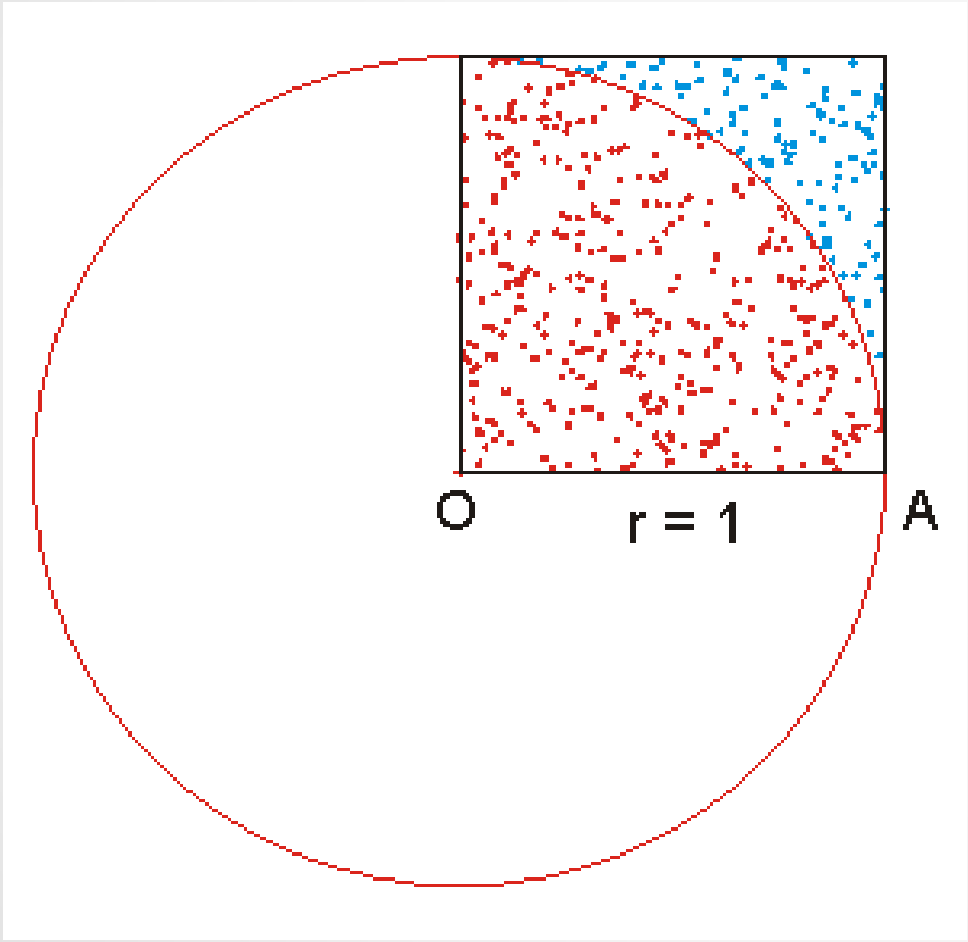
Esercizio

- Scrivere un programma che date due matrici 3x3 riempite con numeri random calcoli e stampi il risultato del prodotto matrice matrice

```
[redo@banquo mtxmtx (master)]$ python mtx.py
Matrix A
[0.1924696838424818, 0.969694776511462, 0.8237325914685499]
[0.177986003994455, 0.975535311913037, 0.8217085413339825]
[0.4127948602822054, 0.05779395844699264, 0.4020898243848140]
Matrix B
[0.7196544280415835, 0.8235257700392403, 0.9349263737223458]
[0.2739352426172888, 0.7093719818717363, 0.4407551720290670]
[0.1260881966959214, 0.939121875851728, 0.4639718068159948]
Matrix C
[0.5080081911273185, 1.619963346520346, 0.9895316704002202]
[0.4989296664411049, 1.610277945334311, 0.9776256401093906]
[0.3636002319703472, 0.7585559701630582, 0.5979641302345579]
```

ESERCIZI ED ESEMPI

Calcolo PI con metodo MC



$$A_S = (2r)^2 = 4r^2$$

$$A_C = \pi r^2$$

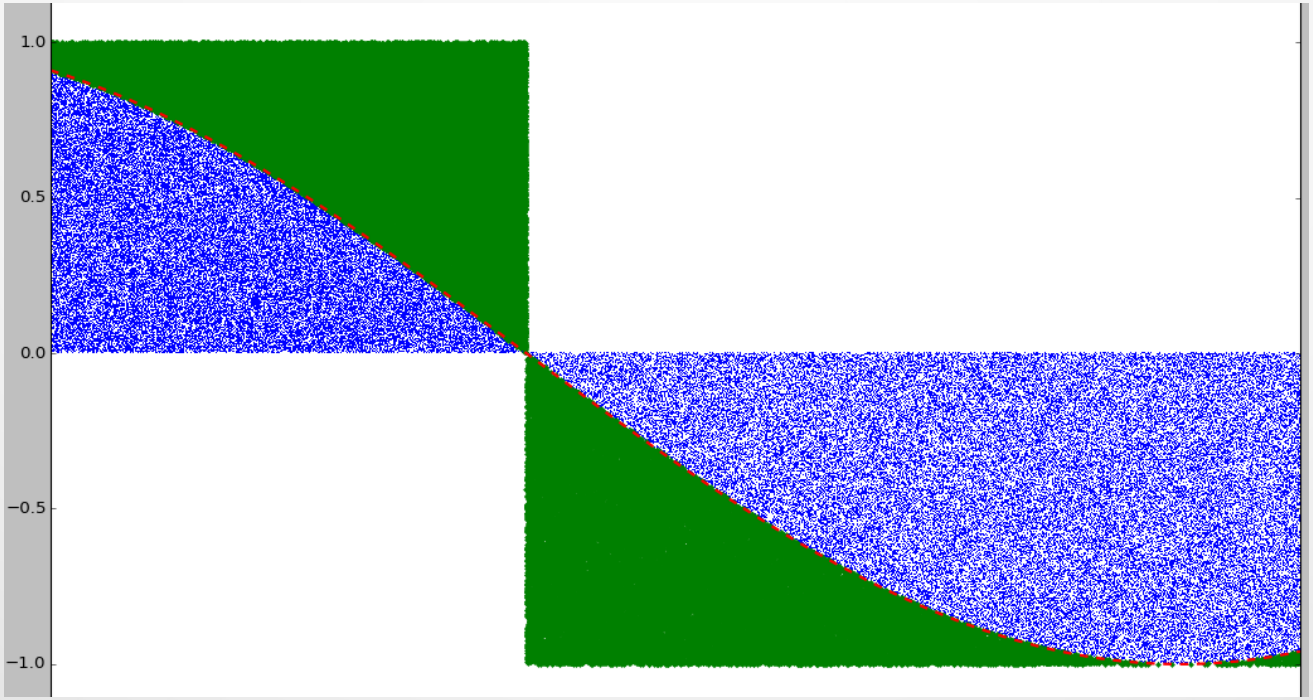
$$\pi = 4 \times \frac{A_C}{A_S}$$

```
import random
import math
import sys

DIM = 100000

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

circle_count = 0

for i in range(0,DIM):

   x = random.uniform(0.0, 1.0)
   y = random.uniform(0.0, 1.0)

   if (math.sqrt((math.pow(x, 2.0) + math.pow(y, 2.0))) < 1.0):
     circle_count = circle_count + 1

pi = float(circle_count) / float(DIM)

print(4.0 * pi)
```

Esercizio

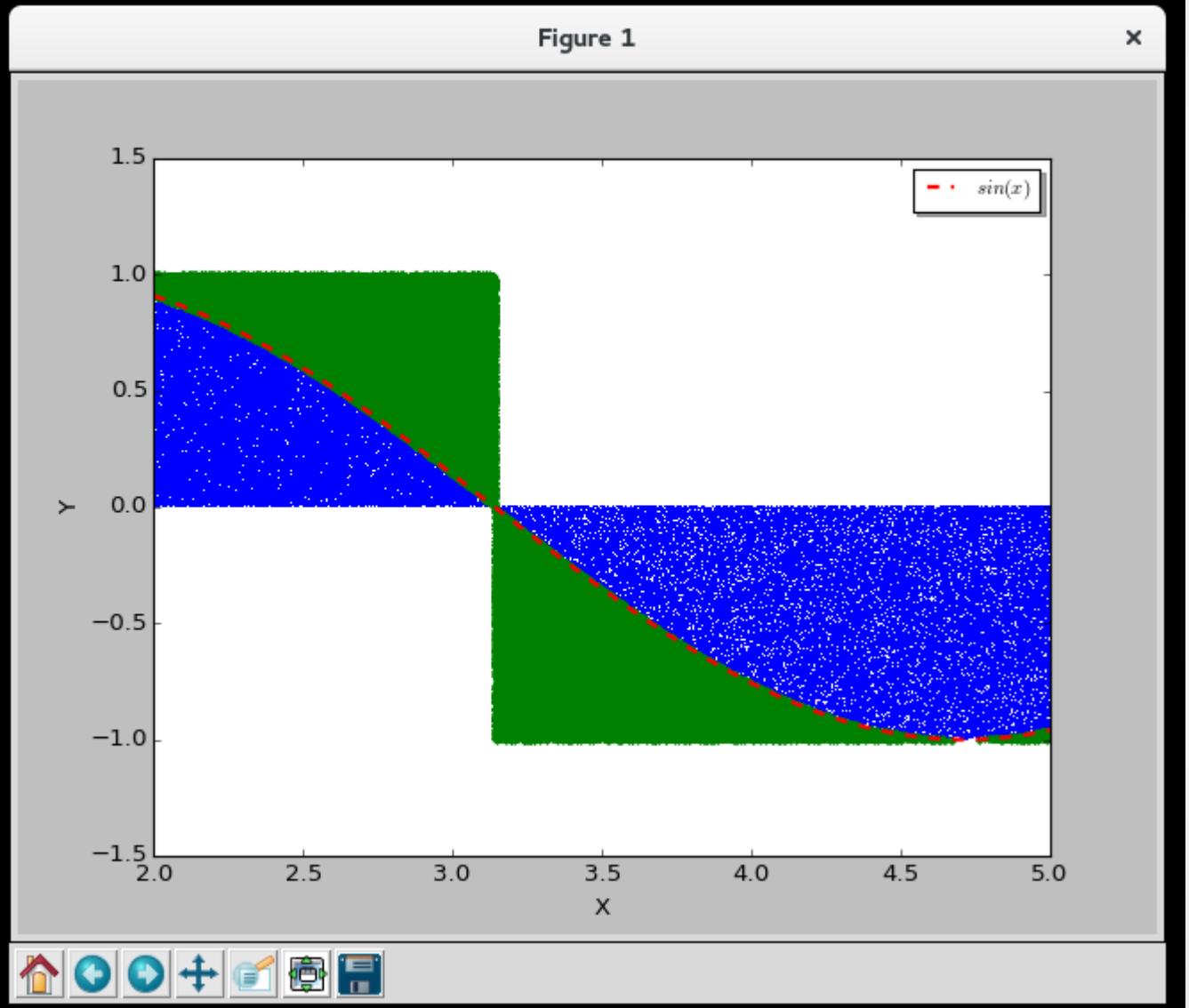
- Calcolo di integrale con metodo MC:

$$\int_2^5 \sin(x)\,dx = \cos(2) - \cos(5) = -0{,}69981$$



```
[redo@banquo mcsin (master)]$ time python mcsin.py 100
-0.18

real    0m0.013s
user    0m0.010s
sys     0m0.003s
[redo@banquo mcsin (master)]$ time python mcsin.py 1000
-0.726

real    0m0.017s
user    0m0.014s
sys     0m0.003s
[redo@banquo mcsin (master)]$ time python mcsin.py 10000
-0.6834

real    0m0.049s
user    0m0.040s
sys     0m0.008s
[redo@banquo mcsin (master)]$ time python mcsin.py 100000
-0.70938

real    0m0.115s
user    0m0.108s
sys     0m0.007s
[redo@banquo mcsin (master)]$ time python mcsin.py 1000000
-0.703062

real    0m1.049s
user    0m1.021s
sys     0m0.027s
[redo@banquo mcsin (master)]$ time python mcsin.py 10000000
-0.7001124

real    0m10.226s
user    0m10.105s
sys     0m0.116s
[redo@banquo mcsin (master)]$
```

```
[redo@banquo mcsin (master)]$ python mcsin_wplt.py 100000
-0.696206603137
```



Vediamo un primo esempio di uso di numpy e matplotlib