RANDOM NUMBERS

- *pseudo* random
- uniformity and correlation
- repeatable (*seed)*

Try:

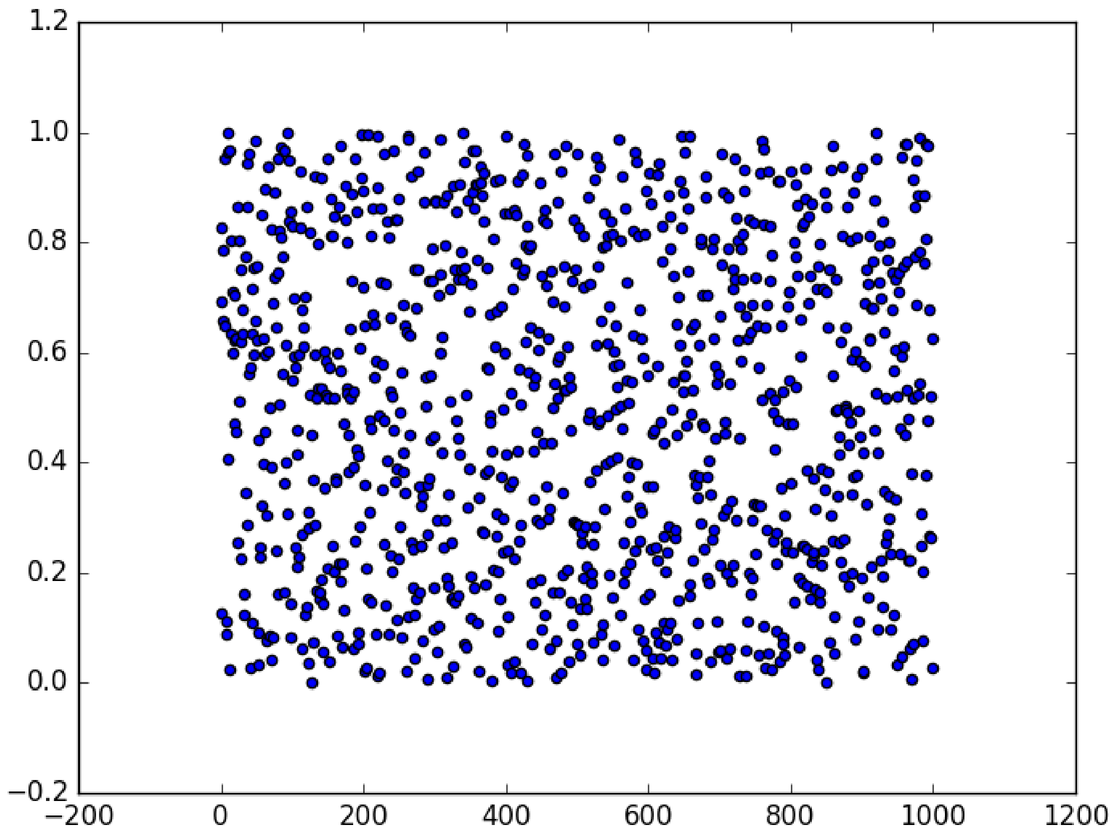
```
import random
import pylab as plt
import numpy as np

random.seed(-1001)
random.uniform(0,1)
random.uniform(0,1)
random.seed(-1003)
random.uniform(0,1)
random.uniform(0,1)
```

Exercize: plot a distribution of 1000 uniform random numbers in [0,1)

rn=np.random.uniform(0,1,1000)

plt.scatter(range(1000), rn)



Exercize: plot a distribution of 1000 random number with Gaussian distribution (hint: use random.normal)

Exercize: make a histogram of the two distributions (using plt.hist)

..look at the available distributions using random?

MAKING A SPHERE

Let's plot a 2D uniform distribution of 10,000 points with center (Xc,Yc) and size L; for example, (10,10) and L=50

N=10000

R=50

XC=10

YC=10

rnx=np.random.uniform(0,R,N)-R/2+XC

rny=np.random.uniform(0,R,N)-R/2+YC

plt.scatter(rnx,rny)

plt.show()

How would you extract a circle from this? Hint: use np.where()

For instance:

x=np.random.uniform(0,1,1000)

i,=np.where(x<0.5)

plt.scatter(range(len(x[i])),x[i])

plt.show()

np.where() is a *very powerful* function! Not that, for giving more conditions, the syntax is the following:

j,=np.where((x>0.2) & (x<0.5))

with the *binary* operators (&, | …)

Problem: in this way, a number of "attempts" are wasted.

Solution: use spherical coordinates and sample r, phi:

r=np.random.uniform(0,1,1000)

phi=np.random.uniform(0,6.28,1000)

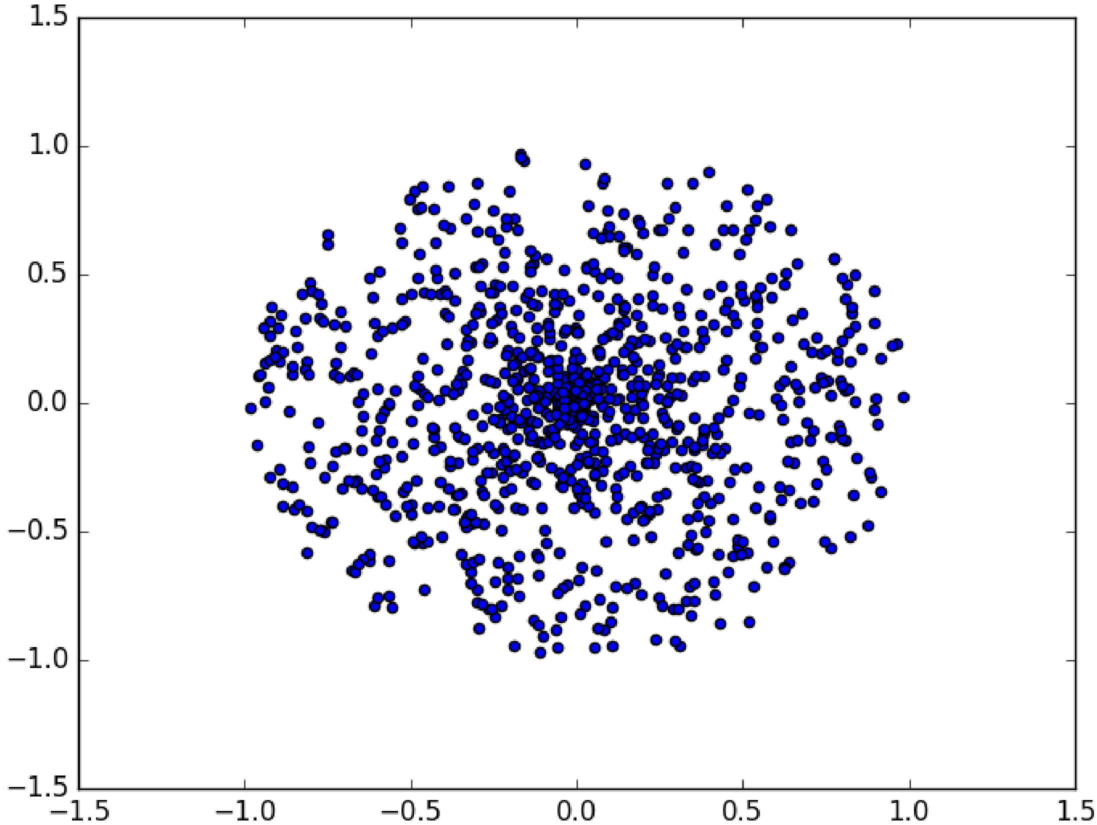
rnx=r*np.cos(phi)

rny=r*np.sin(phi)

but....

plt.scatter(rnx,rny)

plt.show()



and this is *NOT* uniform!

The reason is that in Cartesian coordinates the volume of the circle is

$$\iint_0^{R/2} dx\, dy$$

in polar coordinates is

$$\int_0^{R/2} \int_0^{2\pi} r\, dr\, d\varphi$$

thus, in Cartesian coordinates x,y must be uniform; in polar, rdr, d φ !

To sample a *non*-uniform distribution from a uniform one, one must equate their PDF;

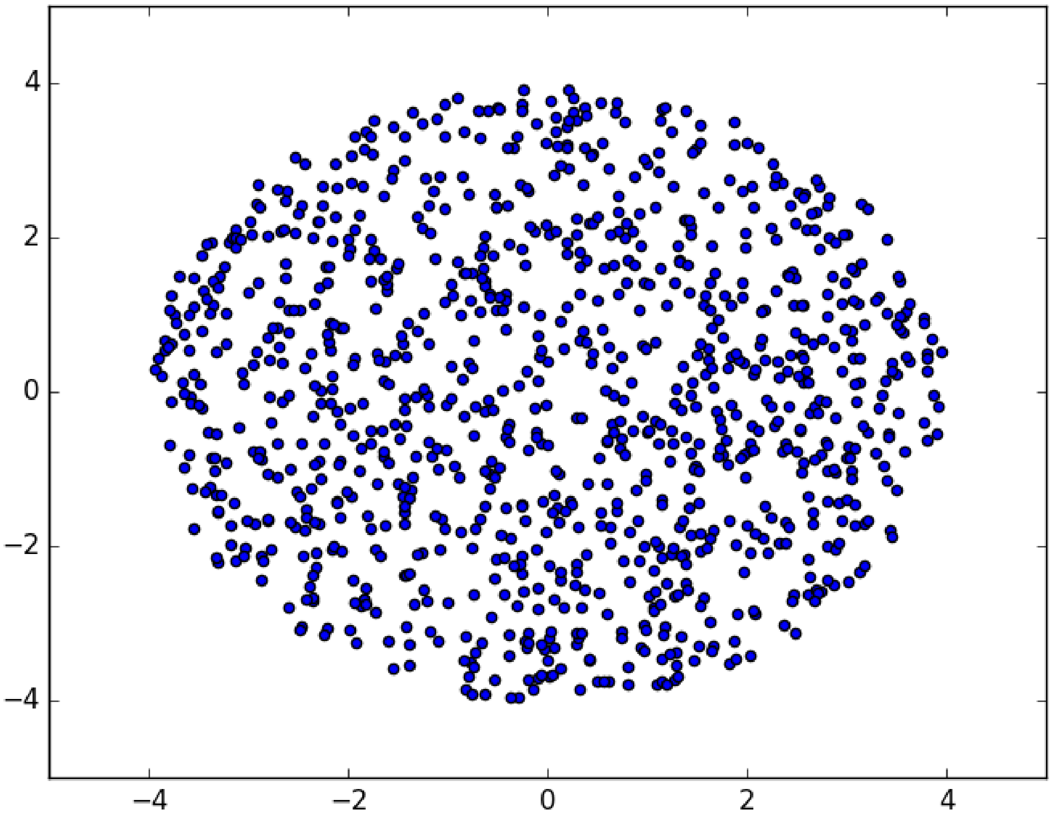
if U(u) is the uniform distribution and F(r) the needed one, we need F(r)=U(u)

where r is our radius and u our uniform number. Thus:

$$\int r\, dr = \int du \qquad 1/2 r^2 = u \qquad r = \sqrt{2u}$$

```
R=4
r=np.sqrt(2.*np.random.uniform(0,R*R/2.,1000))
phi=np.random.uniform(0,6.28,1000)
rnx=r*np.cos(phi)
rny=r*np.sin(phi)
plt.scatter(rnx,rny)
plt.show()
```

..note the extremes...



Exercize: knowing that the volume of a sphere (3D) is

$$\int_0^R dr \int_0^{2\pi} d\varphi \int_{-1}^{1} d\cos\vartheta$$

produce a distribution of points uniformly distributed in the sphere and plot the three projection, with 1000, 10000 and 100000 points.

Please, do that using a function, to generate x,y,z ; it will be needed in the next section.

def sfera(R,N,Xc,Yc,Zc):

r= ….

phi=...

….

x= ...

```
        return x,y,z
R=20
Xc=0.
Yc=0.
Zc=0.
N=1000
x1,y1,z1=sfera(R,N,Xc,Yc,Zc)
...
```

ROUNDOFF ERRORS!

Generate two spheres, with N=1000000 points, centers of masses first in (1000,0,0) and (-1000,0,0), then in (1e20,0,0) and (-1e20,0,0)

Compute the center of mass of the two spheres *using a for cycle*, i.e.:

```
xcm=0.
ycm=0.
zcm=0.
for i in range(N):
    xcm += x1[i]
    ycm += y1[i]
    zcm += z1[i]
for i in range(N):
    xcm += x2[i]
    ycm += y2[i]
```

```
    zcm += z2[i]
print 'CM (1): ',xcm/n, ycm/n, zcm/n
```

where the coordinates of points of the spheres are x1,y1,z1 and x2,y2,z2 and n=2N

...try to *exchange the summations* (first x2, then x1)

...what happens if you compute the center of mass using np.sum()?