COURS N°2 PYTHON

L2 MATHÉMATIQUES

Mathématiques et Python

Vue pour impression

Auteur : Antoine Laurent

LE MODULE RANDOM EN PYTHON

- Ce module donne accès aux nombres aléatoires et permet de opérations stochasitques
- Quelques exemples ...

```
>>> import random
>>> help(random)
>>> random.random()
0.32896683547126826
>>> random.seed(1234)
0.9664535356921388
```

DISTRIBUTION UNIFORME ...

```
>>> random.uniform(-1, 1) # distribution uniforme entre -1 et 1
-0.9850170598828256
>>> random.uniform(-1, 1)
0.8219519248982483
```

DISTRIBUTION NORMALE (GAUSSIENNE) ...

```
>>> random.gauss(0, 1) #0 est la moyenne, 1 est la déviation standard
-0.737130582050503
>>> random.gauss(0, 1)
1.4738187075003342
```

DISTRIBUTIONS EXPONENTIELLE, TRIANGULAIRE

```
>>> random.expovariate(-1) #si lambda > 0, entre 0 et +inf sinon 0 et - inf
-0.27024796731991774
>>> random.triangular(0, 1) #low, high
                            #help(random.triangular)
```

TRI / SÉLECTION ALÉATOIRE DANS UNE LISTE

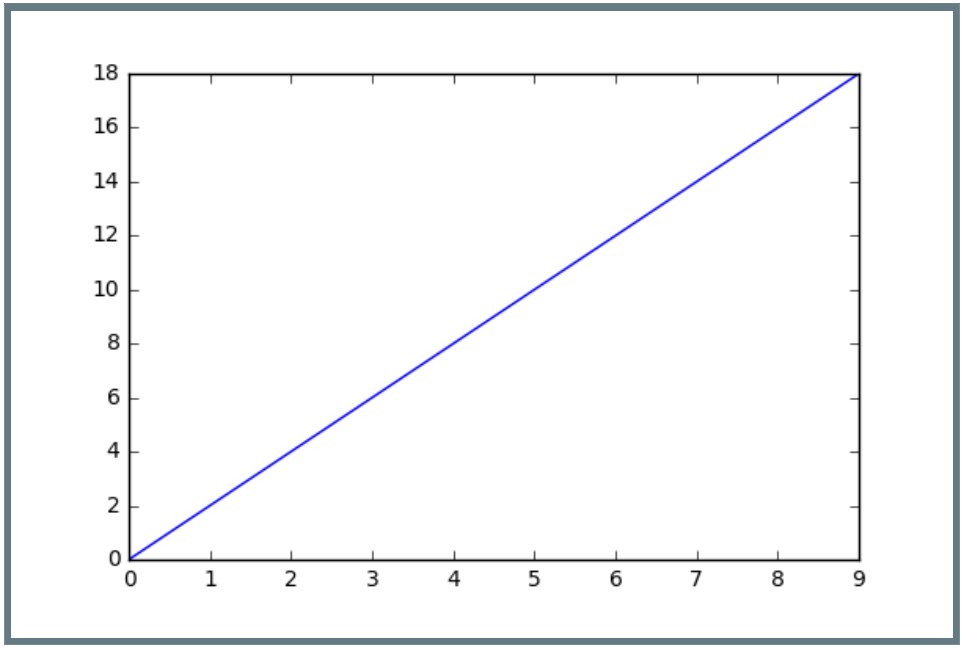
...

```
>>> R = list(range(10))
>>> print(R)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> random.shuffle(R)
>>> print(R)
[3, 2, 0, 5, 1, 9, 8, 4, 7, 6]
>>> random.choice(R)
7
>>> random.randint(0,100) #entier dans [0,100]
36
```

VISUALISER AVEC MATPLOTLIB

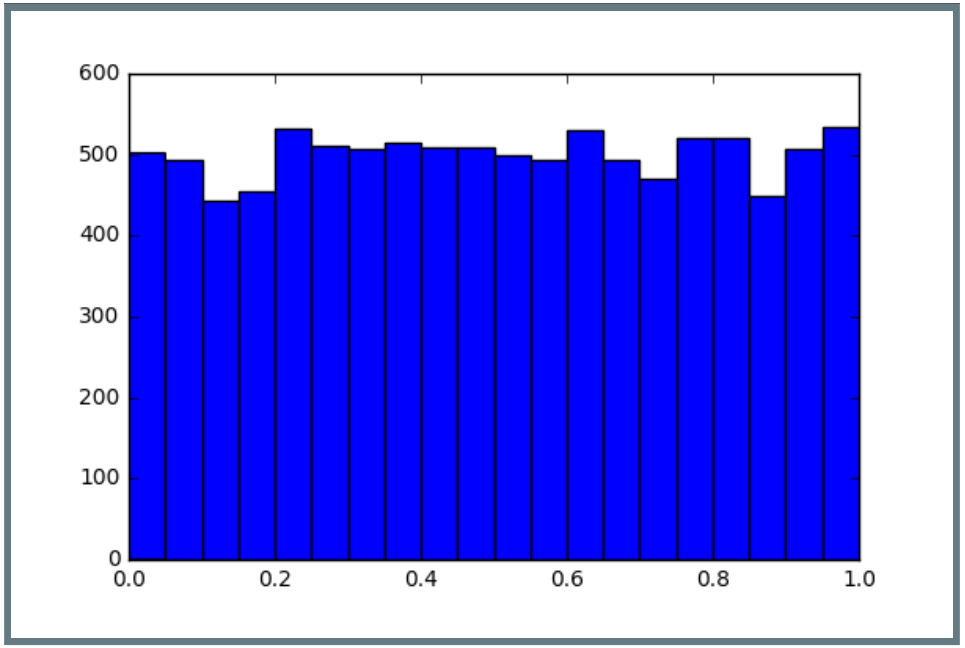
- Exemple très simple :

```
>>> import matplotlib.pyplot as plt
>>> x = list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> y = list(range(20, 2)) #0 à 19 avec pas de 2
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
>>> plt.plot(x,y) # 'x', 'o' ...
>>> plt.show()
```



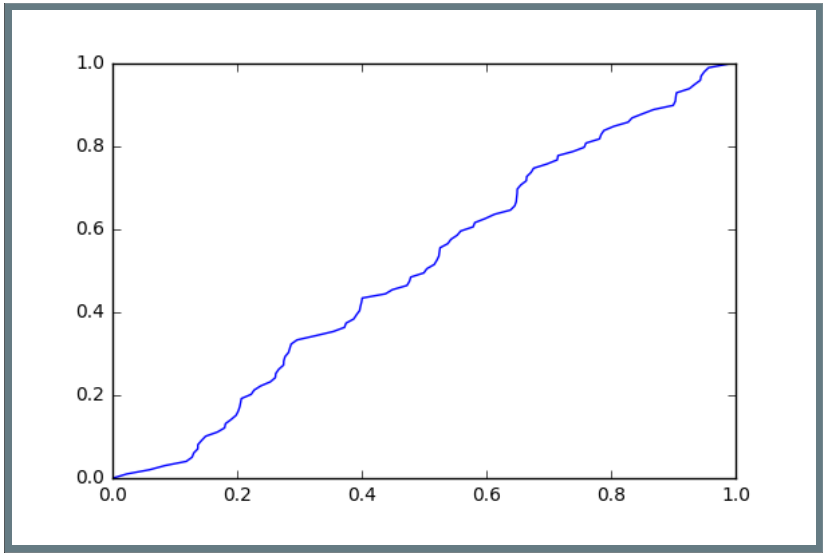
- Random.random (distribution uniforme [0,1])

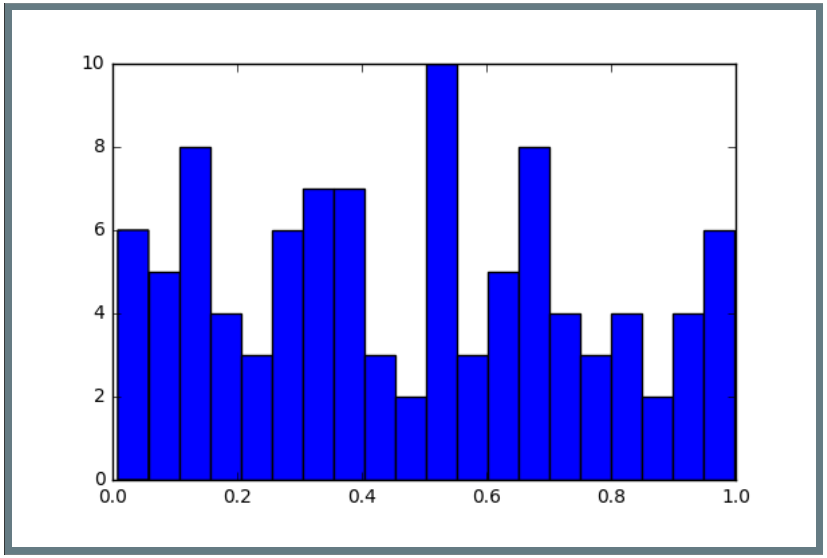
```
L = []
for i in range(10000):
    a = random.random()
    L.append(a)
plt.hist(L,20)
plt.show()
```



- On peut aussi représenter les **n** valeurs (triées) obtenues

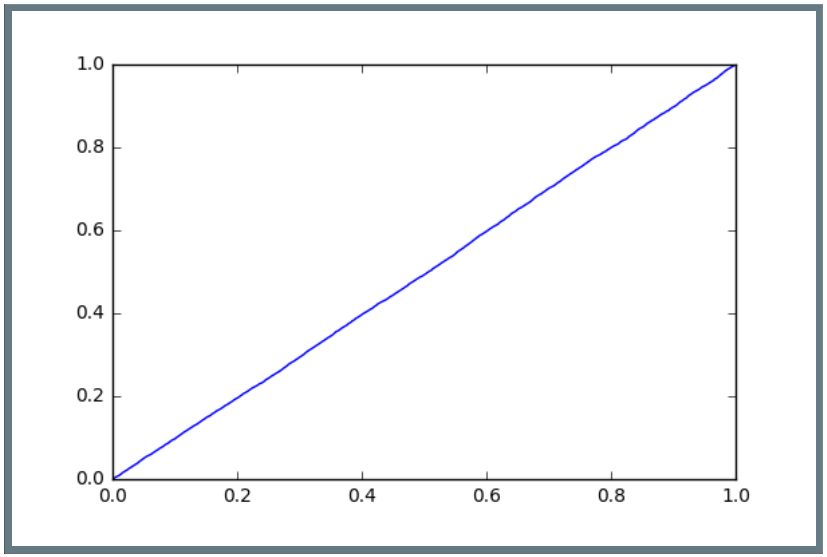
```
L = []
x = [i/99 for i in range(100)]
for i in range(100):
    a = random.random()
    L.append(a)
L.sort()
plt.plot(L,x)
plt.show()
```

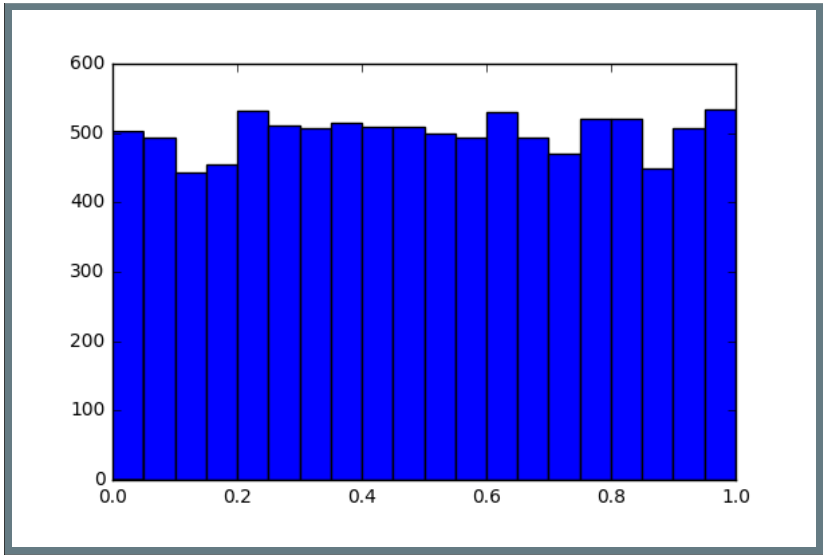




- Et bien sur si on augmente le nombre de tirage aléatoire, on se rapproche de la représentation théorique ...

```
L = []
x = [i/9999 for i in range(10000)]
for i in range(10000):
    a = random.random()
    L.append(a)
L.sort()
plt.plot(L,x)
plt.show()
```





UTILISATION DE LINSPACE

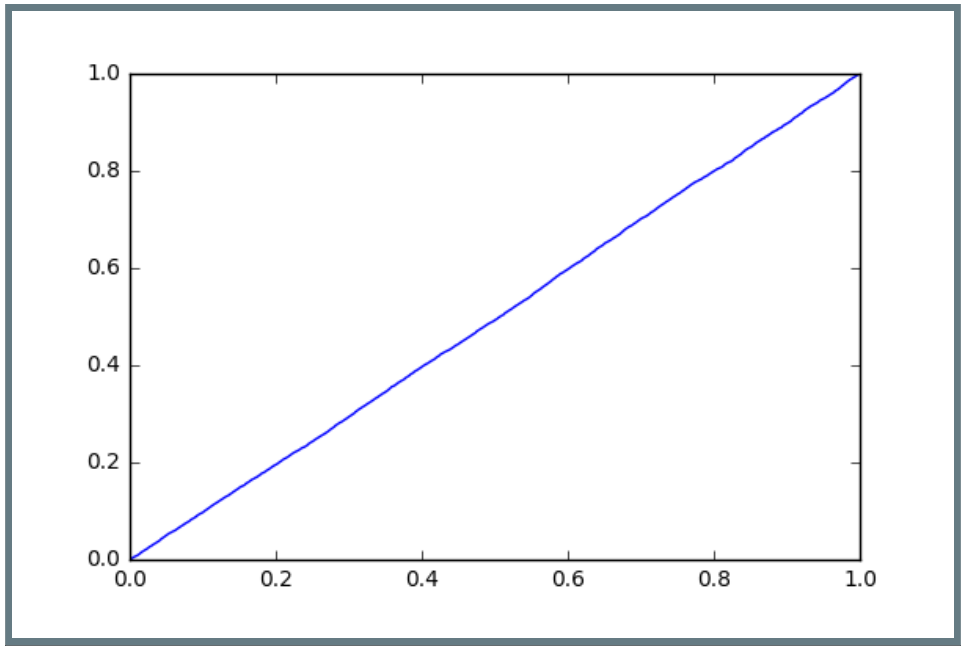
- Si on veut séparer de manière linéaire l'espace dans un intervalle, on peut utiliser numpy.linspace

```
>>> import numpy as np
>>> x = np.linspace(0, 9, 4)
>>> print(x)
[ 0.  3.  6.  9.]
```

UTILISATION DE LINSPACE

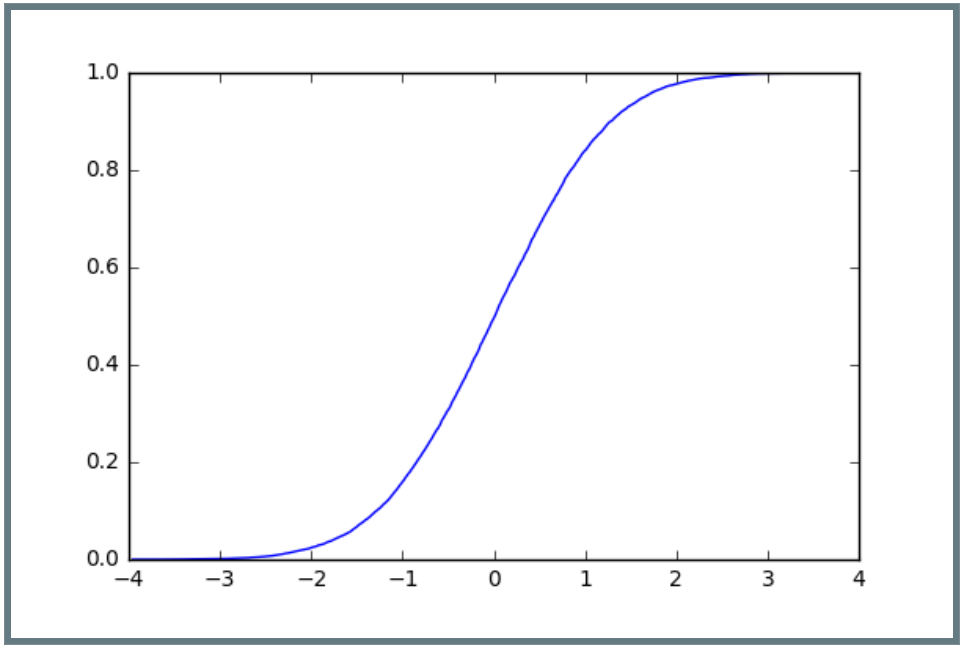
- Si on reprend l'exemple précédent :

```
import numpy as np
#x = np.linspace(0, 1, 10000)
x = [i/9999 for i in range(10000)]
plt.plot(L,x)
plt.show()
```



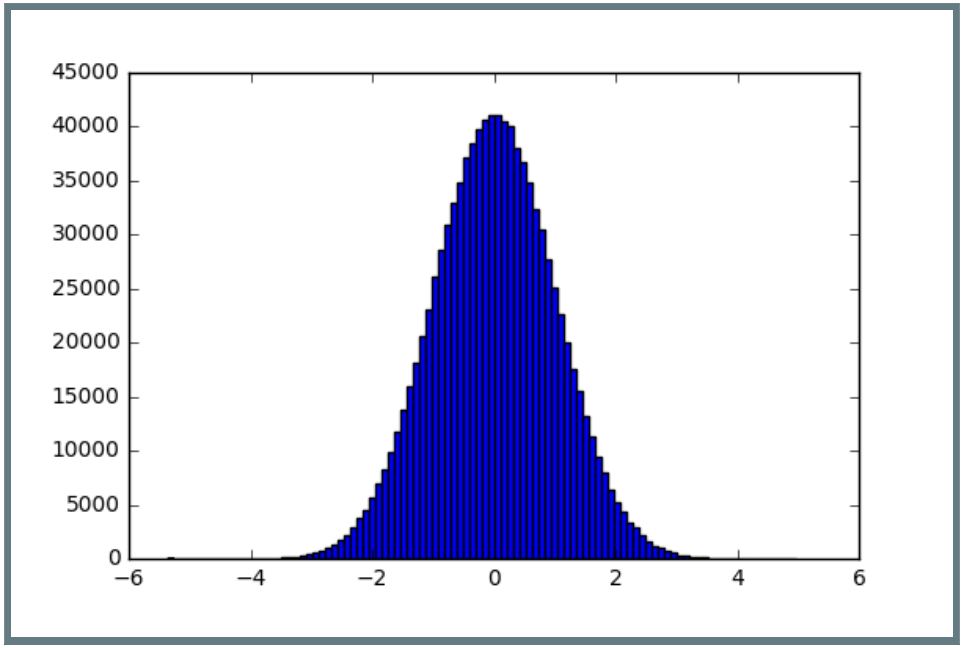
DE LA MÊME MANIÈRE POUR DISTRIBUTION GAUSSIENNE ...

```
L = []
for i in range(10000):
    a = random.gauss(0,1) # moyenne 0, ecart type 1
    L.append(a)
L.sort()
plt.plot(L,x)
plt.show()
```



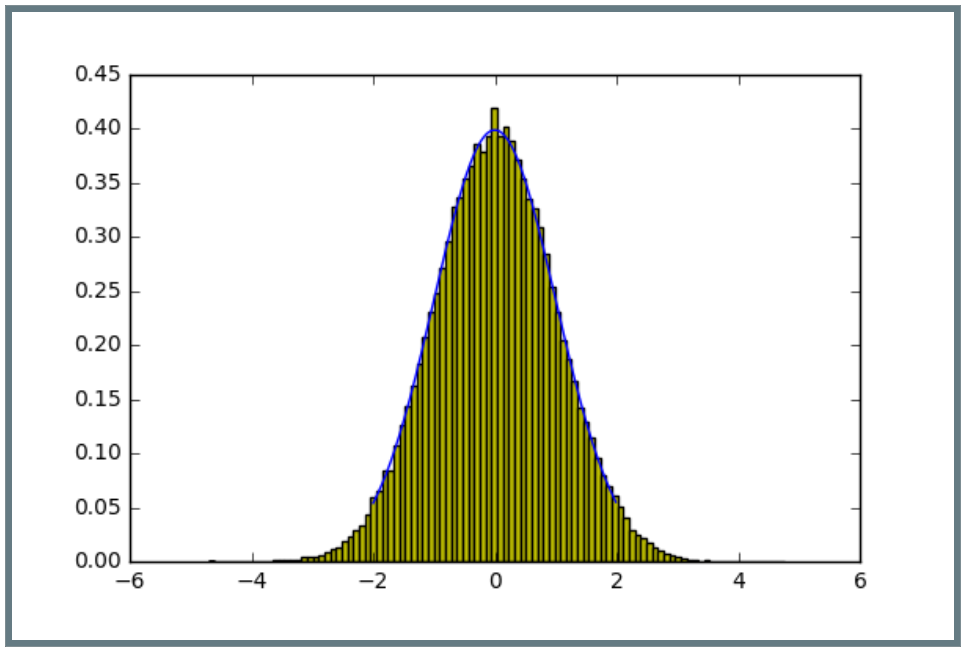
DE LA MÊME MANIÈRE POUR DISTRIBUTION GAUSSIENNE ... (HISTOGRAMMES)

```
L = []
for i in range(10000):
    a = random.gauss(0,1)
    L.append(a)
plt.hist(L, 100)
plt.show()
```



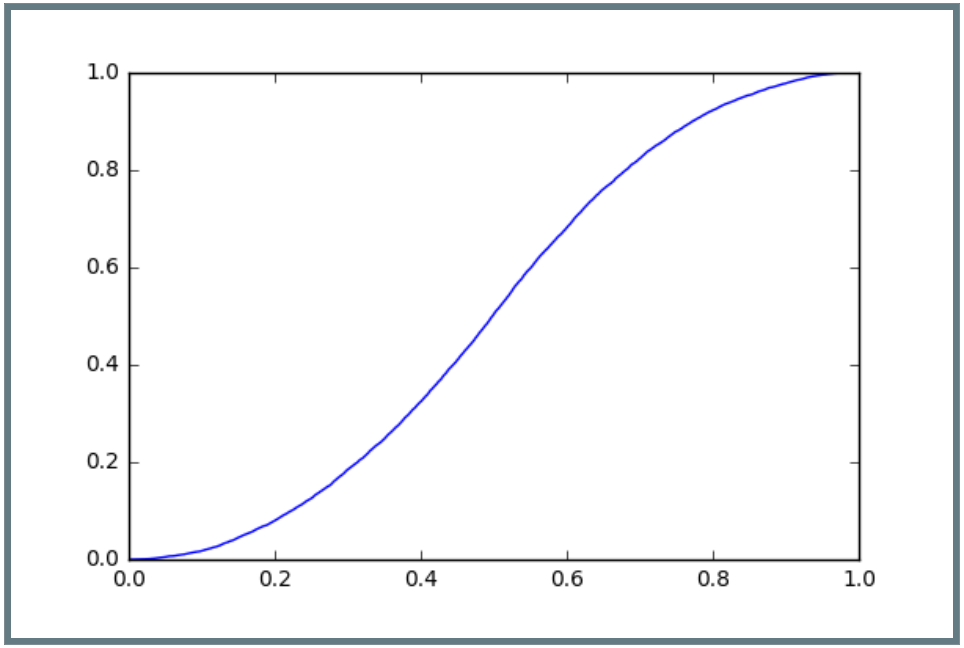
HISTOGRAMME RÉPARTITION GAUSSIENNE ET 'VRAI' GAUSSIENNE

```
import random, math
L = [random.gauss(0.,1.) for i in range(100000)]
plt.hist(L,100,normed='True',color='y')
x = [i/100. for i in range(-200,200)]
y = [math.exp(-s**2 / 2)/(math.sqrt( 2 * math.pi)) for s in x]
plt.plot(x,y)
plt.show()
```



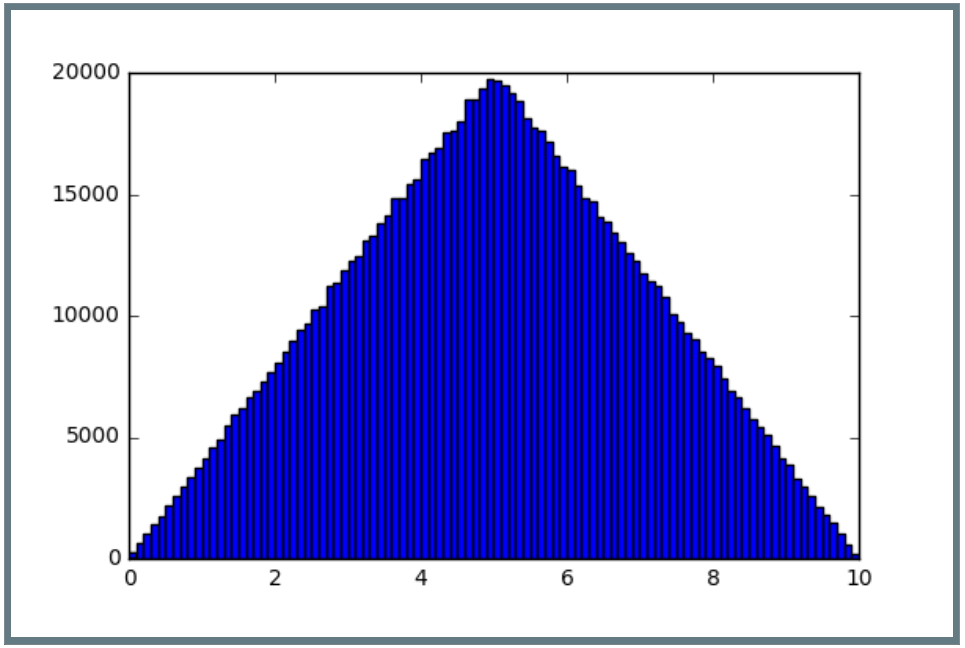
... ET TRIANGULAIRE

```
L = []
for i in range(10000):
    a = random.triangular(0,1)
    L.append(a)
L.sort()
plt.plot(L,x)
plt.show()
```



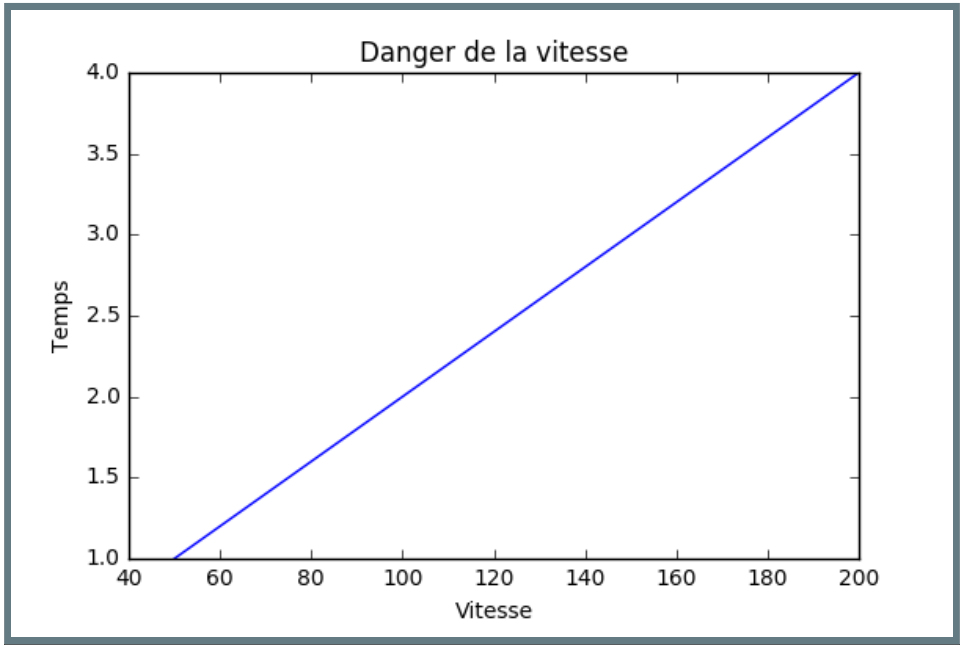
... ET TRIANGULAIRE (HISTOGRAMMES)

```
L = []
for i in range(10000):
    a = random.triangular(0,1)
    L.append(a)
plt.hist(L, 100)
plt.show()
```



AJOUTER DES TITRES, LÉGENDES

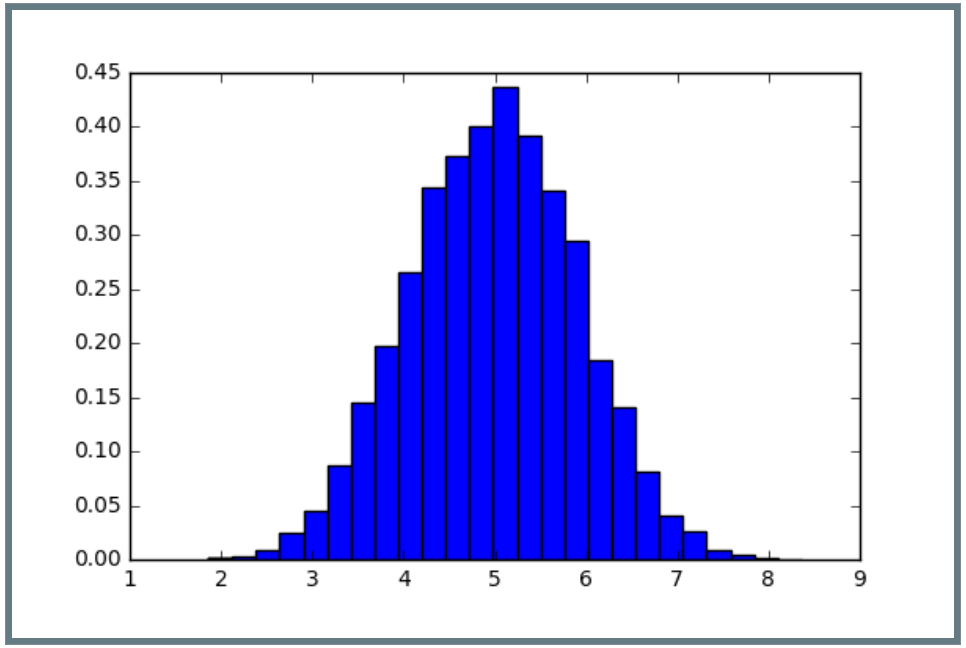
```
plt.title("Danger de la vitesse")
plt.plot([50,100,150,200], [1,2,3,4])
plt.xlabel('Vitesse')
plt.ylabel('Temps')
plt.show()
```



VÉRIFIONS LE THÉORÈME CENTRAL LIMITE :

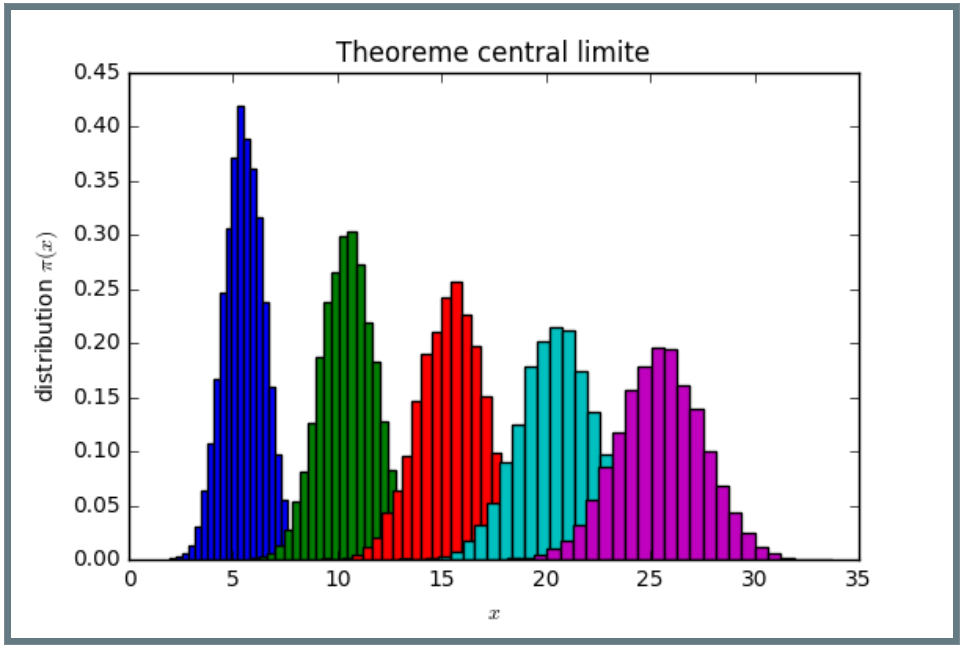
- La somme des variables aléatoires converge vers une distribution gaussienne

```
L = []
k = 10
for i in range(10000):
    a = 0
    for l in range(k):
        a = a+random.random()
    L.append(a)
```



EN UTILISANT PLUSIEURS VALEURS DE K ...

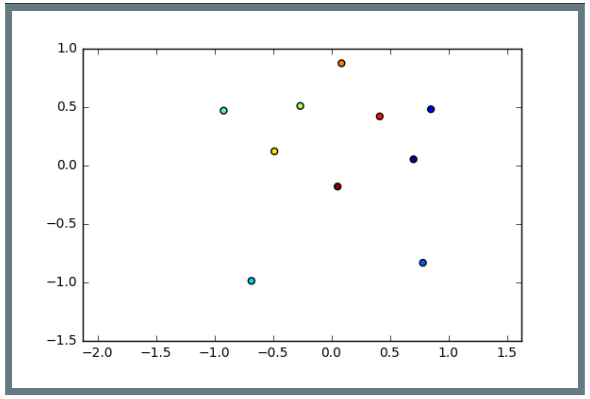
```
for k in range(11,52,10):
L = []
for i in range(10000):
    a = sum(random.random() for l in range(k))
    L.append(a)
plt.hist(L,25,normed='True')
plt.show()
```



MONTE CARLO SIMULATION

- Construisons un vecteur avec des valeurs uniformément distribuées dans [-1,1]

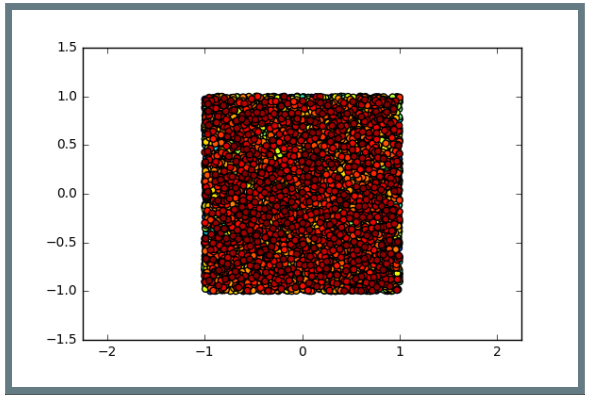
```
x, y = [], []
for i in range(10):
    a, b = random.uniform(-1.,1.), random.uniform(-1.,1.)
    x.append(a)
    y.append(b)
xyc = range( len( x ) )
plt.axis('equal') # meme intervalles entre les abscisses et ordonnées
plt.scatter(x,y, c=xyc, marker = '.', s=100) #taille des points = 100
plt.show()                                    #nb couleur=xyc
```



MONTE CARLO SIMULATION

- ... en augmentant le nombre de valeurs ...

```
x, y = [], []
for i in range(10000):
    a, b = random.uniform(-1.,1.), random.uniform(-1.,1.)
    x.append(a)
    y.append(b)
xyc = range( len( x ) )
plt.axis('equal') # meme intervalles entre les abscisses et ordonnées
plt.scatter(x,y, c=xyc, marker = '.', s=100) #taille des points = 100
plt.show()                                    #nb couleur=xyc
```



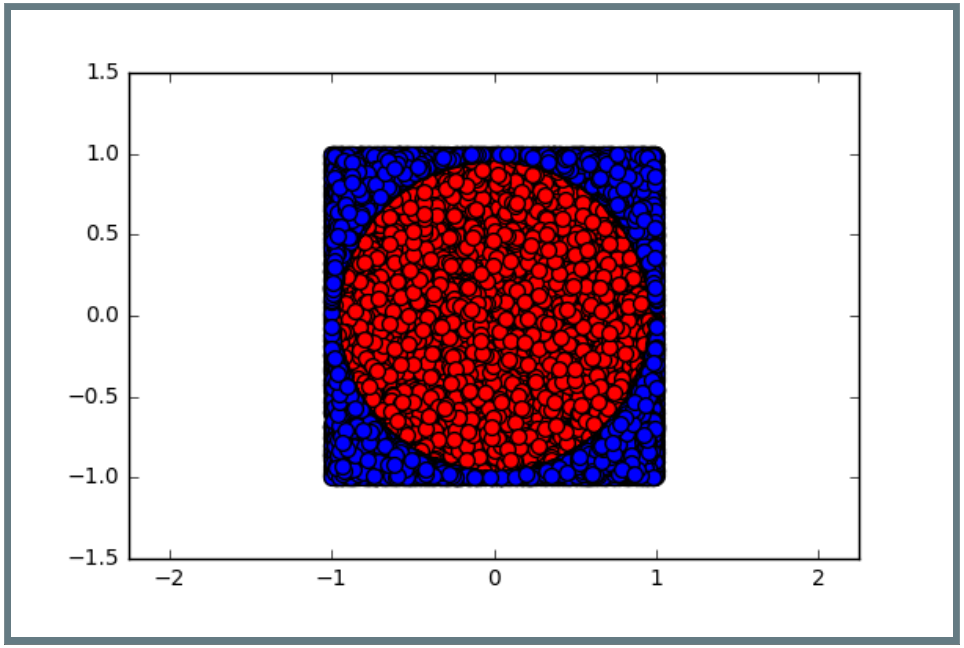
ESTIMONS MAINTENANT LA VALEUR DE PI

- en comptant le nombre des échantillons dans le carré qui se retrouvent à l'intérieur du cercle unité

```
x_inner, y_inner = [], []
x_outer, y_outer = [], []
for i in range(100000):
    a, b = random.uniform(-1.,1.), random.uniform(-1.,1.)
    length = math.sqrt(a**2 + b**2)
    if length < 1:
        x_inner.append(a)
        y_inner.append(b)
    else:
        x_outer.append(a)
        y_outer.append(b)
plt.scatter(x_inner, y_inner,c= 'red', marker = '.', s=200)
plt.scatter(x_outer, y_outer,c= 'blue', marker = '.', s=200)
print (4*len(x_inner)/float(len(x_inner) + len(x_outer)), math.pi)
plt.axis('equal')
plt.show()
```

ESTIMONS MAINTENANT LA VALEUR DE PI

- en comptant le nombre des échantillons dans le carré qui se retrouvent à l'intérieur du cercle unité
- en augmentant le nombre de tirages aléatoires, on améliore la précision de pi...




```
3.1415 3.141592653589793
```