TP2 TID Tsanta RANDRIATSITOHAINA INFO3

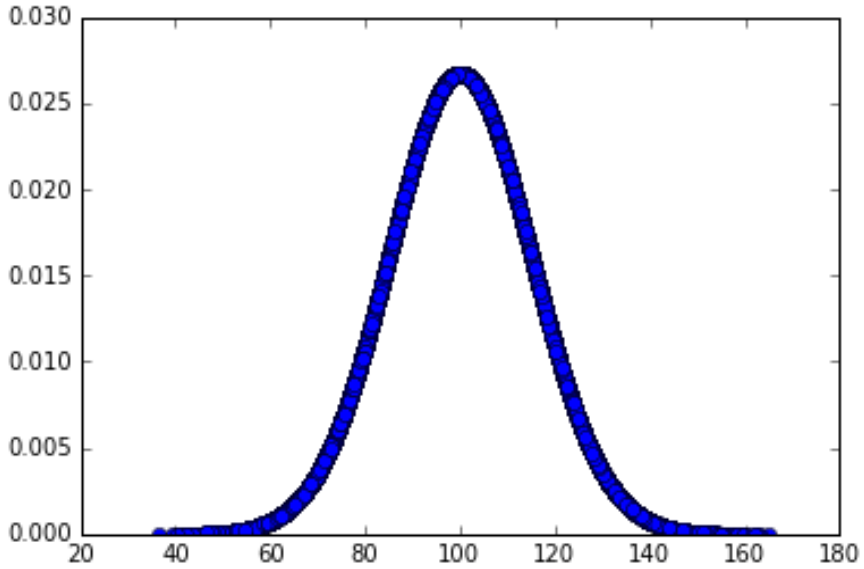
#PARTIE A

```
A = np.random.randn(100000,1) * math.sqrt(225) + 100

# Moyenne
moyenne = 0
for i in A:
moyenne = moyenne + i
moyenne = moyenne / len(A)
```
array([100.02599957])

```
# Variance
variance = 0
for i in A:
variance = variance + (i - moyenne)**2
variance = variance / (len(A) - 1)
```
Out[5]: array([223.34911139])

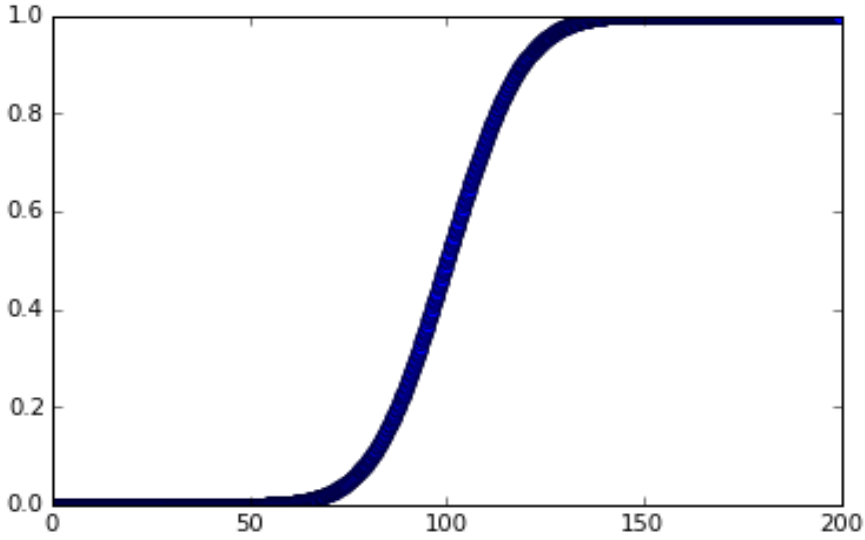
```
ecartType = math.sqrt(variance)
```
Out[6]: 14.944869065803076

```
# Densité de probabilité
def f(x, moyenne, ecartType):
return (1/(ecartType*math.sqrt(2*math.pi)))*math.exp(-0.5*((x - moyenne)/ecartType)**2)
f = np.vectorize(f)
Y = f(A, moyenne, ecartType)


plt.plot(A, Y, 'o')
plt.show()
```




```
# Fonction de repartition
j =0
rg = np.arange(0.2, 200, 0.2)
B = np.array([])
for k in rg:
```

```
B = np.append(B, float(len(A[A <= k])) / float(len(A)))
j = j + 1
plt.plot(rg, B, 'o')
plt.show()
```




```
# QI superieur à 130
q130 = float(len(A[A >= 130])) / float(len(A))
```
Out[7]: 0.02264

```
q60 = float(len(A[A <= 60])) / float(len(A))
```
Out[9]: 0.004

```
# les bornes 95% soit 2.5% pour les bornes inf et sup
borneInf = len(B[B <= 0.025])
borneSup = len(B[B <= 0.975])

valInf = rg[borneInf]
```
Out[11]: 71.0
```
valSup = rg[borneSup]
```
Out[7]: 129.40000000000001

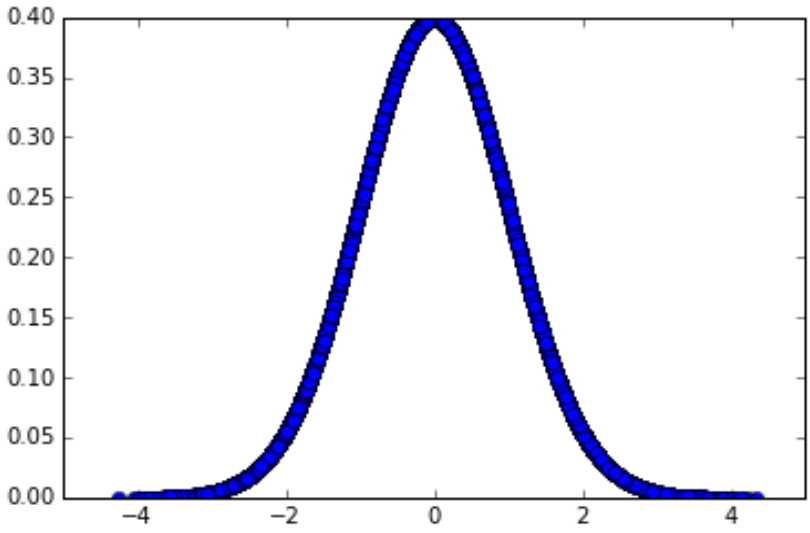
 # Partie B
```
A_c_r = (A - moyenne) / ecartType

# moyenne
moyenneR = 0
for i in A_c_r:
moyenneR = moyenneR + i
moyenneR = moyenneR / len(A_c_r)
```
Out[2]: array([-6.70579769e-14])

```
# Variance
varianceR = 0
```

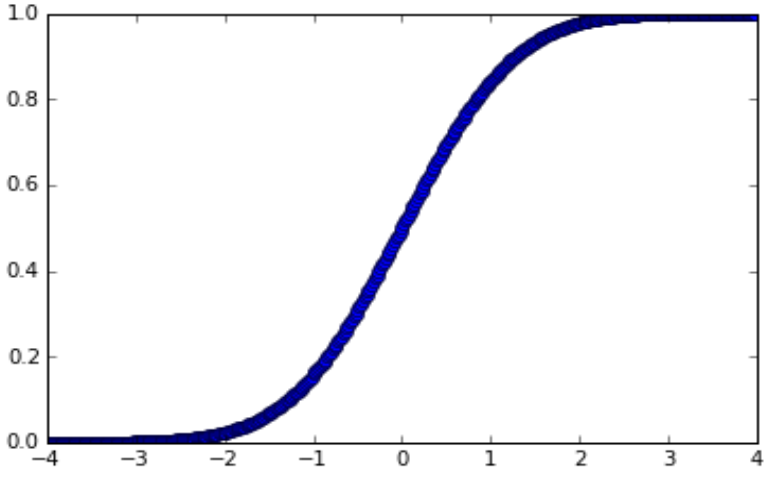
```
for i in A_c_r:
varianceR = varianceR + (i - moyenneR)**2
varianceR = varianceR / (len(A_c_r) - 1)
```
Out[3]: array([1.])

```
ecartTypeR = math.sqrt(varianceR)
```
Out[4]: 0.9999999999999906

```
#Densite & fct repartition
YR = f(A_c_r, moyenneR, ecartTypeR)
plt.plot(A_c_r, YR, 'o')
plt.show()
```




```
j =0
rg = np.arange(-4, 4, 0.02)
BR = np.array([])
for k in rg:
BR = np.append(BR, float(len(A_c_r[A_c_r <= k])) / float(len(A_c_r)))
j = j + 1
plt.plot(rg, BR, 'o')
```



q120_125 = float(len(A[A >= 120])) / float(len(A)) + float(len(A[A <= 125])) / float(len(A))
Out[8]: 1.04363

q120_125R = (float(len(A_c_r[A_c_r >= ((120 - moyenne)/ecartType)])) + float(len(A_c_r[A_c_r <= ((125 - moyenne)/ecartType)]))) / float(len(A_c_r))

Out[9]: 1.04363