

```

##
# Corrigé des exercices "plus difficiles"
##

## dates d'anniversaires

#1)
nbre_eleves = 35
p = 1 # p comme produit

for k in range(nbre_eleves):
    p = p*(365-k)/365
print(1-p)

#2)

def proba_meme_jour_naissance(nbre_eleves):
    p = 1 # p comme produit
    for k in range(nbre_eleves):
        p = p*(365-k)/365
    return 1-p

# pour afficher les valeurs
for nbre_eleves in range(0,35):
    print("nbre_eleves: " + str(nbre_eleves) + "--> proba : " + str(
        proba_meme_jour_naissance(nbre_eleves)))

# script pour tracer la courbe
import matplotlib.pyplot as plt

nbre_eleves = [k for k in range(80)]
probas = [proba_meme_jour_naissance(k) for k in nbre_eleves]

plt.xlabel("nombre d'élèves")
plt.ylabel("probabilité $1-p$")

plt.plot(nbre_eleves, probas)

plt.show()

## Méthode de Monte-Carlo
import random
import numpy as np

# 1
def simulation(nbre_lancers):
    nbre_succes = 0
    for _ in range(nbre_lancers):
        x = random.uniform(-1,1)
        y = random.uniform(-1,1)
        if x**2 + y**2 <= 1:
            nbre_succes += 1
    return 4* nbre_succes / nbre_lancers

# 2
s = 0
for _ in range(100):
    pi = simulation(1000)
    s = s + pi
print(s / 100)

# 3
plt.clf()
nbre_lancers = 1000

```

```

for _ in range(nbre_lancers):
    x = random.uniform(-1,1)
    y = random.uniform(-1,1)
    if x**2 + y**2 <= 1:
        plt.plot([x],[y], '.r') # affiche un seul point
    else:
        plt.plot([x],[y], '.b')

# tracé du cercle unité
T = np.linspace(0,2*np.pi,200) # subdivision de l'intervalle [0,2pi] en 200 points
X = np.cos(T) # X = [cos(t) for t in T]
Y = np.sin(T)
plt.plot(X,Y)
plt.show()

```

Ex0 21 suite de Syracuse

```

#1)
def f(n):
    if n % 2 == 0:
        return n//2
    else:
        return 3*n+1

```

```

#3)
def retouralunite(u0):
    n = 0
    u = u0
    while u != 1:
        u = f(u)
        n += 1
    return n

```

```
retouralunite(13) # renvoie bien 9
```

Exo 23 Autour des nombres premiers

```

#1)
def est_premier(n):
    if n <= 1:
        return False #
    # si n>=2
    d = 2
    while d**2 <= n:
        if n % d == 0: # si k divise n
            return False
        d = d + 1
    # arrivé ici, on est sûr que n est premier
    return True

```

```
#2) dans le pire des cas, si n est premier, racine de n divisions environ
```

```

#3)
est_premier(2**31-1) # il est premier
#4)

```

```

def premierSuivant(n):
    p = n
    while not est_premier(p):
        p = p + 1
    # arrivé ici, on est sûr que p est premier
    return p

```

Exo 24

```
#1)

liste_premiers = []
for k in range(10000):
    if est_premier(k):
        liste_premiers.append(k)

# ou avec une comprehension list

liste_premiers = [k for k in range(2,10000) if est_premier(k)]

#2) il y en a 1229
print(len(liste_premiers))

#3) il y en a 609
sous_liste = [k for k in liste_premiers if k % 4 == 1]
```