```python
#%% TP 20

import random as rd

#%% exo 1

# q1
def lancer_de():
    return rd.randint(1,6)

# q2
def tirer():
    urne = ['R','R','R','B','B','V']
    return rd.choice(urne)

# q3
def tirer_suc(n):
    return [tirer() for k in range(n)]

# q4
def tirer_sim(n):
    urne = ['R','R','R','B','B','V']
    return rd.sample(urne,n)

# q5
def hello(p):
    t = rd.random()
    if t<p :
        return "bonjour"
    else :
        return "au revoir"

# q6
def jour():
    t = rd.random()
    if t<0.2 :
        return "lundi"
    elif t<0.9 :
        return "mardi"
    else :
        return "mercredi"

#%% exo 2

# q1
def repete_de(N):
    X = 0
    for k in range(N):
        res = lancer_de()
        if res == 6:
            X = X+1
    return X

# q2
def frequence(N):
    return repete_de(N)/N

# q3
# frequence(10000) renvoie environ 1/6

# q4
```

```
absi = [N for N in range(1,2000)]
ordo = [frequence(N) for N in absi]
import matplotlib.pyplot as plt
plt.plot(absi,ordo)
plt.show()

#%%
# q5
X = 0
N = 10000
for k in range(N):
    if jour()=="lundi":
        X = X+1
print(X/N)

#%%
# q6
def nb_pair(n,d):
    urne = [k for k in range(1,n+1)]
    liste_tires = rd.sample(urne,d)
    Y = 0
    for x in liste_tires:
        if x%2==0:
            Y = Y+1
    return Y

# q7
X = 0
N = 10000
for k in range(N):
    if nb_pair(10,2)==0:
        X = X+1
print(X/N)

#%% exo 3

# q1
def reponse():
    categ = rd.choice(["H","L","S"])
    t = rd.random()
    if categ=="H":
        return t<=0.9
    if categ=="L":
        return t<=0.6
    else :
        return t<=0.3

# q2
X = 0
N = 10000
for k in range(N):
    if reponse():
        X = X+1
p = X/N

# q3
def essais(n):
    for k in range(n):
        if reponse():
            return True
    return False
```

```
#%% exo 4

# q1
def pas():
    t = rd.random()
    if t<0.15:
        return 1
    elif t<0.4:
        return 2
    else :
        return 3

# q2
def position():
    x = 0
    for k in range(10):
        x = x + pas()
    return x

# q3
X = 0
N = 100000
for k in range(N):
    if position()==25:
        X = X+1
print(X/N)
# on trouve environ 0,166

# q4
# on fait la moyenne des positions
pos = 0
N = 100000
for k in range(N):
    pos = pos + position()
print(pos/N)
# on trouve environ 24,5

#%% exo 5

# q1
# det(A)=ad-bc
# A est inversible si et seulement si det(A) != 0

# q2
import numpy as np
def inversible(A) :
    det = A[0,0]*A[1,1] - A[0,1]*A[1,0]
    return det != 0
# à tester avec A = np.array([[1,2],[3,4]])

# q3
def matrice_alea():
    a = rd.random()
    b = rd.random()
    c = rd.random()
    d = rd.random()
    return np.array([[a,b],[c,d]])

# q4
X = 0
```

```
N = 10000
for k in range(N):
    A = matrice_alea()
    if inversible(A):
        X = X+1
print(X/N)
# on obtient une probabilité de 1 :
# chacune des 10 000 matrices aléatoires testées est
# inversible ! En même temps, en prenant a,b,c,d au hasard
# on a peu de chance que a*d = b*c. Tellement peu de chance
# que la probabilité que cela arrive est nulle.
# Cependant, cela ne signifie pas que toutes les matrices
# sont inversibles ; mais seulement que la probabilité
# de trouver une matrice non inversible au hasard est nulle.

#%% exo 6

# q1
# pi

# q2
# P(M appartient à D) = aire(D)/aire(carré) = pi/4

# q3
# on simule l'expérience consistant à placer M au hasard
# dans le carré (aka : tirer une fléchette) et on estime
# la probabilité que M appartienne à D (aka : la probabilité
# d'atteindre la cible). Cette probabilité est une approximation
# de pi/4 donc en multipliant par 4 on a une approximation de pi

def tir_flechette():
    x = rd.uniform(-1,1)
    y = rd.uniform(-1,1)
    return x,y

def cible_atteinte():
    x,y = tir_flechette()
    return x**2 + y**2 < 1

X = 0
N = 10000
for k in range(N):
    if cible_atteinte():
        X = X+1
print(4*X/N)
```