Exercice 1

```
# Exercice 1)
# 1.1) Recherche d'un élément dans une liste

def recherche(L,e): # Recherche si e dans liste L
    for k in range(len(L)):      # Parcours de L
        if L[k] == e:            # Est-ce  e dans L ?
            return True
    return False                 # A LA FIN renvoyer False
```

2.a) Le plus concis est d'utiliser duplication (∗) et concaténation (+)

```
L = [0] * 90 + [1] * 10
```

on peut aussi exécuter deux boucles for pour remplir une liste vide avec 90 zéros et dix uns (mais c'est moins élégant) :

```
L = []
for k in range(90):
    L.append(0)
for k in range(10):
    L.append(1)
```

2.b)

```
from random import randint

B = []
for k in range(5):
    idx = randint(0,len(L)-1)   # Indice au hasard dans L
    e = L.pop(idx)       # Retrait et renvoi de L[idx]
    B.append(e)          # Ajout dans B
```

2.c)

```
if recherche(B,1):       # recherche(B,1) est un booléen
    print("gagne")
else :
    print("perdu")
```

3) Estimation empirique de la probabilité

```
N = 100000
C = 0
for _ in range(N):
    L = [0] * 90 + [1] * 10
    B = []
    for k in range(5):
        idx = randint(0,len(L)-1)   # ou 99-k
        e = L.pop(idx)
        B.append(e)
    if recherche(B,1):
        C = C+1
print("Estimation probabilite : ",C/N)
```

en augmentant la valeur de `N` (ici 100000) on obtient une estimation d'environ 0.41 de la probabilité d'acheter au moins un billet gagnant.

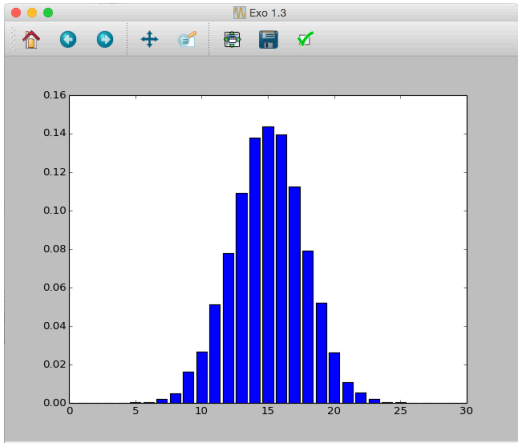
Exercice 2.

```
# Importation des modules nécessaire
from matplotlib.pyplot import bar, show, figure
from random import random


# Exercice 2
# (2.1)
def simulBernoulli(p):
    rnd = random()
    if rnd <= p:
        return 1
    else:
        return 0


# (2.2)
def loiBernoulli(p,k):
    N = 1000
    compteur = 0
    for loop in range(N):
        if simulBernoulli(p) == k:
            compteur = compteur + 1
    return compteur/1000

# (2.3)
def freqBernoulli(p):
    freq = [loiBernoulli(p,0), loiBernoulli(p,1)]
    figure("Exo 1")
    bar([0, 1], freq)
    show()

p = 0.3  # par exemple
freqBernoulli(p)
```

Exercice 3.

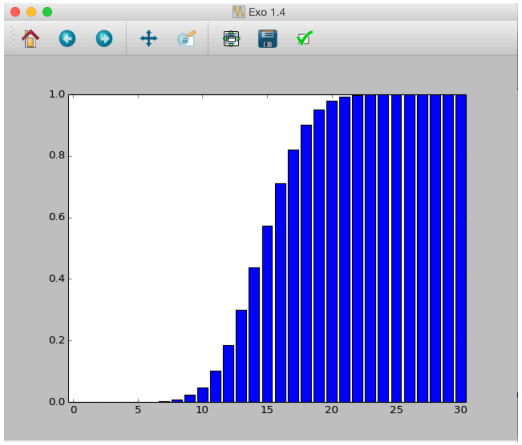
```
# Exercice 3
# (3.1)
def simulBinomiale(n,p):
    y = 0
    for k in range(n):
        y += simulBernoulli(p)
    return y

# (3.2)
def loiBinomiale(n,p,k,N):
    compteur = 0
    for loop in range(N):
        if simulBinomiale(n,p) == k:
            compteur = compteur +1
    return compteur/N

# (3.3)
def freqBinomiale(n,p,N):
    R = [0] * (n+1)
    for i in range(N):
        y = simulBinomiale(n,p)
        R[y] += 1
    return [r/N for r in R]

# (3.4)
N = 10000
n = 30
p = 0.5
Y = freqBinomiale(n,p,N)
X = [x for x in range(n+1)]
figure("Exo 2.3")
bar(X,Y)
show()
```

On obtient le graphique suivant (qui peut varier d'une implémentation à l'autre) :




```
# (3.5)
Z = [0] * (n+1)
Z[0] = Y[0]
for k in range(1,n+1):
    Z[k] = Z[k-1] + Y[k]
figure("Exo 2.4")
bar(X,Z)
show()
```

Ce qui produit :

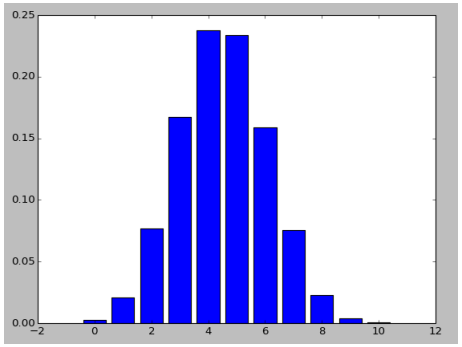


Exercice 4.

```
# 1)
from random import shuffle
L = [1] * 45 + [0] * 55
```

```
shuffle(L)

# 2)

from random import randint

def tir():
    X = 0
    for i in range(n):
        k = randint(0,N-1)
        X = X + L[k]
    return X

# 3)

def frequence(m = 100000):
    Issues = [0] * (n+1)
    for i in range(m):
        x = tir()
        Issues[x] += 1
    Loi = [ k/m for k in Issues ]
    return Loi

# 4)

Y = frequence()
figure("Exo 3")
X = [ x for x in range(n+1) ]
bar(X,Y)
show()
```

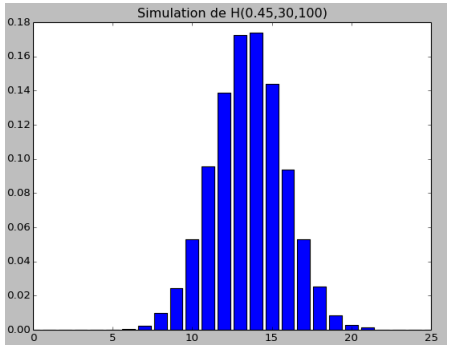
On obtient :



Exercice 5.

```
# 1)
from random import shuffle, randint

# Simulation de la loi hypergeometrique
def hypergeometrique(N, n, p):
    # Constitution de l'urne
    Np = int(N*p)
    Nq = N - Np
    Urne = [1] * Np + [0] * Nq
    shuffle(Urne)
    # Simulation du tirage sans remise
    X = 0
    for k in range(n):
        M = len(Urne)
        i = randint(0,M-1)
        boule = Urne.pop(i)
        X = X + boule
    # X contient le nbre de blanches obtenues
    return X

# 2)
# Obtention du tableau des frequences
def frequence(N, n, p, f=10000):
    Freq = [0] * (n+1)
    for i in range(f):
        X = hypergeometrique(N,n,p)
        Freq[X] += 1
    return [x/f for x in Freq]

# 3)
# Simulation de H(0.3,10,100)
N = 100
n = 30
p = 0.45
K = [k for k in range(n+1)]
X = frequence(N,n,p)


# Histogramme de la loi
from matplotlib.pyplot import bar, show, figure
figure(1)
bar(K,X)
show()
```

Ce qui produit :




```
# 5)
# Histogramme de la fonction de repartition
# Liste des valeurs
Fx = [X[0]] + [0] * n
for k in range(1,n+1):
    Fx[k] = Fx[k-1] + X[k]
# Tracé
figure(2)
bar(K,Fx)
show()
```

On obtient :