Exercice 1.

```
rd.randint(6)+1
```

Exercice 2.

À retenir

⋆ La commande **rd.randint**(n,m,p), où n, m, p sont des entiers naturels, permet de simuler p tirages d'un nombre entier compris dans $[[n; m-1]]$

Exercice 3.

```
x=rd.randint(100)+1
print(x)
n=0
y=int(input('entrer une valeur entre 1 et 100 afin de trouver la valeur cachée: ')
while y<x:
    y=int(input('entrer une valeur supérieure: '))
    n=n+1
while y>x:
    y=int(input('entrer une valeur inférieure: '))
    n=n+1
n=n+1
print('bravo vous avez trouver x=',x)
print('il y a eu ',n,'tentatives')
```

Exercice 4.

```
s=0
for k in range (1,11):
    if rd.randint(6)+1==6
        s=s+1
print(s)
```

Exercice 5. ⋆

```
n=0
while rd.randint(6)+1!=6 or rd.randint(6)+1!=6:
    n=n+1
print(n)
```

⋆

```
import numpy as np
import numpy.random as rd

import matplotlib.pyplot as plt

F=np.zeros(11)
for i in range(1,1001):
    s=0
    for k in range(10):
       if rd.randint(6)+1==6:
                  s=s+1
    F[s]=F[s]+1
print(F)
X=[0,1,2,3,4,5,6,7,8,9,10]
plt.bar(X,F)
plt.xlabel( ' valeurs ')
plt.ylabel( ' nombres ')
plt.title( 'Pour 1000 expériences')
plt.show()
```

Exercice 6.

```
if rd.random()<0.3:
    print('Pile')
else:
    print('Face')
```

Exercice 7. Un dé cubique, dont les faces sont numérotées de 1 à 6, est tel que, lorsqu'on le lance, le 6 sort trois fois plus que le 1 alors que les numéros 1, 2, 3, 4 et 5 ont autant de chances d'apparaître.

1. $p = 0,125$.
2.
```
if rd.random()<0.125:
    print('1')
elif (0,125<=rd.random()<0.25) and (rd.random()<0.25):
    print('2')
elif (0.25<=rd.random()) and (rd.random()<0.375):
    print('3')
elif (0.375<=rd.random()) and (rd.random()<0.5):
    print('4')
elif (0.5<=rd.random()) and (rd.random()<0.625):
    print('5')
else:
    print('6')
```
3.
```
n=int(input('entrer un entier n:'))
F=np.zeros(6)
for k in range(n):
    if rd.random()<0.125:
        F[0]=F[0]+1
    elif (0,125<=rd.random()<0.25) and (rd.random()<0.25):
        F[1]=F[1]+1
    elif (0.25<=rd.random()) and (rd.random()<0.375):
        F[2]=F[2]+1
    elif (0.375<=rd.random()) and (rd.random()<0.5):
        F[3]=F[3]+1
    elif (0.5<=rd.random()) and (rd.random()<0.625):
        F[4]=F[4]+1
    else:
        F[5]=F[5]+1
F=F/n
X=[1,2,3,4,5,6]
plt.bar(X,F)
plt.xlabel( ' numéro du dé ')
plt.ylabel( ' fréquences ')
plt.show()
```
4.
5. Quelle est la probabilité de sortie d'un numéro pair ?
6.
```
n=int(input('entrer un entier n:'))
F=np.zeros(2)
for k in range(n):
    if rd.random()<0.625:
        F[0]=F[0]+1
    else:
        F[1]=F[1]+1
F=F/n
X=['Pair','Impair']
plt.bar(X,F)
plt.xlabel( ' Parité ')
plt.ylabel( ' fréquences ')
plt.show()
```
7.

Exercice 8. On lance une infinité de fois une pièce telle que la probabilité de faire pile soit égale $\frac{1}{3}$.

1.
```
if rd.random()<0.3:
    print(1)
else:
    print(0)
```

2.
```
def nombre_de_pile(n):
    p=0
    for k in range(n):
        if rd.random()<0.3:
            p=p+1
    return p
```

3.
```
def nombre_de_pile(n):
    p=0
    for k in range(n):
        if rd.random()<0.3:
            p=p+1
    return p/n
```

4.

5.
```
def rang_du_pile(n):
    L=[0]
    for k in range(n):
        r=0
        while rd.random()>0.3:
            r=r+1
        L=L+[r]
    return np.mean(L)
```

Exercice 9. On considère la marche aléatoire d'un individu sur l'ensemble $\mathbb{Z}$:

⋆ À l'instant 0, l'individu est à l'abscisse 0 ;

⋆ à tout instant n, il se déplace d'une unité à gauche où à droite avec la même probabilité.

1.
```
n=int(input('Donner une valeur de n:'))
s=0
for k in range(n):
    if rd.random()<1/2:
            s=s+1
    else:
            s=s-1
print(s)
```

2. (a) Que dire de l'événement A_n si n est impair ?

   (b) Calculer la probabilité p_n que l'événement A_n soit vrai.

   (c)
```
n=int(input('Donner une valeur de n:'))
s=0
for k in range(n):
        if r.random()<1/2:
            s=s+1
        else:
        s=s-1
if s==0:
    print('L\'individu est retourné au départ')
else:
    print('L''individu n'est pas retourné au départ')
```

Exercice 10. 1.

2.
```
n=int(input('Donner une valeur de n:'))
p=1
for k in range(1,n)
    p=p*(1-k/365)
    p=1-p
print(p)
```

3.

4.

5.
```
k=2
p=1-1/365
while (1-p)<1/2:
    p=p*(1-k/365)
    k=k+1
print(k)
```