Exo 1 Solution :

```
def predict_state(arr, c_state):

    # count number of transitions from each state to the next and create a transition matrix

    tran_matrix = [[0 for i in range(3)] for i in range(3)]
    prob_matrix = [[0 for i in range(3)] for i in range(3)]

    for i in range(len(arr)-1):

        tran_matrix[arr[i]][arr[i+1]] += 1
    #calculate total for each row
    row = [0, 0, 0]

    for i in range(len(tran_matrix)):
        for j in range(len(tran_matrix)):
            row[i]+=tran_matrix[i][j]

    #calculate probability of transition
    for i in range(len(tran_matrix)):
        for j in range(len(tran_matrix)):
            prob_matrix[i][j] = tran_matrix[i][j]/float(row[i])

    print(prob_matrix)

    return prob_matrix[c_state].index(max(prob_matrix[c_state]))


arr = [1,2,0,1,0,1,0,2,0,1,0,1,2,0,1,2,1,0,1,0,1,0,1,0,2,1,2,1,0,1,0,1,0,2,0,2,0,1,0,1,0,2]
c_state = 2
print(predict_state(arr, c_state))
```

Solution Exo 2:

```
"""
On veut calculer l'aire sous la courbe d'une fonction en approximant la valeur par un
algorithme de Monte Carlo.
Calculer l'aire pour la fonction    y = 0.5*x + 0.2     pour un plan avec x et y allant de 0
à 1.
Conseil : utiliser la méthode random du module random pour générer un nombre aléatoire entre
0 et 1.
"""


import random

def appro_area(nb_points, fonction):
    liste_points = [(random.random(), random.random()) for _ in range(nb_points)]
    liste_points = [point for point in liste_points if point[1] < fonction(point[0])]
    return len(liste_points) / nb_points

affine = lambda x: 0.5*x + 0.2
print(appro_area(666, affine))
```

Solution Exo3

```
# SOLUTION TAKEN FROM https://gist.github.com/louismullie/3769218
import random as r
import math as m

# Number of darts that land inside.
inside = 0
# Total number of darts to throw.
total = 10000

# Iterate for the number of darts.
for i in range(0, total):
    # Generate random x, y in [0, 1].
        x2 = r.random()**2
        y2 = r.random()**2
    # Increment if inside unit circle.
        if m.sqrt(x2 + y2) < 1.0:
            inside += 1

# inside / total = pi / 4
pi = (float(inside) / total) * 4

# It works!
print(pi)
```

Solution exo4 :

- Quel est l'avantage, en termes de sécurité, de mettre à disposition un checksum?
  - S'assurer que le téléchargement est complet et correspond bien au fichier qui était sur le site web ("Integrity")
- Pourquoi comparer le checksum plutôt que le fichier qui est téléchargé?
  - Le checksum de référence ferait la taille du fichier. Si celui-ci fait plusieurs mega/giga… c'est une opération trop coûteuse en bande passante et en ressources informatiques.
- Pourquoi est-ce considéré comme une technique probabilistique?
  - Il y a une très faible probabilité qu'un fichier différent génère le bon checksum (plus d'info sur https://fr.wikipedia.org/wiki/Collision_(informatique). La probabilité est si faible que l'on considère le test suffisant.