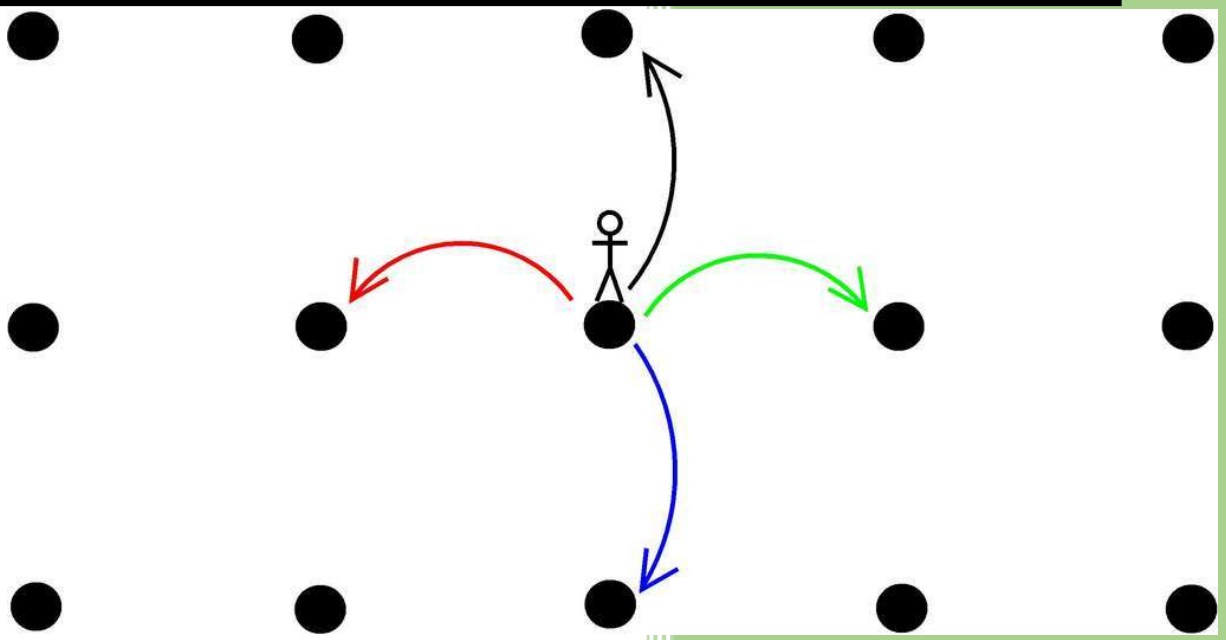
2020

Travail Pratique 2 : Générateurs de nombres aléatoires



ATTIA Christian

UQAC

23/03/2020

Partie 1 : Création d'un générateur :

Pour ce Générateur pseudo-aléatoire, j'ai utilisé python 3, avec la librairie PrettyTable afin d'afficher les tests de manière plus esthétique. Pour ce faire, j'ai utiliser la formule donnée en cours

```
        self.seed = (self.multiplier*self.seed+self.increment)%self.modulus
```

Ensuite, afin de metre en place un minimum et un maximum j'ai utilisé la formule suivante :

```
        return int(min + ((self.seed - 0) / (self.modulus - 0)) * (max - min))
```

On peut faire un liste de nombre aléatoire avec la formule lcg_list :

```
print(gen.lcg_List(10,0,7))
```

qui donne le résultat :

[0, 3, 4, 0, 4, 4, 1, 0, 2, 6]

Partie 2 : Test du générateur

On sait que lancer deux dés indépendamment, la somme des deux résultats sera :

```
expectedResult=[1/36, 1/18, 1/12, 1/9, 5/36, 1/6, 5/36, 1/9, 1/12, 1/18, 1/36]
```

en utilisant le test de KHI Deux à 5% avec 11 catégories

```
#Chi2 Calculé :

somme(Effectif Theorique - Effectif Reel)^2 / Effectif Théorique
```

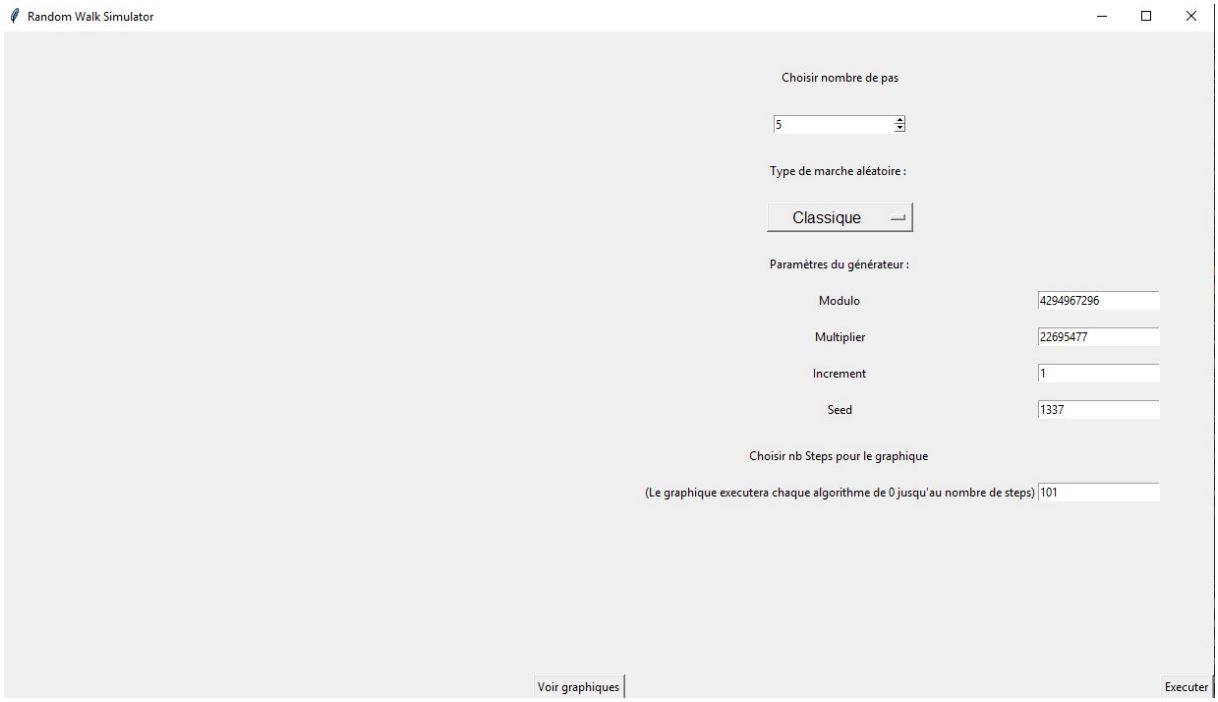
Nous avons la plupart du temps sur 1000 lancés indépendants en utilisant mon générateur de nombres pseudo aléatoires, des tests résultants un succès. On est sur que c'est indépendants car on ne recrée pas un nouvel objet Generator(), on utilise le même on utilise pas la même suite de nombres.

De temps à autres, nous pouvons voir une erreur comme tel :

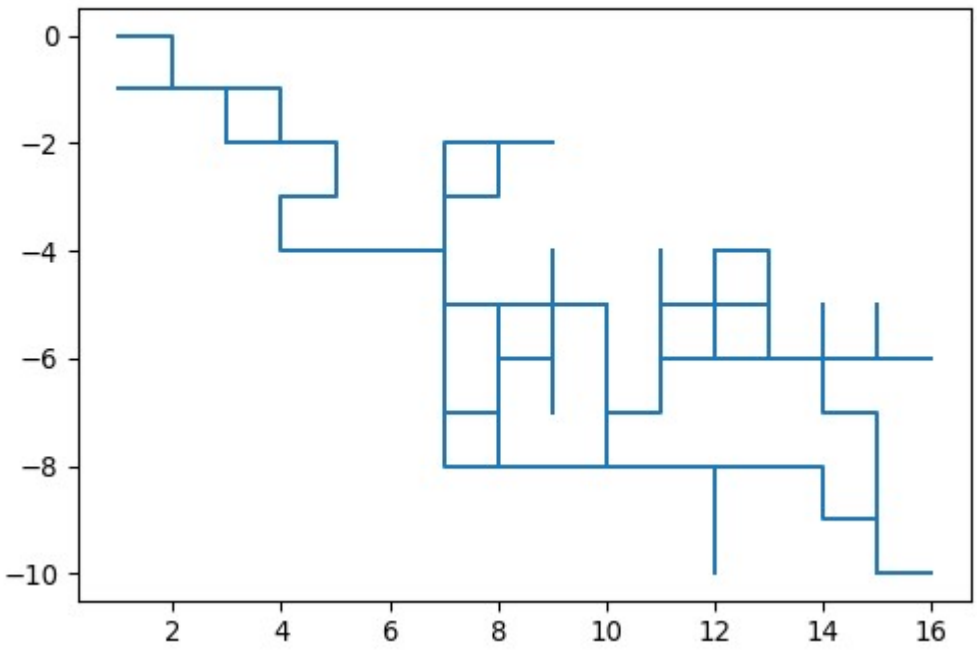
```
--------------------  TEST KHI DEUX INCORRECT  -------------------------
+-----------+------------------------+-----------------+--------------------+-------------+
| Iteration | Probabilités attendues | Nos probabilités |  Valeur attendues  | Nos valeurs |
+-----------+------------------------+-----------------+--------------------+-------------+
|     1     |  0.027777777777777776  |      0.027      | 27.777777777777775 |      27     |
|     2     |  0.05555555555555555   |      0.068      | 55.55555555555555  |      68     |
|     3     |  0.08333333333333333   |      0.085      | 83.33333333333333  |      85     |
|     4     |   0.1111111111111111   |      0.105      | 111.1111111111111  |     105     |
|     5     |   0.1388888888888889   |      0.123      | 138.88888888888889 |     123     |
|     6     |  0.16666666666666666   |      0.194      | 166.66666666666666 |     194     |
|     7     |   0.1388888888888889   |      0.124      | 138.88888888888889 |     124     |
|     8     |   0.1111111111111111   |      0.098      | 111.1111111111111  |      98     |
|     9     |  0.08333333333333333   |      0.102      | 83.33333333333333  |     102     |
|     10    |  0.05555555555555555   |      0.057      | 55.55555555555555  |      57     |
|     11    |  0.027777777777777776  |      0.017      | 27.777777777777775 |      17     |
|     1     |  0.027777777777777776  |      0.018      | 27.777777777777775 |      18     |
|     2     |  0.05555555555555555   |      0.058      | 55.55555555555555  |      58     |
|     3     |  0.08333333333333333   |      0.081      | 83.33333333333333  |      81     |
|     4     |   0.1111111111111111   |      0.146      | 111.1111111111111  |     146     |
|     5     |   0.1388888888888889   |      0.114      | 138.88888888888889 |     114     |
|     6     |  0.16666666666666666   |      0.191      | 166.66666666666666 |     191     |
|     7     |   0.1388888888888889   |      0.133      | 138.88888888888889 |     133     |
|     8     |   0.1111111111111111   |      0.104      | 111.1111111111111  |     104     |
|     9     |  0.08333333333333333   |      0.073      | 83.33333333333333  |      73     |
|     10    |  0.05555555555555555   |      0.061      | 55.55555555555555  |      61     |
|     11    |  0.027777777777777776  |      0.021      | 27.777777777777775 |      21     |
|     1     |  0.027777777777777776  |      0.035      | 27.777777777777775 |      35     |
|     2     |  0.05555555555555555   |      0.063      | 55.55555555555555  |      63     |
|     3     |  0.08333333333333333   |      0.092      | 83.33333333333333  |      92     |
|     4     |   0.1111111111111111   |      0.101      | 111.1111111111111  |     101     |
|     5     |   0.1388888888888889   |      0.134      | 138.88888888888889 |     134     |
|     6     |  0.16666666666666666   |      0.182      | 166.66666666666666 |     182     |
|     7     |   0.1388888888888889   |      0.135      | 138.88888888888889 |     135     |
|     8     |   0.1111111111111111   |      0.093      | 111.1111111111111  |      93     |
|     9     |  0.08333333333333333   |      0.064      | 83.33333333333333  |      64     |
|     10    |  0.05555555555555555   |      0.072      | 55.55555555555555  |      72     |
|     11    |  0.027777777777777776  |      0.029      | 27.777777777777775 |      29     |
+-----------+------------------------+-----------------+--------------------+-------------+
```

Partie 3 Utilisation du générateur : marche aléatoire

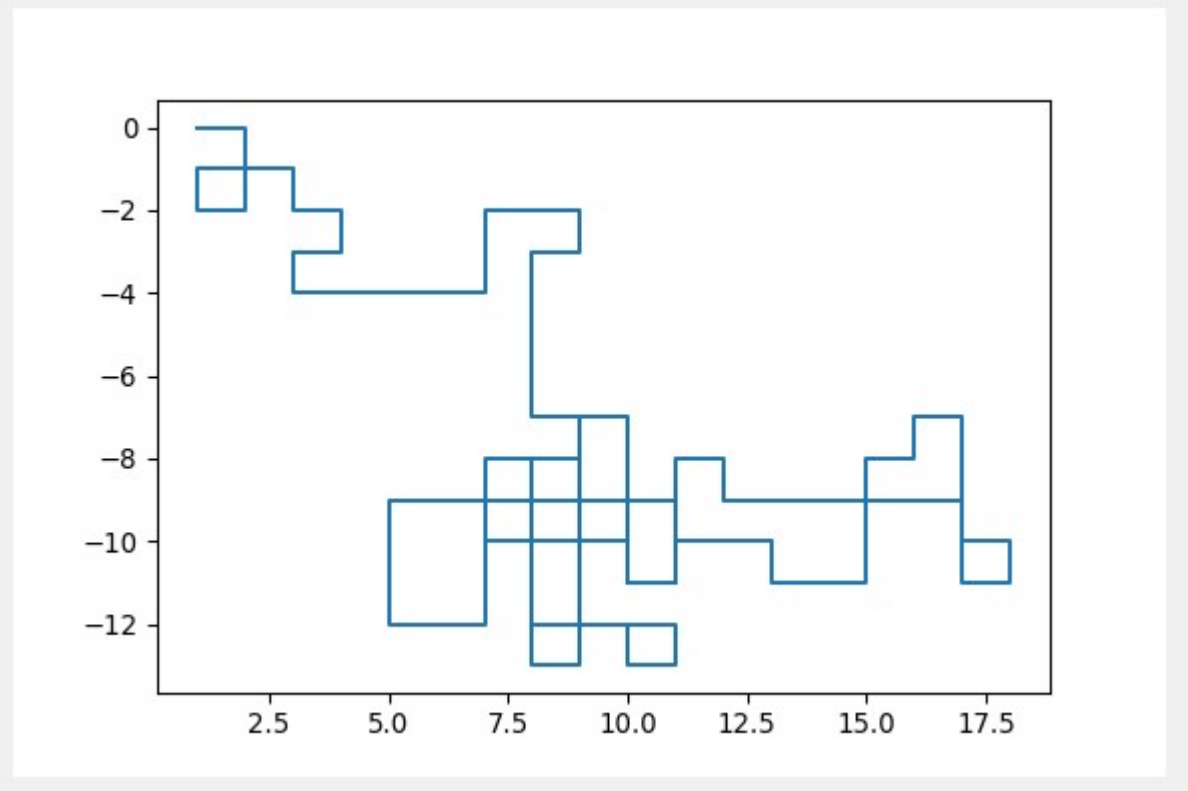
Pour cette partie, j'ai créé une IHM à l'aide de TKinter et Matplotlib pour la réalisation des différents graphiques.



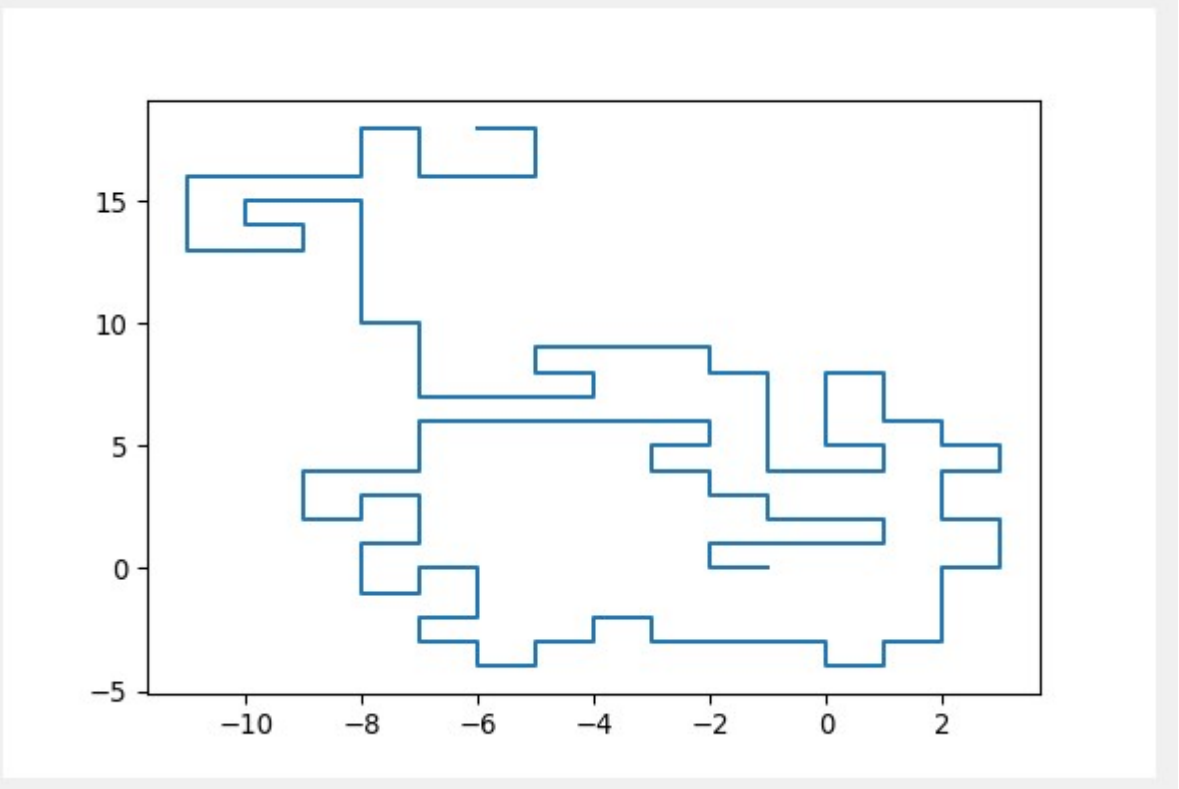
Voici le résultat pour 128 pas pour la marche aléatoire :



Marche aléatoire sans retour :



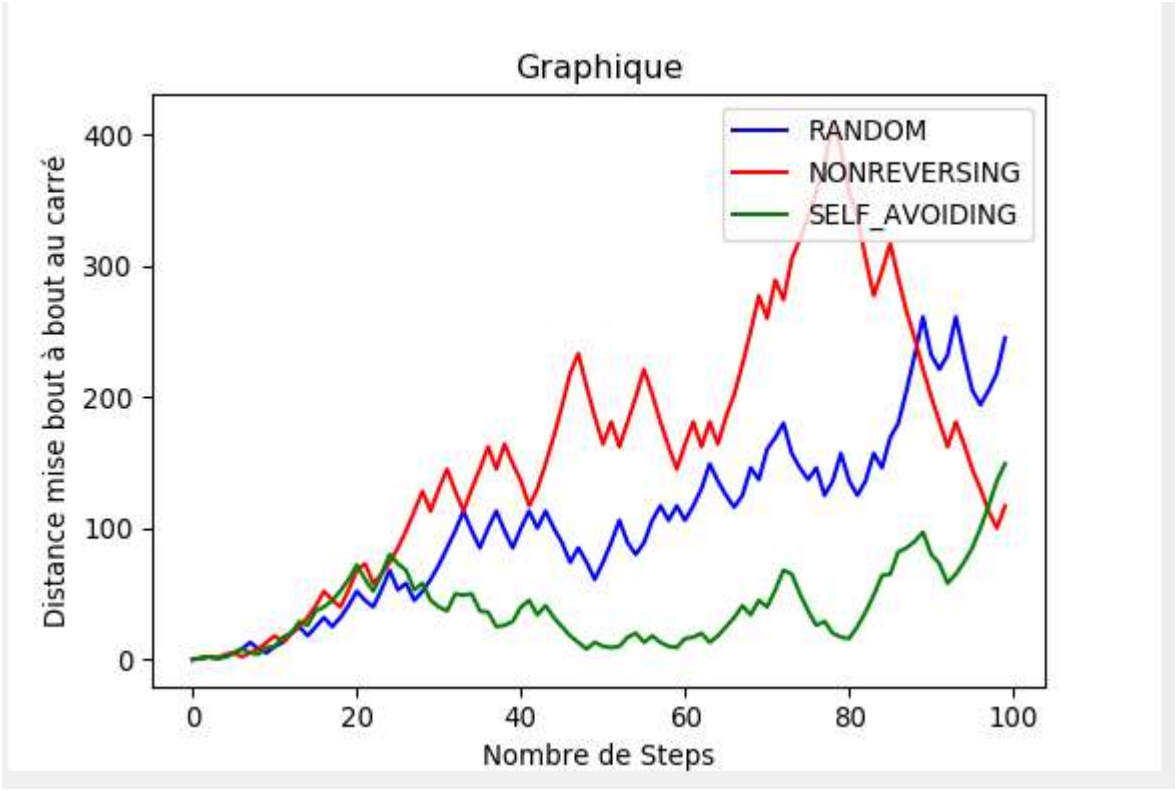
Marche aléatoire avec passage unique :

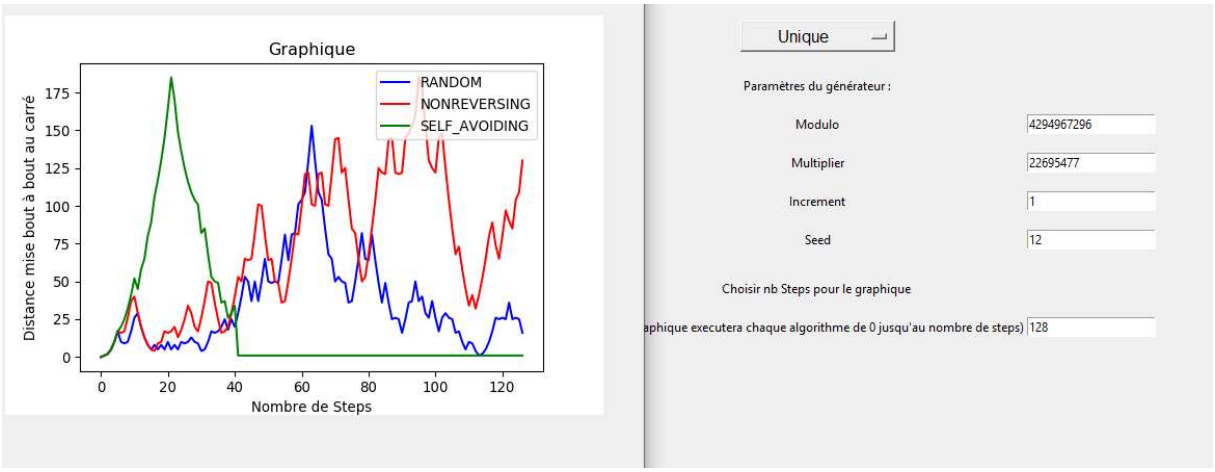


Avec les paramètres de ce générateur, le maximum est de 128 pas, sinon il est coincé.

On peut modifier les paramètres du générateur, le nombre de pas et le type d'algorithme que l'on utilise.

Pour chacun de ces paramètres, on peut générer un graphe en fonction de chacun de paramètres, il faut choisir ses paramètres et cliquer sur Voir graphiques :





Afin d'utiliser ce programme il faut :

Avoir les librairies suivantes :

```
from RandomNumberGenerator import Generator #Mon Générateur
from prettytable import PrettyTable # Utilisé dans RandomNumberGenerator
import tkinter as tk #Pour l'IHM
import math #pour tout les calculs
from itertools import combinations #pour tout les calculs
import numpy #utilisation d'array
```

```
import pylab
import matplotlib #Pour les graphiques
matplotlib.use('TkAgg')
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.pyplot as plt
```