Psi 999 – 2021/2022 ψ **TP Python**

Marches aléatoires

Samedi 29 janvier 2022

Buts du TP

— Réaliser et visualiser des marches aléatoires dans $\mathbb{Z}$, puis le plan, l'espace et même le cercle trigonométrique.
— Faire des statistiques sur de telles marches pour vérifier/tester des résultats plus ou moins standards.

Exercice 1. *Créer (au bon endroit) un dossier associé à ce TP. Y placer une copie du fichier récupéré dans le dossier partagé de travail de la classe :* `cadeau_marches.py`
Lancer Spyder/Pyzo/Idle, sauvegarder au bon endroit le fichier `tp_marches_ouuntruccommeça.py` *; écrire une commande absurde, de type* `print( 6*7 )` *dans l'éditeur, sauvegarder et exécuter. En profiter pour importer ce qu'il faut : voir l'entête du fichier-cadeau.*

Si vous êtes sous Spyder, changez (via F6) les options d'exécution, pour avoir à chaque exécution une nouvelle console et garder la main dessus. (il y a donc deux cases à vérifier/cocher). Si vous êtes sous Pyzo, la commande CTRL SHIFT S lui expliquera qu'il est prié de sauvegarder les fichiers dans le répertoire courant.

Exercice 2. *Vérifier que le premier exo a effectivement bien été fait : tout manquement donnera lieu de ma part à une agitation néfaste pour tout le monde...*

1 Marches dans $\mathbb{Z}$

On commence par des marches dans $\mathbb{Z}$ avec une fonction réalisant un pas. Elle prend en entrée $p \in [0,1]$, puis renvoie 1 avec probabilité p et -1 avec probabilité $1-p$. On utilisera la fonction `random` qui, appelée sans argument, renvoie un flottant entre 0 et 1, avec une loi de répartition uniforme, de sorte que `randint() <= p` est vrai avec probabilité p.

Exercice 3. *Écrire une telle fonction réalisant un pas.*

Une marche est obtenue en sommant des pas !

Exercice 4. *Écrire une fonction réalisant une marche. On donnera comme paramètres un nombre de pas et une probabilité (d'avancer à droite à chaque fois !).*

```
>>> pas(0.5)
1
>>> pas(0.5)
-1
>>> pas(0.5)
-1
>>> pas(0.8)
1
>>> pas(0.2)
-1
```

```
>>> marche(10, 0.5)
[0, -1, -2, -1, 0, -1, 0, -1, -2, -3, -2]
>>> marche(10, 0.5)
[0, -1, 0, 1, 2, 1, 0, 1, 2, 3, 2]
>>> marche(10, 0.5)
[0, 1, 0, 1, 0, -1, -2, -3, -2, -1, -2]

>>> marche(10, 0.3)
[0, -1, -2, -3, -2, -1, 0, -1, -2, -3, -4]
```

Dans le fichier-cadeau, quelques lignes de code sont fournies, permettant de visualiser des marches.

Exercice 5. *Faites un copier/coller du code concerné, exécutez et regardez les fichiers produits. Comprenez. Passez à la suite !*

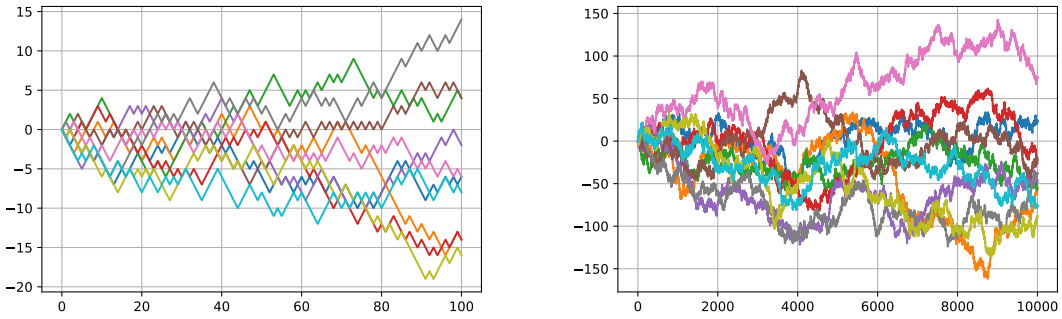

FIGURE 1 – Des marches aléatoires non biaisées...

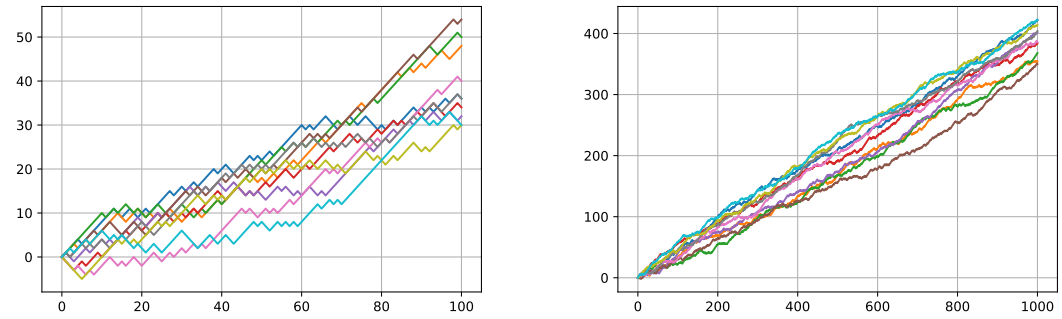

FIGURE 2 – Et des marches biaisées (devinez la valeur de p).

On va maintenant tester quelques résultats portant sur une marche $S_n = X_1 + \cdots + X_n$ (les X_i sont des pas indépendants) tels que (dans le cas non biaisé $p = 1/2$) :

- L'espérance de S_n est nulle.
- Sa variance est $\mathrm{Var}(S_n) = \mathbb{E}(S_n^2) = n$.
- L'espérance de sa valeur absolue est majorée par la racine de l'espérance de son carré : $\mathbb{E}(|S_n|) \leqslant n$.
- Avec probabilité 1, l'ivrogne retournera au pub. De même, avec probabilité 1 il repassera devant chez lui.

Cadeau : on donne une fonction permettant de calculer la moyenne des positions issues de marches successives (on réalise des marches, et on garde la dernière valeur : on fait la somme de toutes ces dernières valeurs avant de diviser par le nombre de marches).

```
def valeur_moyenne(nb_pas, p, nb_marches):
    return sum(marche(nb_pas, p)[-1] for _ in range(nb_marches)) / nb_marches
```

```
>>> valeur_moyenne(100, 0.5, 10)
0.0
>>> valeur_moyenne(100, 0.5, 10)
1.0
>>> valeur_moyenne(100, 0.5, 10)
-5.4
```

```
>>> valeur_moyenne(100, 0.5, 10**4)
0.1628
>>> valeur_moyenne(100, 0.7, 10**3)
39.87
>>> valeur_moyenne(100, 0.2, 10**3)
-59.858
```

Exercice 6. *Exécuter les tests tels que ceux vus plus haut. Expliquer les résultats.*

Exercice 7. *En s'inspirant de ce qui précède, évaluer les valeurs moyennes de* $|S_n|$ *et de* S_n^2 *pour* $n = 10$ *puis* $n = 100$ *(à vous de déterminer le nombre de marches qui vous semble pertinent).*

Ne passez pas à la suite avant d'avoir noté quelque part les résultats et réfléchi à leur vraisemblance, sinon ce TP ne sert à rien !

On s'intéresse maintenant à la question du passage par un point imposé : on fixe un objectif (un entier relatif, ici), et on réalise un certain nombre de marches pour savoir si on passe par cet objectif. La théorie dit qu'avec probabilité 1 on passe par cet objectif dans $\mathbb{Z}$ et $\mathbb{Z}^2$... mais pas $\mathbb{Z}^3$! Bien entendu si on s'impose un nombre de pas maximum, cette probabilité est < 1 (elle augmente avec le nombre de pas maximum, et diminue avec la distance entre l'objectif et l'origine).

Exercice 8. *Écrire une fonction réalisant une marche d'au plus N pas (N est un paramètre), retournant* `True` *si la marche est passée par l'objectif (paramètre de la fonction), et* `False` *sinon. On pourra prendre un troisième paramètre $p \in [0, 1]$ permettant de biaiser la marche (sans quoi, on fait des marches symétriques, ce qui est déjà intéressant).*

```
>>> repasse_par(0, 10, 0.5)
True
>>> repasse_par(0, 10, 0.5)
True
>>> repasse_par(0, 10, 0.5)
False
>>> repasse_par(0, 10, 0.5)
True
```

```
>>> repasse_par(0, 10, 0.7)
False
>>> repasse_par(0, 10, 0.7)
True
>>> repasse_par(0, 10, 0.7)
False
>>> repasse_par(0, 10, 0.7)
False
>>> repasse_par(0, 10, 0.7)
False
```

Exercice 9. *Écrire une fonction réalisant des statistiques sur la question du passage par un objectif (on donne comme paramètre supplémentaire le nombre de marches)*

```
>>> proba_passage(0, 2, 10**4, 0.5)
0.4978
>>> proba_passage(0, 3, 10**4, 0.5)
0.5021
>>> proba_passage(0, 4, 10**5, 0.5)
0.62563

>>> proba_passage(0, 10, 10**5, 0.5)
0.7553
>>> proba_passage(0, 100, 10**5, 0.5)
0.91979
>>> proba_passage(0, 10**4, 10**4, 0.5)
0.9913

>>> proba_passage(-5, 4, 10**5, 0.5)
0.0
>>> proba_passage(-5, 10, 10**5, 0.5)
0.10912
```

```
>>> proba_passage(-5, 100, 10**5, 0.5)
0.61873
>>> proba_passage(-5, 10**4, 10**4, 0.5)
0.9637

>>> proba_passage(-5, 10, 10**5, 0.8)
0.00071
>>> proba_passage(-5, 10, 10**5, 0.2)
0.75681
>>> proba_passage(-5, 100, 10**5, 0.8)
0.00107
>>> proba_passage(-5, 100, 10**5, 0.2)
1.0
>>> proba_passage(-5, 10**4, 10**3, 0.8)
0.001
>>> proba_passage(-5, 10**4, 10**3, 0.2)
1.0
```

Ici encore, ne passez pas à la suite avant d'avoir réfléchi à la pertinence/vraisemblance des résultats !

2 Dans le plan, puis l'espace

On passe dans le plan ! Cette fois, un pas est un couple (ou une liste de deux) entiers : il y a 4 déplacements équiprobables.

Exercice 10. *Écrire une fonction réalisant un pas dans le plan.*

On pourra choisir un pas dans la liste des quatre possibles, grâce à la fonction `randint` qui prend en entrée deux arguments entiers, disons a et b, et renvoie un entier n tel que $a \leqslant n < b$ (oui, une inégalité stricte à droite), chacun avec probabilité $\dfrac{1}{b-a}$.

```
>>> randint(0, 4)
3
>>> randint(0, 4)
2
>>> randint(0, 4)
3
>>> randint(0, 4)
0
```

```
>>> pas2()
[0, 1]
>>> pas2()
[-1, 0]
```

Pour réaliser puis représenter une marche dans le plan, allez voir du coté du fichier-cadeau !

Exercice 11. *Copier/coller le code concernant la marche dans le plan, puis le dessin d'une telle marche. Comprendre, exécuter (dans l'ordre que vous voulez !).*

```
>>> marche2(5)
([0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 2, 1])
```

```
>>> marche2(10)
([0, -1, -2, -3, -4, -3, -2, -1, 0, -1, -2],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

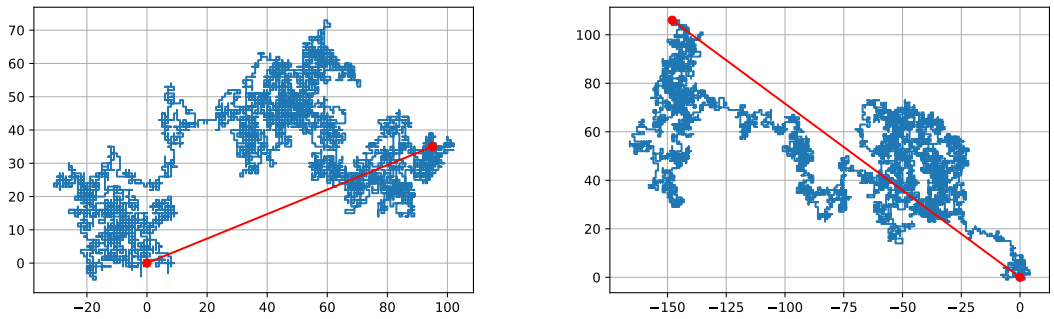

FIGURE 3 – Des marches (de 10^4 pas) dans le plan – j'ai relié l'origine à l'arrivée.

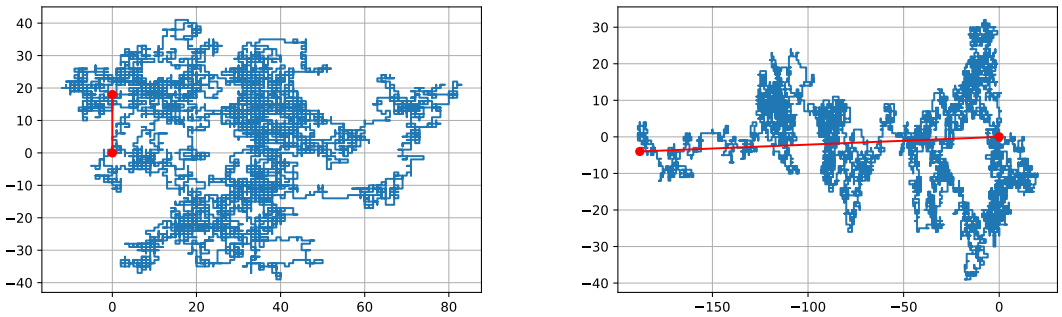

FIGURE 4 – Et deux de plus

Vous pouvez voir qu'il y a une grande variété de scénarios possibles ; en particulier sur la question du retour vers l'origine/le pub !

Exercice 12. *Écrire des fonctions permettant d'évaluer la distance moyenne (resp. : au carré) après une marche de n pas.*

```
>>> distance_moyenne2(10, 10**4)
2.7972986400949882
>>> distance_carre_moyenne2(10, 10**4)
9.9152
```

```
>>> distance_moyenne2(100, 10**4)
8.8857170582871934
>>> distance_carre_moyenne2(100, 10**4)
99.4198
```

On s'intéresse enfin à la question du retour au pub.

Exercice 13. *Écrire des fonctions permettant d'évaluer la probabilité pour qu'une marche (d'une longueur imposée) atteigne un point donné.*

```
>>> proba_passage2((0, 0), 2, 10**4)
0.2543

>>> proba_passage2((0, 0), 10, 10**4)
0.4198
>>> proba_passage2((0, 0), 10, 10**6)
0.420706

>>> proba_passage2((0, 0), 100, 10**4)
0.5808
>>> proba_passage2((0, 0), 1000, 10**3)
0.662
```

```
>>> proba_passage2((-2, 0), 100, 10**3)
0.424
>>> proba_passage2((-2, 0), 1000, 10**3)
0.544

>>> proba_passage2((-5, 0), 100, 10**3)
0.147
>>> proba_passage2((-5, 0), 1000, 10**3)
0.349
```

Et maintenant, dans l'espace ! Ici il y a un grand changement : la probabilité pour que l'ivrogne repasse par le bar est strictement plus petite que 1.

Exercice 14. *Reprendre les questions précédentes (distance (au carré) moyenne, probabilité de passage par l'origine), et tester !*

```
>>> distance_moyenne3(10, 10**4)
2.9089227885883364
>>> distance_carre_moyenne3(10, 10**4)
9.9872

>>> distance_moyenne3(100, 10**4)
9.1756848430914975
>>> distance_carre_moyenne3(100, 10**4)
98.3932
```

```
>>> proba_passage3((0, 0, 0), 2, 10**4)
0.1651

>>> proba_passage3((0, 0, 0), 10, 10**4)
0.251
>>> proba_passage3((0, 0, 0), 10, 10**6)
0.252899

>>> proba_passage3((0, 0, 0), 100, 10**4)
0.3134
>>> proba_passage3((0, 0, 0), 1000, 10**4)
0.3309
>>> proba_passage3((0, 0, 0), 10**4, 10**2)
0.34
>>> proba_passage3((0, 0, 0), 10**5, 10**2)
0.35
```

3 Sur le cercle

On termine avec l'ivrogne trigonométrique, qui se déplace d'un angle $\pm\alpha$ à chaque étape – on part du point de coordonnées $(1, 0)$.

Exercice 15. *Écrire une fonction renvoyant une marche sur le cercle trigonométrique. Les paramètres sont : l'angle de chaque pas, et le nombre de pas. Le résultat renvoyé est constitué des deux listes de coordonnées.*

```
>>> marche_trigo(5, pi/10)
(array([ 1.,  0.95,  0.81,  0.95,  1., 0.95]), array([ 0., -0.31, -0.59, -0.31,  0., -0.31]))
```

La théorie (exercice !) dit que la position moyenne (enfin, son espérance...) après n pas est $(\cos^n \alpha, 0)$.

Exercice 16. *Écrire une fonction calculant la position moyenne après n pas, sur un certain nombre de marches. Comparer à la théorie.*

```
def moyenne_trigo(nb_pas, nb_marches, alpha):
    ...
    return sx/nb_marches, sy/nb_marches
```

```
>>> moyenne_trigo(10, 10**5, pi/10)
(0.60663483107781901, 0.0036916962726706117)
```

```
def moyenne_theo(nb_pas, alpha):
    return cos(alpha)**nb_pas
```

```
>>> moyenne_theo(10, pi/10)
0.60542904971310629
```