Code de partage avec Capytale : e5f7-1512163

Corrigé

Echauffement

Emettre une conjecture sur la valeur de $S = \sum_{n=1}^{+\infty} \frac{1}{n^2}$

On pourra multiplier la série par 6*, puis prendre la racine du nombre conjecturé.*

Python nous permet de calculer des grandes valeurs de $S_n = \sum_{k=1}^{n} \frac{1}{k^2}$ et d'émettre une hypothèse à la fois sur la convergence (on le sait déjà ici) d'une série et sur la valeur de sa limite (sa somme).

Option 1 : la boucle `for` est tout à fait indiquée pour calculer de manière itérative les termes d'une suite (et donc d'une somme), puisqu'ici $S_1 = 1$ et $\forall n \in \mathbb{N}^*, S_{n+1} = S_n + \frac{1}{(n+1)^2}$. On calcule S_{100} avec la commande ci-dessous. On peut aussi créer une fonction `somme` qui nous permet de calculer S_n pour n'importe quelle valeur de n :

```
s=0
for n in range(1,1001):
    s=s+1/n**2

print(s)
```

Option 2 : la liste

Par exemple, la commande `sum([1/n**2 for n in range(1,101)])` nous renvoie d'emblée S_{100}. On peut aussi en faire une fonction ... puis éventuellement représenter les 100 premiers termes avec `x=np.linspace(1,100,100)` puis `plt.plot(x,somme(x))` . . .

```
def somme(n):
  return sum([1/i**2 for
    i in range(1,n)])
```

Dans tous les cas, on arrive à la conjecture que la série converge vers un nombre proche de $1,64$. Et il se trouve que $\sqrt{6 \times 1,64} \simeq 3,14$ donc on peut supposer que la série converge vers $\frac{\pi^2}{6}$, ce qui est le cas.

Variables aléatoires

Exercice 1 - marche aléatoire

Une puce se déplace sur un axe gradué. A l'instant $t = 0$, la puce se trouve à l'origine (point d'abscisse 0). Puis entre un instant $t = n$, et l'instant $t = n + 1$, la puce se déplace de 1 vers la droite ou de 1 vers la gauche avec équiprobabilité. On arrête l'observation à l'instant $t = 200$

1. Ecrire un programme qui renvoie 1 ou -1 avec équiprobabilité.

   On modélise l'aléa avec `rd.randint(0,1)` (ici 2 valeurs, comme un tirage à pile ou face) et on créée une fonction pour renvoyer les valeurs souhaitées : -1 et 1

   ```
   import numpy.random as rd

   def saut():
       a=rd.randint(0,2)
       if a==0 :
           return -1
       else :
           return 1
   ```

2. Compléter le programme précédent, pour qu'il crée un vecteur x de taille 201 tel que $\forall n \in [\![0, 200]\!]$, $x(n)$ donne l'abscisse de la puce à l'instant $t = n$

   On fait sauter la puce 200 fois avec une boule `for` et on créée une variable position pour suivre son déplacement que l'on enregistre dans une liste `x`

   ```
   x=[0]
   position=0

   for i in range(1,201):
       position=position+saut()
       x.append(position)
   ```

3. Faire afficher l'abscisse maximale/minimale de la puce ainsi que l'abscisse moyenne. On pourra exécuter plusieurs fois le programme, et comparer les résultats.

   On utilise les fonctions `min` et `max` que l'on applique à `x`. Pour la moyenne il faut importer la bibliothèque `numpy` puis `np.mean(x)`.

Exercice 2

On lance une pièce équilibrée et on note Z la variable aléatoire égale au rang du lancer où l'on obtient le premier « pile ». Puis si Z a pris la valeur k ($k \in \mathbb{N}^*$), on remplit une urne de k boules numérotées $1, 2, \ldots, k$, et on extrait au hasard une boule de cette urne. Soit X la variable aléatoire égale au numéro de la boule tirée. Ecrire un programme Python qui simule l'expérience ci-dessus, et affiche les valeurs de Z et X

Nous avons deux tirages aléatoires à modéliser, tout d'abord le rang d'apparition du premier pile qui correspond à la simulation d'une loi géométrique (nous le verrons bientôt). Il s'agit simplement de « lancer la pièce » tant que face n'est pas obtenu. Puis, le tirage d'un nombre aléatoire parmi une plage d'entiers consécutifs, ce que la fonction `randint` fait très bien.

```
import numpy.random as rd
Z=1 #compteur de lancers, on commence au premier
while rd.random()<1/2 : # on considère qu'on obtient face si le nombre est entre
   0 et 1/2, et pile sinon
  Z=Z+1 #on passe au lancer suivant
```

```
X=rd.randint(1,Z+1) #tirage aléatoire d'un nombre entre 1 et Z
print(Z,X)
```

Alternative en utilisant la fonction rd.geometric (avec le paramètre $\frac{1}{2}$, qui fait la même chose que ce que nous avaons fait « à la main » ci-dessus.

```
import numpy.random as rd
Z=rd.geometric(0.5) #simulation d'une variable aléatoire suivant une loi
    géométrique de paramètre 0,5
X=rd.randint(1,Z+1) #tirage aléatoire d'un nombre entre 1 et Z
print(Z,X)
```