Code de partage avec Capytale : 8172-1114683

Corrigé

Echauffement

Exercice 1 - chifoumi

Importer `numpy.random` (avec l'alias `rd`) et tester la fonction `rd.randint`

Après avoir exécuté `import numpy.random as rd`, on peut par exemple utiliser la commande `rd.randint(1,7)` qui génère des entiers de manière aléatoire (ici des entiers entre 1 et 6, attention donc le 7 est exclu). Ecrire une fonction qui renvoie `pierre` ou `feuille` ou `ciseaux` avec la même probabilité (une chance sur trois dans chaque cas).

Une des clés du problème est de générer 3 valeurs qui sortent de manière aléatoire et équiprobable. C'est ce que permet `rd.randint(0,2)` qui génère des entiers de manière aléatoire (ici des entiers entre 0 et 2). La liste permet de faciliter l'écriture du résultat : pour chaque nombre i entre 0 et 2 généré de manière aléatoire, la fonction renvoie le $i^{\text{ème}}$ élément de la liste.

```
import random
def chifoumi() :
    L=["pierre","feuille","
   ciseaux"]
    i=random.randint(0,2)
    return L[i]
```

Après tirage aléatoire, on peut également utiliser des conditions : « si $i = 0$ alors la fonction renvoie "pierre" »...

Ecrire un programme qui simule une partie entre deux joueurs.

On fait jouer deux joueurs `a` et `b`. On gère les cas d'égalité avec la boucle `while` : on fait rejouer les joueurs tant qu'il y a égalité. Puis il faut gérer tous les cas : ici tous les cas où le joueur `a` gagne et le reste (ce qui correspond aux cas où le joueur `b` gagne).

```
a=chifoumi()
b=chifoumi()
while a==b :
    a=chifoumi()
    b=chifoumi()
if (a=='ciseaux' and b=='feuille') or  (a=='pierre' and b=='ciseaux') or (a=='
   feuille' and b=='pierre') :
    print("le joueur a a gagné")
    print(a,b)
else :
    print("le joueur b a gagné")
    print(a,b)
```

Test

30 minutes

Exercice 1

Soit $(u_n)_{n\in\mathbb{N}}$ une suite définie par $u_0 = 0$ et $\forall n \in \mathbb{N}, u_{n+1} = 3u_n + 2$

1. Ecrire un programme qui créée une liste contenant les 100 premiers termes de la suite après u_0 (de u_1 à u_{100}).

   On calcule les termes de manière itérative avec une boucle `for` (on pourrait aussi trouver la formule explicite de la suite puisqu'il s'agit d'une suite arithmético-géométrique). Pour en faire une liste, on commence par créer une liste vide puis on complète la liste à chaque itération avec `append`.

```
u=0
L=[]
for i in range(1,101):
    u=3*u+2
    L.append(u)
```

2. Représenter graphiquement cette suite (avec Python).

   Une fois la liste des valeurs de la suite créée (ici dans `L`, il suffit de la représenter avec `plot` (en ayant préalablement importé la librairie nécessaire à la représentation graphique). On ajoute le `+` pour avoir un nuage de points (et non une ligne brisée).

```
import matplotlib.
   pyplot as plt
plt.plot(L,'+')
plt.show()
```

3. Ecrire un programme qui trouve la première valeur de n pour laquelle $u_n \geqslant 50\,000$. Le programme affichera n et u_n comme résultat.

   On peut exploiter à nouveau la liste `L` si on a remarqué que les dernières valeurs dépassent (largement) 50 000. On part du premier terme de la suite et on passe au suivant tant que la valeur est strictement inférieure à 50 000. Pour visualiser le résultat, on affiche à l'issue de la boucle, la valeur de l'indice et la valeur du terme de la suite correspondant.

```
n=0
while L[n]<50000 :
    n=n+1

print(n, L[n])
```

Exercice 2 - fonction

Soit f la fonction définie sur $\mathbb{R}$ par $f(x) = \sin(x)$ (sinus de x)

1. Définir la fonction f (avec Python).

   Il faut simplement importer la bibliothèque `numpy` pour accéder à la fonction `sinus`.

```
import numpy as np
def f(x) :
    return np.sin(x)
```

2. Représenter graphiquement la fonction sur l'intervalle $[-100, 100]$

   Après la défintion de la fonction, il faut choisir une liste d'abscisses, ce que l'on fait avec `linspace` (il faut disposer de la bibliothèque `numpy` pour l'utiliser), ici on choisit 1 000 abscisses entre -100 et 100. Puis on complète avec les commandes habituelles de représentation.

```
import matplotlib.pyplot as plt
x=np.linspace(-100,100,1000)
y=f(x)
plt.plot(x,y)
plt.show()
```

Exercice 3

Etant donnée la liste aléatoire suivante et à la suite de sa définition, écrire un programme qui donne le rang d'apparition du premier 2

```
import random
L=[random.randint(1,6) for i in range(1,101)]
```

En commençant par la première, il faut examiner les valeurs une par une, jusqu'à trouver un 2. La boucle `while` est l'outil indiqué pour cela.

```
i=0
while L[i]!=2:
    i=i+1
print(i,L[i]) # pour visualiser le résultat
```