02_cours_random

May 11, 2025

1 Suite du cours d'introduction à Python

Thème : Utiliser le hasard avec le module `random`, les boucles et les conditions

1.1 Bienvenue

- **Introduction à Python (Partie 2)**
- Diapos PowerPoint – Suite du cours d'introduction à Python
- Thème : Créer des petits jeux de hasard avec le module `random`

1.2 Objectifs de cette partie

À la fin de cette session, vous saurez : - Comprendre pourquoi le hasard est important dans les jeux - Utiliser le module `random` pour simuler des événements aléatoires - Tirer au sort des nombres, des éléments d'une liste, mélanger des données - Utiliser les boucles `for` et conditions `if...elif...else` - Créer de petits jeux basés sur le hasard

1.3 Pourquoi le hasard dans les jeux ?

- Le hasard rend les jeux :
  - Imprévisibles
  - Réjouissants
  - Équitables
- Exemples de jeux utilisant le hasard :
  - Jeux de dés
  - Jeux de cartes
  - Loto, bingo, etc.

1.4 Hasard "vrai" vs "pseudo-aléatoire"

- En informatique, le **vrai hasard n'existe pas**.
- On utilise des **nombres pseudo-aléatoires** générés par des algorithmes complexes.
- Python utilise une fonction mathématique + un germe (`seed`) pour produire ces séquences.

1.5 Le module `random`

Pour utiliser le module `random`, il faut l'importer :

```
from random import *
```

Ou importer uniquement ce dont on a besoin :

```
from random import randint, choice, shuffle, sample
```

1.6 Fonction `random()`

- Génère un nombre décimal entre 0 (inclus) et 1 (exclu).
- Exemple :

```
from random import random
print(random())  # ex: affiche 0.87965...
```

1.7 Fonction `randint(a, b)`

- Retourne un entier entre `a` et `b` inclus.
- Idéal pour simuler des dés :

```
from random import randint
print(randint(1, 6))  # Simule un dé à 6 faces
```

1.8 Boucle `for x in range(n)`

Pour répéter une action plusieurs fois (ici n fois)

```
for i in range(10):
    print(randint(1, 6), end=" ")
```

Affiche 10 jets de dés comme :

```
3 5 6 1 6 6 4 1 3 1
```

x variable qui varie de 0 à (n-1), 9 dans l'exemple

1.9 Fonction `choice(liste)`

Permet de choisir un élément au hasard dans une liste :

```
from random import choice
cartes = ['As', 'Roi', 'Dame', 'Valet']
print(choice(cartes))  # ex: 'Roi'
```

1.10 Fonction `shuffle(liste)`

Mélange les éléments d'une liste :

```
from random import shuffle
paquet = ['Cœur', 'Carreau', 'Trèfle', 'Pique']
shuffle(paquet)
print(paquet)  # ex: ['Trèfle', 'Pique', 'Cœur', 'Carreau']
```

1.11 Fonction `sample(liste, k)`

Tire `k` éléments uniques d'une liste sans répétition :

```
from random import sample
numeros = list(range(1, 51))
```

```
tirage = sample(numeros, 5)
print(tirage)  # ex: [3, 12, 27, 39, 45]
```

1.12 Exercice : Simuler un jeu de dé pipé

- Probabilités personnalisées :
  - 1 → 10%
  - 2,3,4,5 → 15%
  - 6 → 30%

Utiliser `random()` et `if...elif...else` :

```
from random import random

rnd = random()
if rnd < 0.1:
    print(1)
elif rnd < 0.25:
    print(2)
elif rnd < 0.4:
    print(3)
elif rnd < 0.55:
    print(4)
elif rnd < 0.7:
    print(5)
else:
    print(6)
```

1.13 Opérateurs de comparaison

Opérateur	Signification
`<`	inférieur à
`>`	supérieur à
`<=`	inférieur ou égal
`>=`	supérieur ou égal
`==`	égal à
`!=`	différent de

Ne pas confondre `=` (affectation) et `==` (comparaison)

1.14 Mini-Projets possibles

1.14.1 A réaliser en solo ou en binôme

1. Un générateur de mot de passe aléatoire
2. Un simulateur de tirage Euro Millions
3. Une simulation de jeu de bingo (1 à 90)
4. Mélanger l'ordre d'entrée des catcheurs (Royal Rumble)

1.15 Points clés de cette partie

Python ne produit pas de vrai hasard mais des nombres pseudo-aléatoires
Le module `random` permet de gérer le hasard facilement
`randint`, `choice`, `shuffle`, `sample` sont très utiles
Les boucles `for` permettent de répéter des actions
Les conditions `if...elif...else` permettent de prendre des décisions selon le hasard

1.16 Questions & Discussions

- Ouverture aux questions des participants
- Revue rapide des difficultés rencontrées

1.17 À suivre...

1.17.1 Prochain chapitre : Structures de données avancées

- Listes, tuples, dictionnaires
- Manipulation de données complexes
- Jeux plus évolués