Méthologie de la Programmation
TP2 : configuration éditeur de texte
premiers programmes en python

am@up8.edu

Septembre 2022

Dans ce TP :
- Choix et configuration de l'éditeur de texte.
- Écriture du code en python (documentation : `https://docs.python.org/3/`)
- Dépôt sur le moodle (cours Info-L1-MdP, clé d'inscription PROGRAMMATION) avant mardi prochain 13h d'une archive contenant uniquement des fichiers ".py" correspondant aux programmes des différents exercices.
- **notation** :
  - le **nommage** des fichiers, des fonctions, et des variables sera pris en compte dans la notation.
  - votre programme doit comporter une fonction `main` (voir exemple) dans lequel vous effectuez des appels aux fonctions développées pour **tester et montrer leur bon fonctionnement**. Pensez à tester les cas limites.
  - rendre le TP avant la date spécifiée sur le moodle ne donne pas de bonus.
  - pas de rendu sur moodle $\Rightarrow 0$.

Exercice 0

Configurez si cela n'est pas déjà fait un éditeur de texte pour python (coloration syntaxtique et indentation) :
- pour mousepad, consultez `https://doc.ubuntu-fr.org/mousepad`.
- pour gedit, allez dans l'option Affichage > Mode de coloration et choisissez « python »

Dans les deux cas, configurez la largeur des tabulations à '4'. (Préférences > Éditeur)

Ouvrez un document exemple_rendu.py avec l'éditeur de votre choix, recopiez le code ci-dessous et vérifiez que la coloration syntaxique correspond aux mots clé de python.

```
1 # -*- coding: utf-8 -*-
2
3 def multiplie_par_2(n):
4         return n*2
5
6 if __name__ == "__main__":
7     valeur=300
8     resultat=multiplie_par_2(valeur)
```

```
9     print(resultat)
```

Exercice 1 : boucle et test conditionnels

Écrire un programme qui affiche les nombres de 0 à 100, mais remplace les nombres divisibles par 3 par « Trois », ceux divisibles par 5 par « Cinq », et ceux qui le sont par 3 et par 5 par « Trois et Cinq ».

Exercice 3 : Un petit jeu

Écrire une fonction qui (i) prend un entier **saisi de manière interactive par l'utilisateur du programme** en argument, (ii) choisit un nombre aléatoire entre 0 et cet argument, et (iii) le fait deviner au joueur ou à la joueuse en lui donnant un indice après chaque tentative.

(i) Pour lire une entrée clavier, vous pouvez utiliser la fonction `input()`, qui prend en argument une chaîne de caractères à afficher, et qui retournera ce que la personne a saisi.

Par exemple `nom = input("Nom : ")` affiche "Nom : " et stocke dans la variable `nom` le texte entré au clavier.

Pour lire un entier, il faut convertir l'entrée clavier avec la fonction `int()`. On peut utiliser cette fonction directement sur la valeur renvoyée par `input` : `int(input("Âge : "))`.

Pour au contraire manipuler un nombre comme une chaîne de caractères (par exemple pour le concaténer avec l'opérateur `+` a une autre chaîne de caractères), on peut utiliser la fonction `str()`.

(ii) Pour générer un nombre aléatoire entre `a` et `b`, vous avez besoin de la fonction `random.randint(a, b)` qui a été codée dans le module `random` (voir `https://github.com/python/cpython/blob/3.10/Lib/random.py`).

Un **module** est un ensemble de fonctions, classes et variables prédéfinies que vous pouvez importer dans votre code grâce à la directive `import random` .

1. Créer un programme Python `guessing_game.py` qui contient la fonction `guess_the_number()` qui fait deviner le nombre au joueur.
2. Quand le nombre est correctement deviné par le joueur ou la joueuse, on veut lui afficher le nombre de coups que ce·tte dernièr·e a dû utiliser.

   Modifier la fonction `guess_the_number` en conséquence, et lui faire retourner cette valeur.
3. On souhaite proposer au joueur ou à la joueuse de faire plusieurs parties. En dehors de la fonction, on veut que le programme demande le nombre maximum à faire deviner, si celui ci est 0 alors il s'arrête, sinon il lance le jeu (c'est à dire appelle la fonction guess_the_number) avec ce nombre comme maximum.

   Modifier le programme `guessing_game.py` pour implémenter ce comportement.
4. Maintenant, on voudrait afficher un résumé des parties quand le jeu est quitté : pour chaque partie jouée, on veut afficher le nombre maximum à deviner et combien de coups il a fallu pour trouver le bon nombre.

   En utilisant une liste (un élément par partie) de dictionnaires (avec une entrée pour le maximum et une pour le nombre de coups), implémenter le comportement souhaité dans le programme `guessing_game.py`.

Exercice 3 : Coder Fibonacci

Écrire une fonction retournant la liste des n premiers nombres de Fibonacci sachant que :

- $F_0 = 0$
- $F_1 = 1$
- $F_n = F_{n-1} + F_{n-2}$