Soutien en algorithmique et programmation

Séance 1 : premier programme et programmation modulaire

Premier programme

On va commencer par écrire, exécuter et vérifier un programme très simple qui demande son nom à l'utilisateur et affiche un message de bienvenue :

- créez un fichier `hello.py` dans un répertoire, à l'aide de votre éditeur de texte préféré : on recommande d'utiliser VSCode ou Atom pour écrire du Python, mais si vous êtes déjà habitués à utiliser un autre éditeur de texte, cela ne pose pas de problème ;
- écrivez le *shebang* `#!/usr/bin/env python3` sur la première ligne de ce fichier : on rappelle que cela permet d'indiquer au système où trouver l'interprête Python qui va exécuter notre programme ;
- écrivez une fonction `main` qui doit d'abord demander son nom à l'utilisateur et le stocker dans une variable, puis afficher un message du type "Bonjour Toto !" où Toto doit être le nom de l'utilisateur ;
- n'oubliez pas d'appeler la fonction `main` tout en bas de votre programme, sinon rien ne sera exécuté !
- et n'oubliez pas non-plus de bien sauvegarder votre programme dans votre éditeur... on recommande de prendre l'habitude de sauvegarder très régulièrement ses programmes pour ne pas risquer de les perdre en cas de plantage de la machine ;
- dans un terminal, rendez le programme `hello.py` exécutable grâce à la commande Unix `chmod u+x hello.py` ;
- exécutez le programme dans votre terminal à l'aide de la commande `./hello.py`.

Le programme devrait s'exécuter sans erreur, mais ce n'est pas suffisant : il faut également vérifier que l'on a écrit du code Python de bonne qualité. On va utiliser pour cela l'outil `pylint` :

- tapez dans votre terminal la commande `pylint --reports=no ./hello.py` ;
- si tout est parfait, on doit obtenir un message du type `Your code has been rated at 10.00/10` ;
- sinon, le logiciel nous donne des pistes pour améliorer notre code : par exemple, un problème courant est d'oublier d'ajouter un commentaire en haut du programme (juste sous le *shebang*) pour décrire ce que fait le programme ; dans ce cas, `pylint` nous donnera un message du type : `hello.py:1:0: C0114: Missing module docstring (missing-module-docstring)` qui nous indique où se situe le problème (nom du fichier:numéro de ligne:numéro de colonne) et le type de problèmes ;
- améliorez votre programme jusqu'à obtenir 10/10 à l'exécution de `pylint`.

Eléments de correction

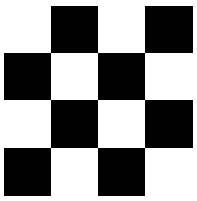
```
#!/usr/bin/env python3
"""
Premier programme
"""

def main():
"""
Fonction principale
"""
mon_nom = input("Entrez votre nom : ")
print(f"Bonjour {mon_nom} !")

main()
```

Programmation modulaire et paramétrée

On va écrire un programme qui dessine un échiquier 4x4 en utilisant le format SVG que vous avez vu en TP. Pour se simplifier la vie, vu que le format SVG est très verbeux, on fournit un module qui va implanter les fonctions dont on aura besoin, que vous pouvez copier-coller dans un fichier `mod_svg.py` :



```
def debut(hauteur, largeur):
    """
      Affiche le code de debut d'un fichier SVG
    """
    print(f"<svg xmlns=\"http://www.w3.org/2000/svg\" width=\"{largeur}\" height=\"{hauteur}\">")

def fin():
    """
      Affiche le code de fin d'un fichier SVG
    """
    print("</svg>")

def rect(abs1, ord1, larg, haut, coul):
    """
      Affiche un rectangle en SVG
    """
    print(f"  <rect x=\"{abs1}\" y=\"{ord1}\" width=\"{larg}\" height=\"{haut}\" fill=\"{coul}\"/>")
```

On va écrire un programme `echiquier.py` utilisant notre module pour générer un fichier SVG contenant le dessin de l'échiquer. Dans cet exercice, on supposera qu'**on ne sait pas encore utiliser les boucles** (`for`, `while`).

Le programme que vous allez écrire devra respecter les directives suivantes :

- il devra utiliser le module SVG fourni : on rappelle qu'on peut importer les fonctions d'un module avec la ligne `from mod_svg import debut, fin, rect` ;
- il devra être facilement paramétrable : cela veut dire qu'on devra pouvoir facilement changer certains paramètres du programme sans avoir à modifier son code ; par exemple, vous pouvez définir les constantes suivantes, à déclarer en haut de votre programme :

```
NBR_CASES = 4
TAILLE_CASE = 25
COULEURS = ("white", "black")
```

- s'il nous prenait l'envie de dessiner un échiquer très coloré, on pourrait alors simplement remplacer la dernière constante par `COULEURS = ("blue", "green", "red", "yellow")` sans avoir à modifier le reste du code ;
- toujours pour que le code soit paramétrable, vous devez écrire une fonction `coul_suiv(num_coul)` qui prend en paramètre une couleur et renvoie en résultat la couleur suivante ; en pratique, les couleurs sont définies par un *tuple* et on sait que les tuples sont numérotés séquentiellement à partir de 0 : il suffit donc d'ajouter 1 et de faire un modulo (opérateur `%`) avec la taille du *tuple* (qu'on obtient facilement en utilisant l'opérateur *len*) ;
- on voit bien que notre code va être très répétitif : on va donc le factoriser un peu en écrivant une fonction `dessine_ligne(prem_coul, lig)` qui va dessiner toute une ligne d'un coup, en prenant en paramètre la couleur de la première case de la ligne et le numéro de la ligne courante (en commençant à 0 : en informatique, on commence très souvent à compter à partir de 0 !) ;
- cette fonction devra être elle-même très paramétrée : on utilisera donc autant que possible les constantes définies plus haut, ainsi que les fonctions `coul_suiv` et `rect` pour dessiner les carrés ;
- enfin, on va écrire la fonction `main` qui va dessiner tout l'échiquier, c'est à dire appeler 4 fois la fonction `dessine_ligne` en lui passant les bons paramètres ; n'oubliez pas d'appeler la fonction `debut` du module SVG pour commencer le dessin et la fonction `fin` pour le terminer.

Lorsqu'on va exécuter notre programme, il va afficher le code SVG directement dans le terminal. On peut commencer par lancer l'exécution en tapant `./echiquier.py` pour voir ce qui est produit et s'il n'y a pas d'erreur. Mais ensuite, on veut générer un fichier SVG qu'on pourra visualiser avec un logiciel comme Inkscape par exemple. Pour cela, il suffit de rediriger la sortie du programme dans un fichier, comme vous l'avez surement vu en Unix : `./echiquier.py > image.svg`.

Vérifiez bien aussi que votre programme donne 10/10 lorsqu'on le teste avec `pylint`.

En regardant votre code, vous vous rendrez surement compte qu'il est très répétitif et qu'on a copié-collé des lignes, ce qui n'est jamais une bonne pratique. Une fois que vous saurez utiliser des boucles, vous pourrez facilement modifier le code pour le simplifier.

Eléments de correction

```
from mod_svg import debut, fin, rect

NBR_CASES = 4
TAILLE_CASE = 25

COULEURS = ("white", "black")
# COULEURS = ("blue", "green", "red", "yellow")

def coul_suiv(num_coul):
    """
      Renvoie la couleur suivant celle de numero num_coul
    """
    return (num_coul + 1) % len(COULEURS)

def dessine_ligne(prem_coul, lig):
    """
      Dessine une ligne de l'échiquier
    """
    col = 0
    num_coul = prem_coul
    rect(col * TAILLE_CASE, lig * TAILLE_CASE, TAILLE_CASE,
         TAILLE_CASE, COULEURS[num_coul])
    num_coul = coul_suiv(num_coul)
    col += 1
    rect(col * TAILLE_CASE, lig * TAILLE_CASE, TAILLE_CASE,
         TAILLE_CASE, COULEURS[num_coul])
    num_coul = coul_suiv(num_coul)
    col += 1
    rect(col * TAILLE_CASE, lig * TAILLE_CASE, TAILLE_CASE,
         TAILLE_CASE, COULEURS[num_coul])
    num_coul = coul_suiv(num_coul)
    col += 1
    rect(col * TAILLE_CASE, lig * TAILLE_CASE, TAILLE_CASE,
         TAILLE_CASE, COULEURS[num_coul])

def main():
    """
      Fonction principale
    """
    debut(NBR_CASES * TAILLE_CASE, NBR_CASES * TAILLE_CASE)
    prem_coul = 0
    dessine_ligne(prem_coul, 0)
    prem_coul = coul_suiv(prem_coul)
    dessine_ligne(prem_coul, 1)
    prem_coul = coul_suiv(prem_coul)
    dessine_ligne(prem_coul, 2)
    prem_coul = coul_suiv(prem_coul)
    dessine_ligne(prem_coul, 3)
    fin()

main()
```