TP 1 - Premiers pas -

À la fin de cette séance, vous devrez rendre le TP. En particulier, vous devez récupérer le fichier `TP1rendu.txt` sur la page du cours et le compléter au fur et à mesure.

Avant de commencer le TP :

1. Ouvrez un terminal et dans votre répertoire personel, créez un dossier `Python` :
   ```
   mkdir Python
   ```
2. Placez-vous dans le dossier `Python` et créez un dossier `TP1` :
   ```
   cd Python
   mkdir TP1
   ```
3. Placez-vous dans le dossier `TP1` :
   ```
   cd TP1
   ```

Exercice 1. Il était une fois le *top level*...
Le but de cet exercice est de vous familiariser avec l'interpréteur Python.

1. Lancez le top level Python (tapez la commande suivante dans le terminal) :
   ```
   python3
   ```
   L'interpréteur Python (aussi appelé *shell Python*) apparaît : il est matérialisé par des petits chevrons : `>>>` que l'on appelle *invite commande* (ou *prompt*). Cela signifie que vous allez pouvoir discuter avec votre Python...
2. Tapez les commandes suivantes. Expliquez les résultats que vous obtenez. Essayez d'anticiper les réponses, et lorsque cela ne correspond pas, expliquez pourquoi.

```
1 >>> 20 + 1
2 >>> 20 / 3
3 >>> 20 // 3
4 >>> 20 % 3
5 >>> 5.45 * 10
6 >>> 2 ** 4
7 >>> (3+2) * 5
8 >>> 3+2 * 5
```

Exercice 2. Chaînes de caractères

1. Au fait, vous n'avez pas été très poli avec Python. Essayez de lui dire bonjour. Que se passe-t-il ? En quelle langue vous répond-il ?
2. Ré-essayez en utilisant des apostrophes.
   Remarque : au top level, vous pouvez utilisez l'*historique de commandes* (flèches haut et bas) qui vous permet de ré-afficher des commandes que vous avez déjà tapées.

Cet énoncé de TP s'inspire très largement d'un cours de programmation en Python donné à l'IUT d'Orsay.

3. Que se passe-t-il si on additionne un chat et de l'eau (la chaîne de caractères chat et la chaîne eau) ? Essayez.
4. Que se passe-t-il si on multiplie (la chaîne) bonjour par 3 ? Essayez.
5. Que se passe-t-il si on ajoute 3 à (la chaîne) bonjour ? Essayez.

Exercice 3. Erreurs
Tapez les commandes suivantes et expliquez ce que vous obtenez (ne vous contentez pas de traduire, expliquez ce qui ne va pas).

```
>>> 20 / 0
>>> 20 @ 3
>>> 'bonjour' / 3
>>> 'bonjour' + 5
>>> (3+2)) * 5
>>> (3+2 * 5
```

Exercice 4. Types
Les expressions en Python (comme `20`, `5.45`, `'bonjour'`, ... que nous venons d'utiliser) ont toutes un type.

1. Déterminez le type des expressions utilisées dans les exercices précédents et vérifiez en utilisant la *fonction* `type()` qui prend une expression en paramètre et renvoie son type.
2. Sur quels types de données peut-on utiliser les opérateurs `+`, `*`, `/`, `//`, `%`, `**` ? Quel est le type du résultat ? Vous pouvez faire des tests en tapant d'autres instructions au top-level.
3. Déterminez également le type et le résultat des expressions suivantes :

```
'3' + '4'
3 + '4'
'3' + 4
'3' * '4'
3 * '4'
'3' * 4
'3' * 4.0
```

4. En Python, vous pouvez "forcer" le type d'une expression à être autre chose que son type (on appelle cela le *transtypage* ou *cast* en anglais). Pour cela, on utilise les fonctions `int()`, `float()` et `str()` pour transformer respectivement en entier, flottant ou chaîne de caractères.
   (a) Essayez de transformer les expressions `3.3`, `2.7`, `'2'`, `'2.3'`, `'a'` en `int`. Que se passe-t-il ? Commentez.
   (b) Essayez de transformer les expressions `3`, `3/2`, `'2'`, `'2.3'` en `float`. Que se passe-t-il ? Commentez.
   (c) Essayez de transformer les expressions `3`, `3/2`, `3.2` en `str`. Que se passe-t-il ? Commentez.

Exercice 5. Variables
Qu'affichent les commandes suivantes quand elles sont exécutées les unes à la suite des autres ? Vous devez essayer de deviner avant de tester !

```
>>> foo
>>> foo = 2.1
>>> foo
>>> type(foo)
>>> foo = 2
>>> type(foo)
>>> bar = foo
>>> foo = 3
>>> bar
>>> foo = foo * bar
```

Exercice 6. Premiers pas avec `idle` : nommage des variables

Nous allons maintenant utiliser un éditeur de texte pour lire et écrire des programmes en Python. Celui que vous utiliserez en TP s'appelle `idle` et est généralement installé en même temps que Python (ou python-tk).

Récupérez le fichier `charabia.py` sur la page du cours et lancez la commande (sans oublier le `&`) :

```
idle-python3.1 charabia.py &
```

Attention, un shell Python va se lancer en même temps : nous déconseillons de l'utiliser en tant que top-level, car il est moins pratique que celui que vous obtenez en lançant `python3` dans un terminal.

1. Avez-vous remarqué les couleurs ? À quoi correspondent-elles ?
2. Que calcule ce programme (`a` et `b` sont les données) ?
3. Vous pouvez exécuter votre programme en lançant la commande :

   ```
   python3 charabia.py
   ```

   ou en tapant sur la touche "F5".
4. Maintenant que vous savez ce que fait ce programme, pour le rendre lisible, choisissez des noms de variables plus "explicites". Voici quelques rappels sur les conventions de nommage des variables :
   - le nom contient uniquement :
     - des lettres sans accents,
     - des chiffres,
   - il débute par une lettre minuscule (les noms commençant par des majuscules servent pour autre chose)
   - si l'on veut que le nom de variable contienne plusieurs mots, on colle les mots en mettant une majuscule à tous les mots sauf le premier comme par exemple dans `hauteurCylindre`.
   - il ne doit pas être un mot réservé du langage comme `if`, `True`, `import`,... (ils sont faciles à reconnaître grâce aux couleurs).
5. Utilisez la fonction `input()` pour que l'utilisateur choisisse les valeurs de `a` et `b` et faites en sorte que l'affichage final soit plus clair.
6. Enfin, rajoutez des commentaires dans votre programme pour décrire ce qu'il fait.

À partir de maintenant, tous vos programmes devront être commentés et utiliser des noms de variables explicites afin d'être facilement compréhensibles.

Exercice 7. Tortue : affectation de variables

Récupérez le fichier `tortue.py` sur la page du cours. Ce programme contient 2 parties :

- la première qui contient des variables et que vous pourrez modifier ;
- et la seconde à laquelle vous ne devez pas toucher. Le code de la deuxième partie utilise la *tortue* qui permet de :
  - ouvrir une fenêtre de taille `600 x 600` pixels,
  - poser le stylo au point de coordonnées `(xInit,yInit)`,
  - tracer `nbLignes` de longueur `lgLigne` pixels en tournant de `angle` degrés vers la gauche à chaque fois.

1. Lancez le programme en tapant l'instruction suivante dans le terminal :
   ```
   python3 tortue.py
   ```
2. Que voyez-vous ? Avez-vous remarqué la petite flèche symbolisant la tortue ?
3. Modifiez les variables afin de centrer le carré.
4. Modifiez les variables pour que la longueur du carré soit `300` pixels.
5. Votre carré est-il toujours centré ?
6. Modifier les variables pour que le carré soit centré quelle que soit la longueur de ses côtés (en restant inférieure à 600). Vous pouvez utiliser la fonction `input()` pour que l'utilisateur choisisse la longueur.
7. Modifiez les variables pour que le programme dessine un triangle équilatéral.
8. Modifiez les variables pour que le programme dessine un pentagone régulier.
9. Modifiez les variables pour que le programme dessine un polygone régulier quel que soit le nombre de lignes choisi (utilisez la fonction `input()`).
10. Que se passe-t-il si on demande un polygone à 50 côtés ? Corrigez les variables si nécessaire.
11. ★ Votre dessin est-il toujours centré ?

    Pour centrer votre polygone, vous aurez besoin des fonctions trigonométriques comme cosinus, sinus et tangente. Pour y avoir accès en Python, il faut rajouter au tout début de votre programme la ligne :
    ```
    from math import *
    ```
    Vous pourrez ensuite avoir accès à la constante `pi` et aux fonctions `cos`, `sin` et `tan` qui attendent un angle donné en radian (et non pas en degrés).

```
>>> from math import *
>>> pi
3.141592653589793
>>> cos(pi)
-1.0
```