TP 1 (programmation) - Premiers pas.

Ce TP (ainsi que les suivants) est à rendre en fin de cette séance. Vous devrez rendre un rapport en PDF et des fichiers .py pour les derniers questions. (Le barème est indicatif donc sujet à modifications).

Sauf indication, **les questions ★ ne sont pas facultatives**. Elles sont juste plus difficiles.

Avant de commencer le TP :

1. Ouvrez un terminal et dans votre répertoire personnel, créez un dossier `Python-prog` :

   ```
   $ mkdir Python-prog
   ```

2. Placez-vous dans le dossier `Python-prog` et créez un dossier `TP1` :

   ```
   $ cd Python-prog
   $ mkdir TP1
   ```

3. Placez-vous dans le dossier `TP1` :

   ```
   $ cd TP1
   ```

Exercice 1. Il était une fois le *top-level*...(1 point)
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 lorsqu'elles ne correspondent pas à vos attentes, expliquez pourquoi.

```
>>> 20 + 1
>>> 20 / 3
>>> 20 // 3
>>> 20 % 3
```

```
>>> 5.45 * 10
>>> 2 ** 4
>>> (3+2) * 5
>>> 3+2 * 5
```

Exercice 2. Chaînes de caractères (2 points)

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.
3. Que se passe-t-il si on additionne un chat et de l'eau (les chaînes de caractères) ? 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 (2 points)
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
>>> (3+2))* 5
```

```
>>> ( 3 + 2 * ) 5
>>> 2 = b
>>> 2 == b
```

Exercice 4. Types (3 points)
Les expressions en Python (comme `20`, `5.45`, `'bonjour'`, ... que nous venons d'utiliser) ont toutes un type.

1. Déterminez le type des expressions suivantes et vérifiez (quand c'est possible) en utilisant la *fonction* `type()` qui prend une expression en paramètre et renvoie son type.

   Remarque : à partir de la ligne 4, on suppose que l'affectation `a = 2` a été effectuée.

```
>>> 3 + 4.0
>>> 'bon' + 'jour'
>>> a = 2
>>> a == 2
```

```
>>> a / 2
>>> a // 3
>>> print(2)
>>> print
```

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. Pour chacune des expressions suivantes, indiquez si elles sont valides ou non. Si oui, donnez leur type et leur valeur :

```
>>> 3 + 4
>>> 3 + '4'
>>> '3' + '4'
>>> 3 * 4
```

```
>>> '3' * 4
>>> '3' * 4.0
>>> '3' * '4'
>>>
```

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 les données respectivement en entier, flottant ou chaîne de caractères.

   Essayez de transformer les expressions ...

   (a) ...`3.3`, `2.7`, `'2'`, `'2.3'`, `'a'` en `int`. Que se passe-t-il ? Commentez.

   (b) ...`3`, `3/2`, `3//2`, `'2'`, `'2.3'` en `float`. Que se passe-t-il ? Commentez.

   (c) ...`3`, `3/2`, `3.2` en `str`. Que se passe-t-il ? Commentez.

Exercice 5. Variables (3 points)
Devinez ce qu'affichent les commandes suivantes quand elles sont exécutées les unes à la suite des autres, **puis** testez pour vérifier.

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

```
>>> bar = foo
>>> bar == foo
>>> foo = 3
>>> bar == foo
>>> bar
>>> foo = foo * bar
```

Exercice 6. Si seulement...(4 points)

1. Donnez une valeur de votre choix à la variable `x`.
2. Écrivez le code qui affiche "froid" si la valeur de `x` est négative et "chaud" sinon.
3. Inversez la valeur de `x` et reprenez la question précédente.
4. On commence à atteindre les limites de l'utilisation du top-level : l'indentation n'est pas commode et l'historique des commandes sur plusieurs lignes est fastidieuse. Quittez le top-level, recopiez le code que vous venez d'écrire dans un fichier nommé `test.py` et tapez la commande suivante dans le terminal pour exécuter votre programme.

   ```
   $ python3 test.py
   ```

5. Si rien ne s'affiche, expliquez pourquoi et modifiez votre programme pour qu'il affiche "chaud" ou "froid".

Exercice 7. Tortue : affectation de variables (5 points)

Récupérez le fichier `tortue.py` sur la page du cours et ouvrez-le dans un éditeur de texte (comme `Geany`, par exemple). 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 centrée en $(0, 0)$, dont la taille en pixels s'affiche dans le terminal,
  — poser le stylo au point de coordonnées `(xInit,yInit)`,
  — tracer `nbLines` de longueur `length` pixels en tournant de `angle` degrés vers la gauche à chaque fois.

Ce programme utilise le **module standard** `turtle`, dont on peut consulter la documentation ici : `https://docs.python.org/3/library/turtle.html`.

1. Lancez le programme `tortue.py` avec `python3`.
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 ? Modifiez les variables pour que le carré soit centré quelle que soit la longueur de ses côtés (en restant inférieure à 600). Utilisez la fonction `input()` pour que l'utilisateur choisisse la longueur (attention au type !).
6. Modifiez les variables pour que le programme dessine un triangle équilatéral.
7. Modifiez les variables pour que le programme dessine un pentagone régulier.
8. Modifiez les variables pour que le programme dessine un polygone régulier quel que soit le nombre de lignes choisi (utilisez la fonction `input()`).
9. Que se passe-t-il si on demande un polygone à 50 côtés ? Corrigez les variables si nécessaire.
10. ★ Votre dessin est-il toujours centré ? Faites en sorte qu'il le soit.

    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).

```
1 >>> from math import *
2 >>> pi
```

```
3.141592653589793
>>> cos(pi)
-1.0
```

11. ★ **(facultatif)** Modifiez le programme pour faire en sorte que la longueur du côté soit ajustée automatiquement aux dimensions de la fenêtre afin que le polygone s'affiche en entier, dans le cas où les valeurs choisies par l'utilisateur ne le permettent pas (par exemple, avec 50 côtés de longueur 100).
12. ★ **(facultatif, pour ceux qui ont déjà programmé en Python)** Utilisez la tortue pour tracer un damier de taille 400×400 ayant n cases noires par ligne et par colonne. Remarque : la fonction `home()`, qui replace la tortue dans son état initial, peut être utile.