TP 1 (programmation) - Premiers pas.

Avant de commencer le TP : nous prendrons l'habitude de consacrer un répertoire à chaque TP, ce qui simplifiera l'organisation de vos fichiers. Pour ce faire :

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 `TP01` :

```
$ cd Python-prog
$ mkdir TP01
```

3. Placez-vous dans le dossier `TP01` :

```
$ cd TP01
```

Remarque : sauf indication contraire, **les questions ★ ne sont pas facultatives**. Elles sont juste plus difficiles.

1 Applications

Exercice 1 (Il était une fois l'interpréteur...)

Le but de cet exercice est de vous familiariser avec l'interpréteur Python.

(a) Lancez l'interpréteur 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 dialoguer avec l'interpréteur.

(b) 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)

(a) 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 ?

(b) Ré-essayez en utilisant des apostrophes.
Remarque : dans l'interpréteur, vous pouvez utilisez l'*historique de commandes* (flèches haut et bas) qui vous permet de ré-afficher des commandes que vous avez déjà tapées.

(c) Que se passe-t-il si on additionne `'chat'` et `'eau'` ? Essayez.

(d) Que se passe-t-il si on multiplie la chaîne `'bonjour'` par 3 ? Essayez.

(e) 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
>>> (3+2))* 5
```

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

Exercice 4 (Types)

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

(a) 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
```

(b) 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 dans l'interpréteur.

(c) 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'

```

(d) Python permet de "forcer" le type d'une expression à être autre chose que son type ; on appelle cela le *transtypage*. Pour cela, on utilise les fonctions `int()`, `float()` et `str()` pour transformer les données respectivement en entier, flottant ou chaîne de caractères. Essayez de transtyper les expressions suivantes :

1. `3.3`, `2.7`, `'2'`, `'2.3'`, `'a'` en `int`. Que se passe-t-il ? Commentez.
2. `3`, `3/2`, `3//2`, `'2'`, `'2.3'` en `float`. Que se passe-t-il ? Commentez.
3. `3`, `3/2`, `3.2` en `str`. Que se passe-t-il ? Commentez.

Exercice 5 (Variables)

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...)

(a) Donnez une valeur de votre choix à la variable `x`.

(b) Écrivez le code qui affiche "froid" si la valeur de `x` est négative et "chaud" sinon.

(c) Inversez la valeur de `x` et reprenez la question précédente.

(d) On commence à atteindre les limites de l'utilisation de l'interpréteur : l'indentation n'est pas commode et l'historique des commandes sur plusieurs lignes est fastidieuse. Quittez l'interpréteur, recopiez le code que vous venez d'écrire dans un fichier nommé `test.py` et exécutez votre programme à l'aide de la commande suivante :

```
$ python3 test.py
```

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

Exercice 7 (Chasse aux erreurs)

Le programme ci-dessous est censé demander l'âge de l'utilisateur et afficher :
— `Vous êtes mineur.` si l'âge est strictement inférieur à 18 ;
— `Vous êtes majeur.` dans le cas contraire.
Cependant, il contient plusieurs erreurs.

```
1 if __name__ = "__main__":
2 age = input("Quel âge avez-vous ? ")
3
4     if age < 18:
5     print("Vous êtes mineur.")
6     if age >= 18
7         print("Vous êtes majeur.")
```

(a) Sans exécuter le programme, repérez le plus grand nombre possible d'erreurs. Pour chaque erreur, indiquez à quelle catégorie elle appartient :
— erreur de syntaxe ;
— erreur d'indentation ;
— erreur liée au type d'une valeur ;
— confusion entre affectation et comparaison.

(b) Recopiez le programme dans un fichier nommé `age.py`, puis essayez de l'exécuter :

```
$ python3 age.py
```

(c) Lisez les messages d'erreur produits par Python et corrigez les erreurs une par une. Après chaque correction, relancez le programme.

(d) Donnez une version entièrement corrigée du programme.

(e) Testez votre programme avec les valeurs suivantes : 12, 17, 18, 25. Indiquez le résultat attendu pour chacun de ces tests.

(f) Modifiez le programme pour qu'il affiche trois résultats différents :
— `Vous êtes mineur.` si l'âge est inférieur à 18 ;
— `Vous avez exactement 18 ans.` si l'âge vaut 18 ;
— `Vous êtes majeur.` si l'âge est supérieur à 18.
N'utilisez que des instructions `if`.

(g) ★ Modifiez le programme afin qu'il affiche un message particulier si l'âge saisi est négatif ou supérieur à 130. Par exemple : `Âge incorrect.` Dans ce cas, le programme ne doit afficher aucun des messages `Vous êtes mineur.` ou `Vous êtes majeur.`

2 Mini-projet : polygones avec la tortue

Récupérez le fichier `tortue_init.py` sur la page du cours et ouvrez-le dans votre éditeur de texte. Ce programme utilise le **module**[1] **standard** `turtle`, dont on peut consulter la documentation ici : `https://docs.python.org/3/library/turtle.html`, et contient deux parties :

— une première partie contenant des variables, que vous pouvez modifier ;
— une seconde partie sous le commentaire `# Ne modifiez rien sous cette ligne`, à laquelle vous ne devez pas toucher.

Le programme ouvre une fenêtre graphique, place une "tortue" (qui joue le rôle d'un crayon avec une certaine orientation) au point de coordonnées `(xInit, yInit)`, puis lui fait tracer `nbLines` segments de longueur `length`. Après chaque segment, la tortue tourne vers la gauche d'un angle de `angle` degrés.

Attention

Vous serez amenés à modifier ce fichier dans le cadre de ce mini-projet. Pensez à en effectuer une sauvegarde pour pouvoir la restaurer en cas de besoin.

Niveau 1 : prise en main

(a) Exécutez le programme récupéré. Qu'affiche-t-il ?

(b) Observez le dessin obtenu et identifiez le rôle de chacune des variables suivantes :

`xInit`, `yInit`, `nbLines`, `length` et `angle`.

N'hésitez pas à modifier leurs valeurs pour voir comment cela affecte le programme.

(c) Modifiez les variables afin que le programme trace un carré de côté 300 pixels.

(d) Faites en sorte que ce carré soit centré dans la fenêtre.

(e) Remplacez le carré par un triangle équilatéral (pas besoin de le centrer).

(f) Remplacez ensuite le triangle par un pentagone régulier (pas besoin de le centrer).

Niveau 2 : programme paramétrable

Le programme en son état actuel n'est pas très souple. Nous allons maintenant le paramétrer pour pouvoir plus facilement modifier son comportement.

(g) Modifiez le programme afin que l'utilisateur choisisse le nombre de côtés du polygone :

```
Nombre de côtés? 7
```

Attention au type de la valeur renvoyée par la fonction `input`.

(h) Calculez automatiquement l'angle de rotation de la tortue en fonction du nombre de côtés choisi. Votre programme doit maintenant être capable de dessiner un polygone régulier ayant un nombre quelconque de côtés supérieur ou égal à 3.

(i) Demandez également à l'utilisateur de choisir la longueur des côtés :

```
Nombre de côtés? 7
Longueur des côtés? 80
```

(j) Testez notamment votre programme avec :

1. Nous parlerons des *modules* dans un cours ultérieur.

— 3 côtés de longueur 200 ;
— 4 côtés de longueur 150 ;
— 8 côtés de longueur 100 ;
— 50 côtés de longueur 20.
Décrivez brièvement ce que vous observez dans le dernier cas.

Défis

(k) ★ Centrez précisément le polygone, quel que soit son nombre de côtés. Pour cela, vous pouvez utiliser les fonctions trigonométriques du module `math`, notamment la constante π et la fonction tangente (qui comme les autres fonctions trigonométriques utilise des angles exprimés en radians). Ajoutez au début du programme :

```
1 from math import cos, pi, tan
```

Remarque : en cas de besoin, cette page contient des informations très utiles.

(l) ★ Faites en sorte que la longueur des côtés soit automatiquement réduite dans le cas où le polygone serait trop grand pour tenir entièrement dans la fenêtre.

(m) ★ Expérimentez avec une valeur de `angle` différente de celle d'un polygone régulier et essayez d'obtenir une étoile ou une rosace. Notez les valeurs de `nbLines` et de `angle` correspondant à votre dessin.

(n) ★ (facultatif) Réalisez un dessin personnel en combinant plusieurs figures : une maison, une enveloppe, une étoile, une rosace ou toute autre figure de votre choix. Consultez la documentation officielle pour plus d'informations sur les fonctionnalités disponibles.