TD: Python, les bases:

Exercice 1: structure conditionnelle: On s'interesse au programme suivant:

```
"""Premier programme"""
a = int(input("x="))
if a < 0:
    print("x negatif")
if a >= 0 and a <= 10:
    print("x est compris entre 0 et 10")
if a >= 0 and a <= 100:
    print("x est compris entre 0 et 100")
else:
    print("x est plus garnd que 100")
```

bases1.py

1. Qu'est ce que ce programme permet de tester?
2. Quelle est la différence entre x et a?
3. x est-il une variable du programme?
4. Quel est le type de a?
5. Tester ce programme avec la valeur 3. Quel problème voit-on apparaître?
6. Avec quel structure condiionnelle peut-on résoudre le problème de la question précédente?

Exercice 2: True-False: On s'interesse au début du programme suivant:

```
age = int(input("Saisissez votre age : "))
majeur = False
```

bases2.py

1. Quels sont les types de age et de majeur?
2. Compléter ce programme **en testant la variable majeur** afin d'afficher si la personne est majeure ou mineure.

Exercice 3: String: On s'interesse au programme suivant:

```
"""Premier programme"""
nom = input("Saisissez votre nom : ")
age = input("Saisissez votre age : ")
phrase = ""
print(phrase)
```

bases3.py

1. Quels sont les variables du programme et leur type?
2. Compléter la variable phrase afin que le programme renvoie: Je suis *nom*, j'ai *age* ans.
3. Il est aussi possible d'utiliser la structure suivante:

```
print("{0} a eu {1} ans. On a fete les {1} ans de {0}." .format("Pierre", 25))
```

bases3.py

La tester dans le programme précédent.

Premier bilan:

- Le nom des variables doit être explicite et écrit en minuscule, y compris la première lettre, en utilisant des lettres ordinaires (pas de cédille, d'accent, de caractères spéciaux...)
- Éviter d'écrire des instruction trop longues sur une même ligne afin de faciliter la lecture du programme.
- Penser à commenter un programme afin d'aider une tierce personne à comprendre les choix qui ont été faits ou à s'aider soi-même lorsqu'on reprend ultérieurement son programme afin de le poursuivre. Les commentaires doivent être précis et concis.

Exercice 4: On s'interesse aux programmes suivant:

```
"""prgm 1"""
x,y,z = 4,0,0
if x == 4:
    y = 1
else:
    y = 2
    z = 3
print(x,y,z)

"""prgm 2"""
x,y,z = 4,0,0
if x == 4:
    y = 1
else:
    y = 2
z = 3
print(x,y,z)
```

bases4.py

Quel est le résultat de chacun? Expliquer.

Exercice 5: Écrire un programme qui demande à l'utilisateur de saisir un nombre entier et qui renvoie en sortie un affichage indiquant s'il est divisible ou pas par 7.

Exercice 6: En utilisant la fonction random(), écrire un programme qui affiche aléatoirement pile ou face de façon équiprobable.

Exercice 7: boucle While: On s'interesse au programme suivant:

```
0 table = 7
  i = 0
2 while i < 10:
      print(str(i + 1) + "x" + str(table) + "=" + str((i + 1) * table))
4     i += 1
```

bases7.py

1. Que permet de faire ce programme?
2. Quel est le rôle de l'instruction de la ligne 4? Peut-on l'écrire autrement?
3. Ce programme ce termine t-il? Le modifier afin qu'il ne se termine jamais.
4. Écrire un programme permettant d'afficher la table des 9.

Exercice 8: boucle For: On s'interesse au programme suivant:

```
0 chaine = "Salut Ã  toi!"
  for lettre in chaine:
2     print(lettre)
```

bases8.py

1. Que permet de faire ce programme?
2. Comment le modifier pour que les valeurs soient affichées sur la même ligne?
3. Écrire un programme utilisant la boucle for qui prend en entrée une chaîne de caractère et qui renvoie la chaîne de caractère avec les consonnes remplacées par *.

Exercice 9: Écrire un programme qui affiche les entiers de 10 à 1:

1. Avec une boucle for.
2. Avec une boucle while.

Second bilan:

- Sans initialisation, la boucle while n'est pas comprise puisque la variable sur laquelle elle porte n'existe pas. De plus, la valeur choisie au départ ainsi que la borne choisie dans la boucle while et l'opérateur conditionnent le nombre de tours effectués par la boucle. Ces choix doivent donc être réfléchis.
- Sans incrémentation, la valeur de la variable reste toujours à sa valeur initiale. La condition pour rentrer dans la boucle est donc toujours remplie. En conséquence, la boucle tournera indéfiniment. On a alors créé une boucle infinie. Python n'étant pas capable de reconnaître un boucle infinie, c'est au programmateur de l'interrompre en tapant Ctrl + C dans la fenêtre de l'interpréteur.

Exercice 10: Combien de points sont affichés à l'exécution des ces deux programmes ? Expliquer.

```
"""prgm 1"""
for i in range(0,100):
  print(".", end="")
for j in range(0,100):
  print(".", end="")


"""prgm 2"""
for i in range(0,100):
  for j in range(0,100):
    print(".", end="")
```

bases10.py

Exercice 11: Écrire un programme affichant le première valeur entière k à partir de laquelle $3k > 10000$.

Exercice 12: Écrire un programme permettant d'afficher toutes les tables de multiplication de 0 à 10.

Exercice 13: Écrire un programme qui affiche la suite de symboles suivante:

```
*
**
***
****
*****
******
*******
```

Figure 1: Suite de symboles

Exercice 14: fonctions: Expliquer ce que fait le programme suivant:

```
def TirerUnTrait():
    print("----------")
    print()
    print()

def AnnoncerUnVol(destination, horaire):
    print("Le vol en direction de ",end="")
    print(destination,end="")
    print(" dÃ©collera Ã  ",end="")
    print(horaire)
    TirerUnTrait()

AnnoncerUnVol("Tokyo", "9h00")
AnnoncerUnVol("Sydney", "9h30")
AnnoncerUnVol("New-York", "10h00")
```

bases14.py

Exercice 14: modules: Expliquer les différences dans les imports du programme suivant:

```
"""prgm 1"""
import math
a = math.sqrt(36)
print(a)

"""prgm 2"""
from math import sqrt
a = sqrt(36)
print(a)

"""prgm 3"""
from math import *
a = sqrt(36)
print(a)
```

bases15.py

Exercice 16:

1. Écrire une fonction max2 qui prend en arguments deux valeurs et renvoie la plus grande.
2. Écrire une fonction repete qui répète le mot NSI un certain nombre de fois au choix.
3. Écrire une fonction tirage qui tire au sort un nombre entier entre deux bornes données en arguments. On pourra utiliser les fonctions du paquet random.
4. Écrire une fonction constructible qui décide s'il est possible de construire un triangle avec trois segments de mesures données.
5. Écrire une fonction max3 qui prend en argument trois valeurs et renvoie la plus grande, en utilisant la fonction max2 de la première question.

Exercice 17:

1. Écrire une fonction qui demande en entrée la longueur de chaque côté d'un triangle (en cm) et renvoie son aire en sortie en utilisant la formule de Heron (`https://fr.wikipedia.org/wiki/Formule_de_H%C3%A9ron`).

   (Source Wikipédia) La formule de Héron présente une instabilité lors du calcul numérique, qui se manifeste pour les triangles en épingle, c'est-à-dire dont un côté est de dimension très petite par rapport aux autres. En choisissant les noms de côtés de sorte à ce que $a > b > c$, et en réorganisant les termes de façon à optimiser les grandeurs ajoutées ou soustraites, on obtient la formule de Kahan, plus stable.

2. Écrire une fonction qui demande en entrée la longueur de chaque côté d'un triangle (en cm) et renvoie son aire en sortie en utilisant la formule de Kahan.

3. Tester les deux programmes avec des valeurs extrêmes (exemple: $a = b = 1000000000$ et $c = 0,000000001$).

Exercice 18: tableaux (ou listes): Expliquer ce que fait le programme suivant:

```
tab1 = [1, 9, 33]
tab1.append(56)
tab1.insert(1,5)

tab2 = [76,25,98]
del tab2[1]

tab = tab1 + tab2

print(tab1)
print(tab2)
print(tab)
print(tab[3])
print(tab[1:4])
print(tab[1:])
print(tab[:4])
```

bases18.py

Exercice 19: copies de tableaux: Expliquer ce que fait le programme suivant:

```
liste_1 = [1, 2, 3]
liste_2 = liste_1

liste_2.append(4)
liste_2.append(5)

print("liste_1 =", liste_1)
print("liste_2 =", liste_2)
```

bases19.py

Exercice 20: tuples: Expliquer ce que fait le programme suivant:

```
tuple_vide = ()
tuple_non_vide = (1, 3, 5)

thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(tuple_vide, tuple_non_vide, thistuple[2:5])
```

bases20.py

Troisième bilan:

- A la différence des listes, les tuples, une fois créés, ne peuvent être modifiés. On ne peut ni y ajouter ni y retirer d'objets.
- Il est assez rare que l'on travaille directement sur les tuples. Néanmoins, il est bon de savoir que cela existe car nous avons eu l'occasion de les utiliser à plusieurs reprises sans nous en rendre réellement compte.

Exercice 21:

1. Créer dans la console un tableau L contenant 12 valeurs qui sont les douze premiers nombres pairs, le premier élément du tableau étant zéro.
2. Faire afficher le dixième terme du tableau.
3. Remplacer le neuvième terme du tableau par 9.
4. Saisir len(L). Quelle est la réponse obtenue ? Quelle est le rôle de la fonction len() ?

Exercice 22: Écrire un programme qui demande à l'utilisateur de saisir les nombres de son choix dans un tableau et qui compte de nombre d'éléments dont la valeur est supérieure à 10.

Exercice 23: Écrire un programme qui demande à l'utilisateur de saisir les nombres de son choix dans un tableau et qui détermine l'élément maximal de ce tableau.

Exercice 24:

1. Créer un tableau contenant dans l'ordre les lettres de l'alphabet de a à j.
2. Écrire un script utilisant une boucle while et permettant d'afficher tous les éléments du tableau.
3. Écrire un script utilisant une boucle for et permettant d'afficher tous les éléments du tableau.

Exercice 25: On dispose d'un tableau de nombres entiers quelconques donnés par l'utilisateur. Certains d'entre eux sont présents en plusieurs exemplaires. Écrire un script qui recopie ce tableau en

omettant les doublons (les nombres qui se répètent n'apparaîtront qu'une seule fois) et où les éléments sont classés par ordre croissant.

Exercice 26: Écrire un programme qui analyse un par un tous les éléments d'un tableau de nombre pour générer deux nouveaux tableaux. L'un contiendra les nombres pairs du tableau initial et l'autre les nombres impairs.

Exercice 27: arborescence:

1. Créer dossiers et fichiers en respectant l'arborescence suivante (les dossiers sont représentés par un rectangle et les fichiers par un ovale):

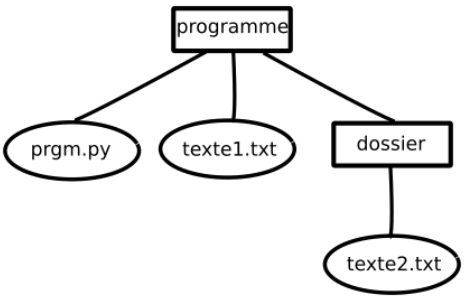

Figure 2: Arborescence

2. Ecrire dans le fichier texte1.txt "contenu du texte 1" et dans le fichier texte2.txt "contenu du texte 2".

3. Compéter le programme prgm.py avec le code suivant:

```
mon_fichier = open("texte1.txt", "r")
contenu = mon_fichier.read()
print(contenu)
mon_fichier.close()
```

prgm.py

4. Expliquer avec précision ce que fait ce programme.

5. Modifier ce programme afin qu'il affiche le contenu de texte2.txt