TP 02

Environnement et système d'exploitation

Savoirs et compétences :

- ❒ *AA.C1 : Manipuler un OS ou un IDE*
- ❒ *AA.S1 : Se familiariser aux principaux composants d'une machine numérique*
- ❒ *AA.S3 : Se familiariser à la manipulation d'un IDE*

Consignes

1. **Lisez attentivement tout l'énoncé avant de commencer.**
2. Après la séance, vous devez rédiger un compte-rendu de TP et l'envoyer au format électronique à votre enseignant.
3. Le seul format accepté pour l'envoi d'un texte de compte-rendu est le format PDF. Votre fichier s'appellera impérativement `tp02_kleim_durif.pdf`, où « `kleim` » et « `durif` » sont à remplacer par les noms des membres du binôme.
4. Ce TP est à faire en binôme, vous ne rendrez donc qu'un compte-rendu pour deux.
5. Vous préciserez en préambule de votre compte-rendu les noms des membres du binôme ainsi que le système d'exploitation sur lequel vous avez travaillé.
6. Ayez toujours un crayon et un papier sous la main. Quand vous réfléchissez à une question, utilisez-les!
7. Vous devez être autonome. Ainsi, avant de poser une question à l'enseignant, merci de commencer par :
   - relire l'énoncé du TP (beaucoup de réponses se trouvent dedans) ;
   - relire les passages du cours [1] relatifs à votre problème ;
   - effectuer une recherche dans l'aide disponible sur votre ordinateur (ou sur internet) concernant votre question.

   Il est alors raisonnable d'appeler votre enseignant pour lui demander des explications ou une confirmation!

Activité 1 : Environnement de développement intégré Python et prise en main élémentaire de Python

Lancer PYZO ou IDLE. Un *interpréteur de commandes*, ou `shell`, s'affiche. Le symbole `>>>` signifie que `Python` attend vos instructions.

Sitôt une instruction tapée et validée (par la touche « Entrée »), le `shell` effectue le calcul demandé puis affiche un résultat, ou un message d'erreur. Il est extrêmement important de bien lire ces messages d'erreur, et de les comprendre!

Q 1 : Taper dans le *shell* les instructions suivantes.

```
x = 42
y = 42.
type(x)
type(y)
x = x+y
x
type(x)
```

Que se passe-t-il? Qu'est-ce que cela signifie?

1. Dans le cas fort improbable où vous ne vous en souviendriez pas.

Q 2 : Décrire ce que font les opérations suivantes, après les avoir étudiées sur des exemples numériques.

```
+    -    *    **    /    //    %
```

Q 3 : Taper dans le *shell* les instructions suivantes.

```
B = 42 > 41.
type(B)
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Q 4 : Décrire ce que font les opérations suivantes, après les avoir étudiées sur des exemples numériques.

```
==     !=    <    >    <=    >=
```

Q 5 : Taper dans le *shell* les instructions suivantes.

```
3/0 > 5
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Q 6 : Taper dans le *shell* les instructions suivantes.

```
B = (42 > 41) or ( 3/0 > 5).
type(B)
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Q 7 : Décrire ce que font les opérations suivantes, après les avoir étudiées sur des exemples logiques.

```
or    and    not
```

Q 8 : Taper dans le *shell* les instructions suivantes.

```
x = -3
abs(x)
help(abs)
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Q 9 : Taper dans le *shell* les instructions suivantes.

```
import math as m
import numpy as np
m.sin(m.pi)
np.sin(np.pi)
np.sin([0,np.pi])
m.sin([0,m.pi])
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Q 10 : Taper dans le *shell* les instructions suivantes.

```
L = [0,1,2,3,4,5,6]
type(L)
L[0]
L[6]
L[-1]
L[-2]
L[7]
L[1:4]
L[2:8]
L.append(7)
L
L = L.append(8)
L
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Nous avons vu comment utiliser des fonctions et des bibliothèques. Nous pouvons bien entendu créer nos propres fonctions (et bibliothèques).

Dans PYZO ou IDLE, ouvrir un nouveau fichier (CTRL+N ou File / New file). L'enregistrer (CTRL + S ou File / Save) sous le nom `TP02.py`.

Q 11 : Taper dans cette fenêtre le script suivant.

```
"""TP n02"""
def somme(n) :
    """Renvoie 0 + 1 + 2 + ... + n
    Precondition : n entier naturel"""
    return n*(n+1) // 2
```

Enregistrer puis appuyer sur la touche Crtl+MAJ+E (sous PYZO) ou F5 (sous IDLE). Le *shell* doit s'afficher. Taper dans le *shell* les instructions suivantes.

```
somme(42)
somme(42.)
somme(-1515)
help(somme)
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Q 12 : Comment peut-on utiliser la fonction écrite précédemment dans un *autre* script `Python` ?

Activité 2 : Types composés

Q 13 : Prévoir les résultats des expressions suivantes, puis le vérifier grâce à l'interpréteur interactif d'IDLE.

a) `[1,2,3,"a"]`
b) `123a`
c) `[]`
d) `[]+[]`
e) `[]+[] == []`
f) `[1,2] + [5,7,9]`
g) `[0,0]+[0]`
h) `len(["a","b"])`
i) `len([])`
j) `len([[]])`
k) `len([[[]]])`
l) `len([0,0]+[1])`

Q 14 : Calculer cette suite d'expressions.

```
[i for i in range(10)]
[compt**2 for compt in range(7)]
[j+1 for j in range(-2,8)]
```

Sur ce modèle, obtenir de manière synthétique :
a) la liste des 20 premiers entiers naturels impairs ;
b) la liste de tous les multiples de 5 entre 100 et 200 (inclus) ;
c) La liste de tous les cubes d'entiers naturels, inférieurs ou égaux à 1000.
d) une liste contenant tous les termes entre -20 et 5 d'une progression arithmétique de raison 0,3 partant de -20.

Activité 3 : Variables

Q 15 : Voici des affectations successives des variables a et b. Dresser un tableau donnant les valeurs de a et b à chaque étape.

```
>>> a = 1
>>> b = 5
>>> a = b-3
>>> b = 2*a
>>> a = a
>>> a = b
```

Q 16 : Écrire une séquence d'instructions qui échange les valeurs de deux variables x et y.

Q 17 : Écrire, sans variable supplémentaire, une suite d'affectation qui permute circulairement vers la gauche les valeurs des variables x, y, z : x prend la valeur de y qui prend celle de z qui prend celle de x.

Q 18 : En fonction de n, et avec les contraintes précédentes, quel est le nombre minimum d'instructions pour calculer x^n ?

Q 19 : Calculer, sans utiliser la fonction `sqrt` ni la division flottante `/`, les nombres suivants.

a) $\dfrac{1}{7,9}$
b) $\sqrt{6,2}$
c) $\dfrac{1}{\sqrt{3,5}}$
d) $2\sqrt{2}$

De base, on ne peut réaliser que des calculs élémentaires avec Python. Cependant, il est possible d'avoir accès à des possibilités de calcul plus avancées en utilisant une *bibliothèque*. Par exemple, la bibliothèque `math` permet d'avoir accès à de nombreux outils mathématiques. On peut donc taper

```
from math import sqrt, log, exp, sin, cos, tan, pi, e
```

pour avoir accès à toutes ces fonctions.
Calculer les nombres suivants (on n'hésitera pas à consulter l'aide en ligne).

a) e^2
b) $\sqrt{13}$
c) $\cos\left(\dfrac{\pi}{5}\right)$
d) $e^{\sqrt{5}}$
e) $\ln 2$
f) $\ln 10$
g) $\log_2 10$
h) $\tan\left(\dfrac{\pi}{2}\right)$

Activité 4 : Fonctions

Q 20 :
1. Ecrire une fonction qui à un nombre entier associe le chiffre des unités.
2. Ecrire une fonction qui à un nombre entier associe le chiffre des dizaines.
3. Ecrire une fonction qui à un nombre entier associe le chiffre des unités en base 8.

Q 21 : Ouvrir votre IDE, écrire la fonction suivante dans un fichier, l'enregistrer, taper *run* (F5) puis utiliser la fonction dans l'interpréteur interactif. Décrire ensuite précisément ce que réalise cette fonction.

```
def split_modulo(n):
  """A vous de dire ce que fait cette fonction !"""
  return (n%2,n%3,n%5)
```

Q 22 : Écrire une fonction `norme` qui prend en argument un vecteur de $\mathbb{R}^2$ donnée par ses coordonnées et renvoie sa norme euclidienne. Vous devrez spécifier clairement le type de l'argument à l'utilisateur via la *docstring*.

Q 23 : Écrire une fonction `lettre` qui prend en argument un entier `i` et renvoie la `i`$^{\text{e}}$ lettre de l'alphabet.

Q 24 : Écrire une fonction `carres` qui prend en argument un entier naturel `n` et qui renvoie la liste des `n` premiers carrés d'entiers, en commençant par 0.