TP 02

Fonctions

Combinaison des exos

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*

1 Applications

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!

1.1 Prise en main élémentaire de `Python`.

Lancer IDLE (via un terminal, n'oubliez pas l'autocomplétion, ou le menu). 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?

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

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

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

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 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 F5 ou (Run / Run Module). 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`? **Q 13 : Prévoir les résultats des expressions suivantes, puis le vérifier grâce à l'interpréteur interactif.**

a) `(1,2)`
b) `(1)`
c) `(1,)`
d) `(,)`
e) `()`
f) `()+()`
g) `()+() == ()`
h) `(1,2)+3`
i) `(1,2)+(3)`
j) `(1,2)+(3,)`
k) `(1,2)+(3,4,5)`
l) `len((1,7,2,"zzz",[]))`
m) `len(())`
n) `len(("a","bc")+("cde",""))`

Q 14 : Pour chaque séquence d'instruction, prévoir son résultat puis le vérifier grâce à l'interpréteur interactif.

a)
```
t = (2,"abra",9,6*9,22)
print(t)
t[0]
t[-1]
t[1]
t[1] = "cadabra"
```

b)
```
res = (45,5)
x,y = res
(x,y) == x,y
(x,y) == (x,y)
print x
print(y)
x,y = y,x
print(y)
```

c)
```
v = 7
ex = (-1,5,2,"","abra",8,3,v)
5 in ex
abra in ex
(2 in ex) and ("abr" in ex)
v in ex
```

Q 15 : Prévoir les résultats des expressions suivantes, puis le vérifier grâce à l'interpréteur interactif.

a) `"abba"`
b) `abba`
c) `""`
d) `"" == " "`
e) `""+""`
f) `""+"" == ""`
g) `"May"+" "+"04th"`
h) `"12"+3`
i) `"12"+"trois"`
j) `len("abracadabra")`
k) `len("")`
l) `len("lamartin"+"2015")`

Q 16 : Pour chaque séquence d'instruction, prévoir son résultat puis le vérifier grâce à l'interpréteur interactif.

a)
```
t = "oh oui youpi !"
print(t)
t[0]
```

```
t[-1]
t[1]
t[2]
t[1] = "o"
```

b)

```
ex = "abdefgh"
"a" in ex
a in ex
"def" in ex
"adf" in ex
```

Q 17 : 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 18 : Pour chaque séquence d'instruction, prévoir son résultat puis le vérifier grâce à l'interpréteur interactif d'IDLE.

a)

```
t = [1,2,3,4,5,6]
u = ["a","b","c","d"]
print(t+u)
t[0]
t[-1]
z = t[3]
print(z)
t.append(7)
print(t)
```

c)

```
ex = ["sin","cos","tan","log","exp"]
"log" in ex
log in ex
"l" in ex
z = ex.pop()
print(z)
z in ex
print(ex)
```

d)

```
u = [1,2,3,4,5,6]
L = u
u = [1,2,3,42,5,6]
print(L)
```

e)

```
u = [1,2,3,4,5,6]
L = u
u[3] = 42
print(L)
```

Q 19 : 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.

Q 20 :
a) Affecter à `v` la liste `[2,5,3,-1,7,2,1]`
b) Affecter à `L` la liste vide.
c) Vérifier le type des variables crées.
d) Calculer la longueur de `v`, affectée à `n` et celle de `L`, affectée à `m`.
e) Tester les expressions suivantes : `v[0]`, `v[2]`, `v[n]`, `v[n-1]`, `v[-1]` et `v[-2]`.
f) Changer la valeur du quatrième élément de `v`.
g) Que renvoie `v[1:3]` ? Remplacer dans `v` les trois derniers éléments par leurs carrés.
h) Que fait `v[1] = [0,0,0]` ? Combien d'éléments y a-t-il alors dans `v` ?

Q 21 : Quel type choisiriez-vous pour représenter les données suivantes ? Vous justifierez brièvement chaque réponse.
a) Le nom d'une personne.
b) L'état civil d'une personne : nom, prénom, date de naissance, nationalité.
c) Les coordonnées d'un point dans l'espace.
d) L'historique du nombre de 5/2 dans la classe de MP du lycée.
e) Un numéro de téléphone.
f) *Plus difficile :* l'arbre généalogique de vos ancêtres.

======= Évaluer les expressions suivantes.

a) `4.3+2`
b) `2.5-7.3`
c) `42+4.`
d) `42+4`
e) `42.+4`
f) `12*0.`
g) `11.7*0`
h) `2,22/(1.6-2*0.8)`
i) `42/6`
j) `1,8/7`
k) `(447+3*6)/5`
l) `0/0`

Q 22 : 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 23 : Écrire une séquence d'instructions qui échange les valeurs de deux variables x et y.

Q 24 : 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 25 : Dans les cas où c'est possible, affecter les valeurs aux variables correspondantes à l'aide de l'interpréteur interactif. On notera `var` ← `a` pour dire que l'on affecte la valeur `a` à la variable `var`.

a) `ArthurDent` ← `42`
b) `4` ← `0.`
c) `L` ← `[]`
d) `list` ← `[1,2,3]`
e) `int` ← `5`
f) `s` ← `""`
g) `True` ← `1`
h) `ok` ← `ok`
i) `x` ← `"x"`
j) `a` ← `1<0`
k) `lam` ← `1/0`
l) `or` ← `"xor"`

Q 26 : On part des affectations suivantes : `a` ← `5` et `b` ← `0`. Pour la suite d'instructions suivante, prévoir ligne à ligne le résultat affiché par l'interpréteur interactif de Python ainsi que l'état des variables. Le vérifier grâce à l'interpréteur interactif d'IDLE, en prenant soin de partir d'une nouvelle session.

```
a*b
x = a**b + a
print(x)
print(y)
z = x
x = 5
print(z)
a = a+a**b
print(a)
```

Q 27 : Affecter des valeurs toutes différentes aux variables `a`, `b`, `c` et `d`.
Á chaque fois, effectuer les permutations suivantes de manière naïve (c'est-à-dire, sans utiliser de `tuple`).

a) Échanger les contenus de `a` et de `b`.
b) Placer le contenu de `b` dans `a`, celui de `a` dans `c` et celui de `c` dans `b`.
c) Placer le contenu de `a` dans `d`, celui de `d` dans `c`, celui de `c` dans `b` et celui de `b` dans `a`.

Reprendre cet exercice en effectuant chaque permutation en une instruction à l'aide d'un `tuple`.

Q 28 : Combien d'affectations sont suffisantes pour permuter circulairement les valeurs des variables $x_1, \cdots, x_n$ sans utiliser de variable supplémentaire ? Et en utilisant autant de variables supplémentaires que l'on veut ?

Q 29 : Mêmes questions en remplaçant suffisantes par nécessaires.

Q 30 : Supposons que la variable x est déja affectée, et soit $n \in \mathbb{N}$. On veut calculer x^n sans utiliser la puissance, avec uniquement des affectations, autant de variables que l'on veut, mais avec le moins de multiplications possible. Par exemple, avec les 4 instructions :

on calcule x^5, qui est la valeur de `y4`.

Mais 3 instructions suffisent :

```
>>> y1 = x * x
>>> y2 = y1 * y1
>>> y3 = y2 * x
```

Q 31 : En fonction de n, et avec les contraintes précédentes, quel est le nombre minimum d'instructions pour calculer x^n ?

Q 32 : 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) $\tan\left(\dfrac{\pi}{2}\right)$

Q 33 :

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 34 : 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 35 : 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 36 : Écrire une fonction `lettre` qui prend en argument un entier `i` et renvoie la `i`$^{\text{e}}$ lettre de l'alphabet.

Q 37 : 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.

Q 38 : Les expressions suivantes sont-elles équivalentes ?

a) `8.5 / 2.1`
b) `int(8.5) / int(2.1)`
c) `int(8.5 / 2.1)`

Q 39 : Et celles-ci ?

a) `float(8 * 2)`
b) `8 * 2`
c) `8. * 2.`

Q 40 : Prévoir la valeur des expressions suivantes puis vérifier cela (avec IDLE).

a) `1.7 + 1.3`
b) `2 - 1`
c) `2. - 1`
d) `2 - 1.`
e) `(2 - 1).`
f) `.5 + .5`
g) `4 / (9 - 3**2)`
h) `4 / (9. - 3**2)`