TP_04 NSI 1ère -- lycée de l'Elorn 2023 --

Les instructions et les structures de base en Python

Le NoteBook Jupyter

Nous utiliserons de préférence cet environnement pour mettre en oeuvre des scripts Python. C'est un bon support pour les tests que nous conduirons lors de nos apprentissages. Ce fichier peut être converti au format html pour être sauvegardé.

Je vous conseille d'utiliser une installation en ligne grace au lien suivant :

https://capytale2.ac-paris.fr/web/c/1630-741225/ttt

Une authentification Toutatice vous sera demandée.

Le plus simple pour déployer un serveur Jupyter sur votre ordinateur personnel est d'installer **l'IDE Anaconda** sur windows, linux ou Mac. Vous trouverez aussi plusieurs applications en ligne en mesure d'exploiter ce fichier.

Les cellules si dessous sont éditables et exécutables. Vous pouvez **exécuter** une cellule à l'aide du bouton ou encore en vous servant des touches :

- "Shift + Enter" : le code de la cellule est exécuté et le curseur va à la cellule suivante.
- " Ctrl + Enter" : le code de la cellule est exécuté et le curseur reste sur la même cellule.
- " Alt + Enter" : le code de la cellule est exécuté et le notebook crée une nouvelle cellule immédiatement après.

Pour effacer toutes les sorties : Menu Cell -> All output -> Clear

La fonction print()

In [110…

```
print("Bonjour les NSI")
print("Vous avez vu comment cela fonctionne ?")
```

```
Bonjour les NSI
Vous avez vu comment cela fonctionne ?
```

In [111…

```
#l'argument end="xxx" permet de préciser ce qui se passe après l'affichage
print("Bonjour les NSI", end=" ---> ")
print("Vous avez vu comment cela fonctionne ?")
```

```
Bonjour les NSI ---> Vous avez vu comment cela fonctionne ?
```

In [112…

```
# /n pour faire une mise à la ligne  et /t  pour avoir une tabulation
print("Bonjour les NSI \n\nVous avez vu comment cela fonctionne ? \n \t oui ? \n\t non ?\n\nC'est pratique non ?")
```

```
Bonjour les NSI

Vous avez vu comment cela fonctionne ?
         oui ?
         non ?

C'est pratique non ?
```

In [113…

```
#c'est la même chose que :
print("Bonjour les NSI ")
print("")
print("Vous avez vu comment cela fonctionne ?")
print("         oui ?")
print("         non ?")
print("")
print("C'est pratique non ?")
#mais c'est moins élégant comme programmation ...
```

```
Bonjour les NSI

Vous avez vu comment cela fonctionne ?
         oui ?
         non ?

C'est pratique non ?
```

In [114…

```
#Si vous trouvez que le print est trop long utilisez le caractère \ pour aller à la ligne dans le print :
print("Bonjour les NSI \n\nVous avez vu comment cela fonctionne ? \
\n \t oui ?\n\t non ?\n\nC'est pratique non ?")
```

```
Bonjour les NSI

Vous avez vu comment cela fonctionne ?
         oui ?
```

```
        non ?

C'est pratique non ?
```

In [115…

```python
#Pour écrire le résultat d'une opération :
print("2000+20 =", 2000+20)
print("50 en binaire s'écrit :", bin(50))
print("50 en hexadécimal s'écrit :", hex(50))
print("0xF0 en décimale s'écrit :", int(0xABCD))
```

```
2000+20 = 2020
50 en binaire s'écrit : 0b110010
50 en hexadécimal s'écrit : 0x32
0xF0 en décimale s'écrit : 43981
```

A VOUS : Créer, à l'aide d'un seul print(), le code permettant d'afficher ce message en utilisant les fonctions bin(), hex() et int()

Voici la conversion de quelques valeurs décimales :

```
10 en décimale s'écrit 0b1010 en binaire
1111 en binaire s'écrit 0xF en hexadécimale
ABCD en hexadécimale s'écrit 43981 en décimale
```

Variables et types primitifs

Python fournit des types de base, dits types primitifs :

- int : entier
- float : réel (nombre à virgule)
- str : string (chaine de caractère)
- bool : booléen (True ou False)

En Python, les variables sont auto-déclarées. L'affectation se fait avec le symbole "=". Ainsi pour définir une variable note qui aurait pour valeur 12, il suffit de taper : note=12

In [116…

```python
# Entier
n = 2020
print(n)
type(n)
```

```
2020
```

Out[116…
```
int
```

In [117…
```
# Réel
y = 0.25
print(y)
print(type(y))
```

```
0.25
<class 'float'>
```

In [118…
```
# Chaine de caractère
s = "Lycée Elorn"
print(s)
type(s)
```

```
Lycée Elorn
```

Out[118…
```
str
```

In [119…
```
#En Python, les chaines de caractère peuvent aussi bien s'écrire entre simples guillemets (') qu'entre doubles guillemets (").

s1 = "Je suis au Lycée "
s2 = 'avec mon prof de NSI'

print(type(s1))
print(type(s2))
```

```
<class 'str'>
<class 'str'>
```

In [120…
```
#On peut concaténer deux chaines grâce à l'opérateur +
s1 = "Je suis au Lycée "
s2 = "avec mon prof de NSI."
s3 = " Je suis content(e)."
print(s1+s2+s3)
```

```
Je suis au Lycée avec mon prof de NSI. Je suis content(e).
```

In [121…
```
#l'opérateur * fonctionne avec les chaines de caractère.
s = " Je suis content(e)."
print(5*s)
```

```
Je suis content(e). Je suis content(e). Je suis content(e). Je suis content(e). Je suis content(e).
```

In [122…

```python
#Il est possible d'afficher le contenu de plusieurs variables (quel que soit leur type) en les séparant par des virgules :
nom="Jojo"
age=17
taille=185
print(nom, "est agé de", age, "ans et mesure", taille, "cm")
```

```
Jojo est agé de 17 ans et mesure 185 cm
```

In [123…

```python
#l'écriture formatée permet une meilleure organisation de l'affichage des variables
nom="Jojo"
age=17
taille=185
print("{} est agé de {} ans et mesure {} cm".format(nom, age, taille))
```

```
Jojo est agé de 17 ans et mesure 185 cm
```

In [124…

```python
#même chose que ci-dessus avec des numéros  0 pour afficher en 1er ... etc
print("{0} est agé de {2} ans et mesure {1} cm".format(nom, taille, age))
```

```
Jojo est agé de 17 ans et mesure 185 cm
```

In [125…

```python
#la fonction len() permet de connaitre la taille d'une chaine de caractère.
s = "Je suis au Lycée avec mon prof de NSI"
print("il y a ", len(s),"caractères dans cette variable")
```

```
il y a  37 caractères dans cette variable
```

In [126…

```python
# Booléen
a = True  # La majuscule est importante !
type(a)
```

Out[126…

```
bool
```

Les opérateurs de base en python

```
+   addition
-       soustraction
*       multiplication
/   division        2/2 donne 2.5
**  puissance       2**10 donne 1024
//  division entière    5//2 donne 2
%   reste de la division entière   5%2 donne 1

==  égalité  (à ne pas confondre avec l'affectation)
!=  différent
<, >, <=, >=   inférieur, supérieur, inférieur ou égal, supérieur ou égal

and opérateur booléen ET
or  opérateur booléen OU
not opérateur booléen NON
```

In [100…
```
3+2
```

Out[100…
```
5
```

In [101…
```
3/2
```

Out[101…
```
1.5
```

In [102…
```
3//2
```

Out[102…
```
1
```

In [103…
```
1<2
```

Out[103…
```
True
```

In [104…
```
1>2
```

Out[104… False

In [ ]:
```
1==2
```

In [105…
```
1!=2
```

Out[105… True

In [106…
```
n=10
print(0<n<20)
```

True

In [107…
```
n=10
print(0<n<10)
```

False

In [108…
```
n=10
print((n>0) and (n<20))
```

True

In [109…
```
a=True
b=not(a)
print(b)
```

False

Conversion de type

Grâce aux fontions int, float et str, on peut convertir le type d'une variable.

In [99]:
```
x = "1"
print("x est de type: ", type(x))
x = int(x)
print("x est de type: ", type(x))
```

```
x est de type:  <class 'str'>
x est de type:  <class 'int'>
```

La fonction input()

Sert à saisir une valeur depuis le clavier par exemple

In [ ]:
```
nom=input("Quel est ton nom mon grand ?")
print(nom)
```

In [ ]:
```
#Attention la fonction input() renvoie une chaine de caractère ! Ci-dessous un bug classique, tout est écrit dans le TypeError
age=input("Quel est ton age cette année ?")
print("Dans 10 ans tu auras ", age + 10)
```

A VOUS :

- Débrouillez-vous pour pouvoir afficher le résultat correct dans le print ci-dessus.
- Créez un programme qui demande le prénom d'un utilisateur puis son âge afin d'écrire la chaîne de caractères suivante :
  « Bonjour Jojo, tu es donc né(e) en 2006 » (si la personne s'appelle Jojo et qu'elle a 17 ans).

Les tests

La syntaxe pour effectuer un test en python est la suivante :

```
if *condition* :
    instructions à réaliser
else:
    instructions à réaliser
reste du programme
```

Remarque : On notera l'importance de l'indentation

A VOUS :

Créer une variable *age* contenant votre age puis écrire un test avec les conditions suivantes :

- Si age < 10, alors écrire : "Vous êtes trop jeunes pour démarrer python"
- Si age < 14, alors écrire : "Vous pouvez commencer par scratch"
- Sinon écrire : "Vous pouvez découvrir Python"

Remarque : On pourra utiliser l'instruction elif : lien vers un page expliquant les tests

In [ ]:

```
```

Les boucles

La structure "REPETER TANT QUE" (WHILE)

Dans cette structure on commence par tester la condition qui fera quitter la boucle de répétition

In [ ]:

```
mot_de_passe=""

while mot_de_passe!="bidule":
     mot_de_passe=input("mot de passe svp : ")

print("le mot de passe est validé")
```

La structure "POUR ... DE ... A ... " (FOR)

Lorsque l'on souhaite répéter un nombre donné de fois la même instruction ou le même bloc d'instructions, la commande for est la plus appropriée.

On utilisera souvent la fonction **range()** dans le FOR.

La fonction **range()** permet de créer une liste de valeurs :

- range(n) va créer une liste de n nombres entiers démarrant à 0 (en se terminant à n-1)
- range(1,n) va créer une liste de nombres entiers démarrant à 1 et se terminant à n-1.
- range(1,n,2) va créer une liste de nombres entiers démarrant à 1 et se terminant à n-1 mais progressant par pas de 2.

In [ ]:

```
# Exemple : On compte de 1 à 9
for i in range(1,10):
    print(i)
```

A VOUS : Afficher la liste des entiers pairs inférieurs à 15

In [ ]:

A VOUS : Afficher la liste des multiples de 7 inférieurs à 100

In [ ]:

A VOUS : Avec la structure FOR puis avec la structure WHILE, créer une affichage de la table de multiplication de 7 sous cette forme :

```
TABLE DE 7

0  x 7 = 0
1  x 7 = 7
2  x 7 = 14
...
10 x 7 = 70
```

In [ ]:

4. Manipulation d'images

Nous allons essayer de reproduire les drapeaux de quelques nations. Pour cela nous utiliserons la bibliothèque PIL et le code départ sera le suivant :

In [ ]:

```
#Ce code crée une image rectangulaire, de dimension 300*200 et de couleur noire.

from PIL import Image
image_1=Image.new('RGB',(300,200),'black')
image_1
```

In [ ]:

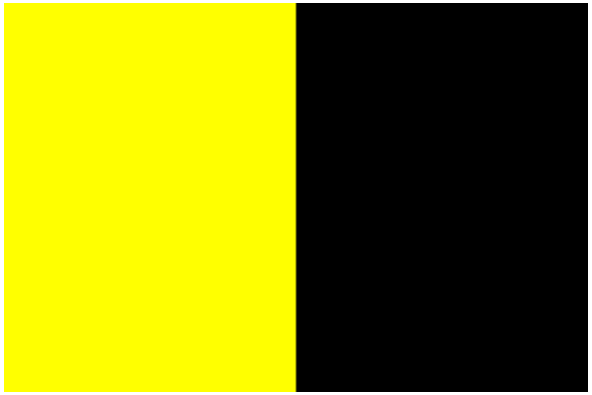
```
#Ce code crée une image de couleur noire puis y change tous les pixels en jaune.

from PIL import Image
image2=Image.new('RGB',(300,200),'black')
#on parcours
for x in range(300):
```

```
    for y in range(200):
        image2.putpixel((x,y),(255,255,0))     #la fonction putpixel change la valeur de la couleur du pixel à la position (x,y)
image2
```

In [127…

```
#Change la couleur de la première moitié de l'image
from PIL import Image
image3=Image.new('RGB',(300,200),'black')
for x in range(150):
    for y in range(200):
        image3.putpixel((x,y),(255,255,0))
image3
```

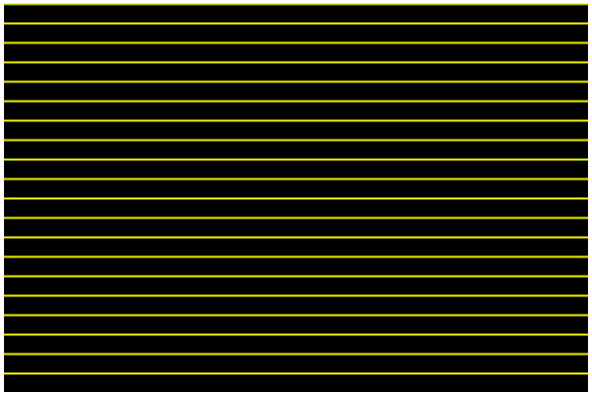
Out[127…



In [128…

```
#Change la couleur de remplissage en jaune d'une ligne sur 10
from PIL import Image
image4=Image.new('RGB',(300,200),'black')
for x in range(300):
    for y in range(0,200,10):
        image4.putpixel((x,y),(255,255,0))
image4
```

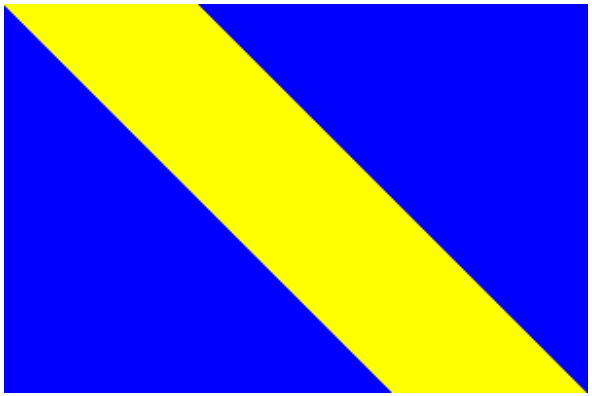
Out[128…



In [129…

```
#Dans cet exemple je vous montre commment créer un remplissage en diagonale
#l'avant dernière ligne de code permet de sauver l'image dans un fichier.
from PIL import Image

image5=Image.new('RGB',(300,200),'blue')
d=0
for y in range(0,200):
    for x in range(d,100+d):
        image5.putpixel((x,y),(255,255,0))
    d+=1
image5.save("mon_drapeau.jpg")
image5
```

Out[129…

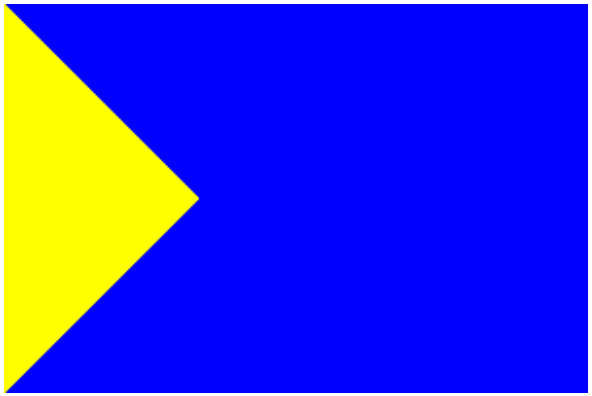


In [130…

```
# Un dernier exemple permettant de créer un triangle sur le coté gauche
from PIL import Image

image6=Image.new('RGB',(300,200),'blue')
```

```
d=0
for x in range(0,300):
    for y in range(d,200-d):
        image6.putpixel((x,y),(255,255,0))
    d+=1
image6
```

Out[130…



A VOUS : Pour chaque drapeau vous créerez une nouvelle variable (drapFr, drapAl, ...)

Vous trouverez des informations sur les couleurs ici.

Créer les drapeaux des pays suivants :

- France
- Belgique
- Italie
- Allemagne
- Autriche
- Hongrie
- Finlande
- Suède
- Norvège
- Islande
- Grèce
- Royaume Uni

In [ ]: