1 La console

Dans la console, on peut taper directement des instructions Python qui s'exécutent immédiatement à condition d'appuyer sur <Entrée> en fin de ligne.

- Lorsque vous avez la main dans la console, le curseur est précédé de trois chevrons >>> ou `[1]:` ....
- La division s'obtient avec « / ».
- La division entière (quotient de la division euclidienne) s'obtient avec « // »
- Le reste de la division euclidienne s'obtient avec « % ».
- La puissance s'obtient par « ** ».
- On peut ajouter des commentaires en commençant par un « # » : à utiliser sans modérations pour s'y retrouver dans un programme.
- On peut aussi écrire une suite d'instructions en les séparant par « ; »

Anticiper le résultat de chaque instruction dans la console et vérifier le résultat affiché.

```
>>>3+5
>>>3*5
>>>15/4
>>>15//4
>>>15%4
>>>2**3
```

Remarque : Dans la console, après l'exécution d'une instruction, on peut "voir" la valeur de la dernière instruction.

Un variable sert à stocker une information qui peut être sous la forme d'un nombre, une phrase, une liste de nombres, une liste de mots ....
L'affectation des variables dans python se fait avec = et l'affichage du contenu d'une variable se fait avec l'instruction `print`.
Ainsi `a=1` correspond à l'instruction `stocker la valeur 1 dans la variable a` autrement dit la variable `a` prend la valeur 1 que l'on écrit de façon synthétique $a \longleftarrow 1$.
Le nom d'une variable ne peut pas être un nombre, ne doit pas contenir ni accent, ni espace, ne doit pas commencer par un nombre ; de plus,il y a des mots réservés qui prennent une coloration différente quand on les écrit. Il peut être utile de donner des noms explicites dans un programme pour s'y retrouver.
Exemples de nom de variables : `M`, `L1`, `ma_Valeur2`, `ma_photo_25`,...

Exercice pour bien comprendre ce qu'est une variable

1. On considère l'algorithme ci-dessous. Anticiper le résultat afficher en détaillant les étapes.

$x \longleftarrow 3$
$y \longleftarrow 2x+4$
$z \longleftarrow y^2$
afficher(z)
	`>>>x=3`
	`>>>y=2*x+4`
	`>>>z=y**2`
	`>>>print(z)`

2. On considère l'algorithme ci-dessous. Anticiper le résultat afficher en détaillant les étapes.

$x \longleftarrow 3$
$x \longleftarrow 2x+4$
$x \longleftarrow x^2$
afficher(x)
	`>>>x=3`
	`>>>x=2*x+4`
	`>>>x=x**2`
	`>>>print(x)`

Anticiper le résultat de chaque instruction dans la console et vérifier le résultat affiché.

```
>>>a=3
>>>b=5
>>>a+b
>>>a=a+1 ; print(a)
>>>b=a  ; print(b)
>>>a =1; b=2 ; print(a) ; print(b)
>>>A=3; print(a) # Python est sensible à la case.
>>>print("Bonjour tout le monde !")
```

Il existe d'autres fonctions plus complexes mais pour les utiliser, il faut importer un module : sorte de "collection" de fonctions. Par exemple, le module `math` pour les fonctions en mathématiques, le module `csv` pour manipuler des données en CSV, le module `PIL` pour manipuler des photos numériques, le module `follium` pour la géolocalisation, le module `micropython` pour l'informatique embarquée ...
On peut importer un module de plusieurs façons :

```
import nomDuModule
# on importe toutes les fonctions
# ensuite il faut appeler la  fonction comme ceci : nomDuModule.nomFonction

import nomDuModule as alias
# on importe toutes les fonctions
#ensuite il faut appeler la  fonction comme ceci : alias.nomFonction

from nomDuModule import*
# on importe toutes les fonctions
#ensuite il faut appeler la  fonction comme ceci : nomFonction

from nomDuModule import nomFonction
# on importe uniquement la fonction nommée et on l'appelle par son nom
```

Pour obtenir de l'aide sur un module dans la console Python, il faut d'abord l'importer avec `import nomDuModule` puis taper `help(nomDuModule)`, mais le mieux est encore de consulter la documentation sur internet : http ://docs.python.org/3/.

Un exemple avec le module `webbrowser`

```
>>>import webbrowser
>>>webbrowser.open('https://www.openstreetmap.org/#map=19/49.17860/-0.38191')
```

2 L'éditeur

L'éditeur sert à taper des programmes en Python, on parle aussi de scripts que l'on pourra sauvegarder dans des fichiers dont l'extension est `.py`
Dans l'éditeur, taper le programme ci-dessous, observer la coloration syntaxique, la complétion ...
Sauvegarder ce script dans un dossier de votre choix.
Dans la suite, vous enregistrerez tous vos scripts dans ce dossier. Essayer de donner un nom explicite à vos dossiers et fichiers, en évitant les lettres accentuées et les espaces ; par exemple : `mon_essai.py` est préférable à `mon essai.py` .

```
print("Bonjour")
x=42
print(x)
```

Dans la plupart des environnements, on peut exécuter un script en passant par la barre de menu : `Exécuter` ou `Run` ...
Dans Pyzo, on peut exécuter un script avec la combinaison Ctrl-Maj-E.

Un exemple avec le module `folium`
Écrire ce programme dans un script, le sauvegarder dans votre dossier. Puis ouvrir le fichier `lieuConnu.htlm` que vous trouverez dans le dossier où vous avez sauvegardé vos scripts.

```
import folium
coordGPS=(49.1785935,-0.3819152)
carte=folium.Map(location=coordGPS,zoom_start=20)
folium.Marker(coordGPS).add_to(carte)
carte.save('lieuConnu.html')
```

3 Types de variables

Python est un langage typé et contient une grande variété de types différents.

Taper chaque instruction dans la console.

```
>>>type(42)
>>>type(3.5)
>>>type("bonjour" )
>>>n = 3 ; type(n)
>>>x = 3.0 ; type(x)
>>>test = 2<5 ; type(test)
>>>T =(1,2) ; type(T)
>>>L=[1,2] ; type(L)
```

3.1 Les entiers : le type `int`

Ils supportent les opérations +, −, ∗, /, ∗∗ (puissance), // (division entière), % (reste de la division entière), `abs()` (valeur absolue) ... Les ordres de priorité sont les mêmes qu'en mathématiques.

3.2 Les nombres flottants : le type `float`

Ils supportent les opérations usuelles. On peut convertir des expressions d'un type à un autre.

Anticiper le résultat de chaque instruction dans la console et vérifier le résultat affiché.

```
>>>float(42)
>>>int(42.1)# renvoie la troncature.
>>>int(8.5)
>>>8.5 / 2.5
>>>int(8.5)/int(2.5)
>>>int(8.5/2.5)
```

3.3 Les booléens : le type `bool`

Ils ne peuvent prendre que deux valeurs : `False` ou `True`.
`False` a pour valeur 0 et `True` a pour valeur 1 : on peut les ajouter !
Ils sont générés par les opérateurs de comparaison <, >, <=, >=, == (test d'égalité), ! =(test de différence) et les opérateurs logiques `not, or` et `and. A or B` est vraie si au moins une des deux propriétés est vraie.

Anticiper le résultat de chaque instruction dans la console et vérifier le résultat affiché.

```
>>>x=16
>>>y=-2
>>>x==y
>>>x==16
>>>x!=5
>>>a=True ; b=False
>>>c=not a ;print(c)
>>>print(a and b)
>>>print(a or b)
>>>print(a and not b)
```

3.4 Les uplets : le type `tuple`

Les **tuple** contiennent des éléments qui peuvent être de type quelconque, éventuellement de types différents. Ils sont délimités par des parenthèses ( ) et les éléments sont séparés par une virgule. Chaque élément possède un indice, et le premier élément porte l'indice 0, le deuxième porte l'indice 1 ... On peut repérer un élément en commençant par la fin : le dernier porte l'indice −1, l'avant dernier porte l'indice −2, l'antépénultième porte l'indice −3 ...
Un tuple est un objet non mutable, on ne peut pas modifier la valeur d'un élément, ajouter ou supprimer des éléments à un tuple. On peut concaténer 2 tuple, compter les occurrences d'un élément, ou le nombre d'éléments du tuple, tester l'appartenance d'un élément au tuple ...

Anticiper le résultat de chaque instruction dans la console et vérifier le résultat affiché.

```
>>>T1=(1,2,3); T2=(4,5)
>>>print(T1[0])
>>>print(T1[-2])
>>>len(T1) #renvoie le nombre d'éléments de T1
>>>print(T2[3])
>>>T3=T1+T2; T4=T2*2; print(T3);print(T4)
>>>coordGPS=(49.1785935,-0.3819152)
>>>latitude=coordGPS[0]; longitude=coordGPS[1]
>>>print(latitude)
```

Si `t` est un tuple, on peut utiliser les fonctions `min(t)`, `max(t)`, `sum(t)`, `sorted(t)` ... lorsque cela a un sens.
On peut aussi tester la présence d'un élément `val` dans un tuple `t` avec l'opérateur `val in t` qui renvoie `True` ou `False`.

Remarque : Sous python, certaines fonctions font partie de la "définition" des objets, on parle alors de méthode propre à l'objet. La syntaxe est alors un peu différente d'une fonction "classique" : `objet.methode(...)`

Un tuple possède beaucoup de méthodes, en voici deux :
⋆ la méthode `index` qui renvoie le plus petit index d'un élément : `t.index(val)`
⋆ la méthode `count` qui renvoie le nombre d'occurrences d'un élément : `t.count(val)`.

Anticiper le résultat de chaque instruction dans la console et vérifier le résultat affiché.

```
>>>t=(23,4,15,8,16,8,42)
>>>print(min(t)) ; print(max(t))
>>>sum(t)
>>>sorted(t) #renvoie une liste triée dans l'ordre croissant
>>>13 in t
>>>t.index(8)
>>>t.count(8)
```

3.5 Les chaînes de caractères : le type `string`

Une chaîne de caractère est donnée entre guillemets (simples, doubles ou triples). Les caractères peuvent être des lettres, des nombres, un espace ...
Pour définir une chaîne de caractères, on utilise :

- soit les apostrophes par exemple `'Il a dit : "bonjour"'`
- soit les guillemets par exemple `"SNT, c'est génial!"`
- soit des triples guillemets qui permettent de mettre tout ce qu'on veut dans la chaîne de caractères par exemple `"""SNT, c'est pour tous ;)"""` .

Une chaîne de caractères est un objet non mutable. Toutes les opérations vues sur les tuple sont aussi valables sur les chaînes de caractères.

```
>>>chaine="bonjour tout le monde !"
>>>chaine[0]
>>>chaine[7]
>>>chaine[-1]
>>>len(chaine)
>>>chaine1="bon"; chaine2="jour"; chaine3=chaine1+chaine2; print(chaine3)
>>>chaine4=motchaine1*3; print(chaine4)
>>>chaine.count('o')# renvoie le nombre d'occurrence de la lettre o
```

3.6 Les listes : le type `list`

Les **listes** contiennent des éléments qui peuvent être de type quelconque, éventuellement de types différents. Elles sont délimitées par des crochets [ ], et les éléments sont séparés par une virgule. Comme pour les tuple, chaque élément d'une liste possède un indice, le premier élément porte l'indice 0 et le dernier porte l'indice −1. Les listes sont des objets mutables. On peut modifier la valeur de chaque élément, ajouter ou supprimer des éléments à la liste. On peut concaténer 2 listes, trier les éléments, compter les occurrences d'un élément, ou le nombre d'éléments de la liste, tester l'appartenance d'un élément à la liste, etc.

Anticiper le résultat de chaque instruction dans la console et vérifier le résultat affiché.

```
>>> L1=[1, 1.2, 8.3, 4, 5]; L2=[5, 6, 7]
>>> L1[0]
>>> L1[0]=0 ; L1
>>> len(L1)
>>> L3=L1+L2; L3
>>> L1[0]=42 ; L3
>>> L4=L2*2; L4
>>> max(L3) ; min(L3)
>>> L3.count(5)
>>> L3.reverse() ; L3
>>> del L3[3] ; L3
>>> 8 in L3 ; 9 in L3
```

Méthodes pour les listes

- La méthode `append()`ajoute un élément en fin de liste : `L.append(element)`
- La méthode `insert()` permet de spécifier l'indice où insérer un élément : `L.insert(indice, element)`
- La méthode `extend()` permet de réaliser la concaténation de deux listes sans réaffectation : `L1.extend(L2)` au lieu de L3=L1+L2 mais dans ce cas L1 est modifiée.
- La méthode `remove()` supprime la première occurrence de la valeur passée en paramètre : `L.remove(element)`
- La méthode `pop()` supprime l'élément dont l'indice est passée en paramètre : `L.pop(indice)`
- La méthode `sort()` permet de trier la liste :`L.sort()`
- La méthode `reverse()` permet d'inverser l'ordre des éléments :`L.reverse()`
- La méthode `index()` possible d'obtenir l'indice de la première occurrence d'un élément par la méthode : `L.index(element)`

Anticiper le résultat de chaque instruction dans la console et vérifier le résultat affiché.

```
>>>L= [12, 11, 18, 7, 15, 3];L.append(5)
>>>L.insert(2,1); print(L)
>>>L1=[1,2,3]; L2=[4,5,6]
>>>L1.extend(L2); print(L1)
>>>L=[1,2,1,3,1]; L.remove(1); print(L)
>>>L.remove(1)
>>>L=[1,2,3,1]
>>>L= [12, 11, 18, 7, 15, 3]; L.sort(); print(L)
>>>L=[1,2,1,3,1]; L.reverse(); print(L)
```

Remarques sur la copie de listes :
Pour copier une liste, on ne peut pas simplement écrire L2=L1, sinon toute modification de l'une entraîne une modification de l'autre. Les listes L1 et L2 pointent vers la même zone mémoire.

```
>>>L1=[1, 2, 3] ; L2=L1 ; print(L2)
>>>L2[0]=0 ; print(L2) ; print(L1)
```

Une première solution pour effectuer une copie peut être d'utiliser le slicing.
L'opération [:] renvoie une nouvelle liste :

```
>>>L1=[1, 2, 3]; L2=L1[:] ; print(L2)
>>>L2[0]=0 ; print(L2) ; print(L1)
>>>L3=L1[1:] ; print(L3)
```

Cette copie peut paraître satisfaisante et elle l'est, mais à condition de ne manipuler que des listes de premier niveau ne comportant aucune sous-liste :

```
>>>L1=[1,2,3,[4,5,6]] ; L2=L1[:]
>>>L2[3][0]=0 ;  print(L2) ; print(L1)
```

3.7 Les chaînes et les listes

⋆ La fonction `list()` permet de convertir une chaîne en liste, par exemple :

```
>>>chaine = "abcdef"
>>>liste = list(chaine1)
>>>liste
['a', 'b', 'c', 'd', 'e', 'f']
```

⋆ On peut appliquer la méthode `split` à une chaîne :
`chaine.split()` permet de récupérer une liste contenant les éléments de l'objet `chaine` séparés par le caractère espace `' '` ; dans cette liste n'apparaît pas le caractère `\n` (qui indique un saut de ligne) et ses éléments sont de type `str` ; `chaine.split(',')` permet de récupérer une liste contenant les éléments de l'objet `chaine` séparés par le caractère espace `','` .

Exemple :

```
>>>chaine="Le texte a deux phrases.\nLa deuxième phrase a six mots."
>>>print(chaine)
>>>L=chaine.split()
>>>print(L)
```

`chaine.split(',')` permet de récupérer une liste contenant les éléments de l'objet `chaine` séparés par le caractère espace `','` ; dans cette liste n'apparaît pas le caractère `\n` et ses éléments sont de type `str`.

Exemple d'utilisation des listes pour décoder une trame NMEA* :

```
>>>trame="$GNGGA,065020.000,4910.7156,N,0022.9150,W,1,5,1.68,-14.6,M,47.1,M,,*48"
>>>L=trame.split(',')
>>>print(L)
  ['$GNGGA', '065020.000', '4910.7156', 'N', '0022.9150', 'W', '1', '5', '1.68',
  '-14.6', 'M', '47.1', 'M', '', '*48']
>>>
```

Ainsi `L[2]` qui contient la valeur `'4910.7836'` permet de récupérer après manipulation la latitude sous forme d'une chaîne qu'il faudra ensuite transformer en `float` si on veut par exemple la convertir en degrés, minutes, secondes.
Dans cet exemple d'une trame GNGGA (GN : GPS +GLONASS et GGA : syntaxe), on a une latitude de 49 degrés 10.7156 minutes et une longitude de −0 degrés 22.9150 minutes.

∗ NMEA : National Marine Electronics Association.
La norme NMEA 0183 est une spécification pour la communication entre équipements marins, dont les équipements GPS. Une trame NMEA est une suite de caractères contenant des informations de géolocalisation comme la date, la latitude, la longitude, la vitesse ....