Initiation générale - Rappels de L1
2024 - 2025

1er avril 2025

1 Le langage python et son utilisation.

1.1 Python

Python est un langage puissant, à la fois facile à apprendre et riche en possibilités. Dès l'instant où vous l'installez sur votre ordinateur, vous disposez de nombreuses fonctionnalités intégrées au langage que nous allons découvrir tout au long de ce document.

Il est, en outre, très facile d'étendre les fonctionnalités existantes, comme nous allons le voir. Ainsi, il existe ce qu'on appelle des bibliothèques qui aident le développeur à travailler sur des projets particuliers. Plusieurs bibliothèques peuvent ainsi être installées pour, par exemple, développer des interfaces graphiques en Python.

Concrètement, voilà ce qu'on peut faire avec Python :

- de petits programmes très simples, appelés scripts, chargés d'une mission très précise sur votre ordinateur ;
- des programmes complets, comme des jeux, des suites bureautiques, des logiciels multimédias, des clients de messagerie...
- des projets très complexes, comme des progiciels (ensemble de plusieurs logiciels pouvant fonctionner ensemble, principalement utilisés dans le monde professionnel).

Voici quelques-unes des fonctionnalités offertes par Python et ses bibliothèques :

- créer des interfaces graphiques ;
- faire circuler des informations au travers d'un réseau ;
- dialoguer d'une façon avancée avec votre système d'exploitation ;

En effet python offre de nombreuses possibilités, et en particulier grâce à ces nombreuses bibliothèques. Les bibliothèques python sont des modules de python qui vous donnent accès à de nombreuse fonctionnalité de toutes sortes. Voici quelques exemples des bibliothèques que vous serez amener à utiliser :

- numpy : Bibliothèque spécialisée dans le calcul matriciel et la manipulation de tableaux.
- math : Bibliothèque donnant accès à des fonctions mathématiques en tous genre.
- scipy : Bibliothèque consacrée principalement au calcul scientifique.
- matplotlib : Bibliothèque très riche de création de graphiques en 2D et 3D.
- panda : Bibliothèque pour le traitement de données expérimentales.
- time : Bibliothèque pour le traitement du temps
- os : Bibliothèque qui vous permet d'interagir avec votre système d'exploitation de votre ordinateur

1.2 Spyder

Spyder est un environnement d'exploitation pour python comme il en existe d'autre. C'est l'environnement que vous allez utiliser lors de vos futur TP car il est facile à prendre en main et offre de nombreuses possibilités de traitement.

Spyder est composé de trois fenêtres principales :

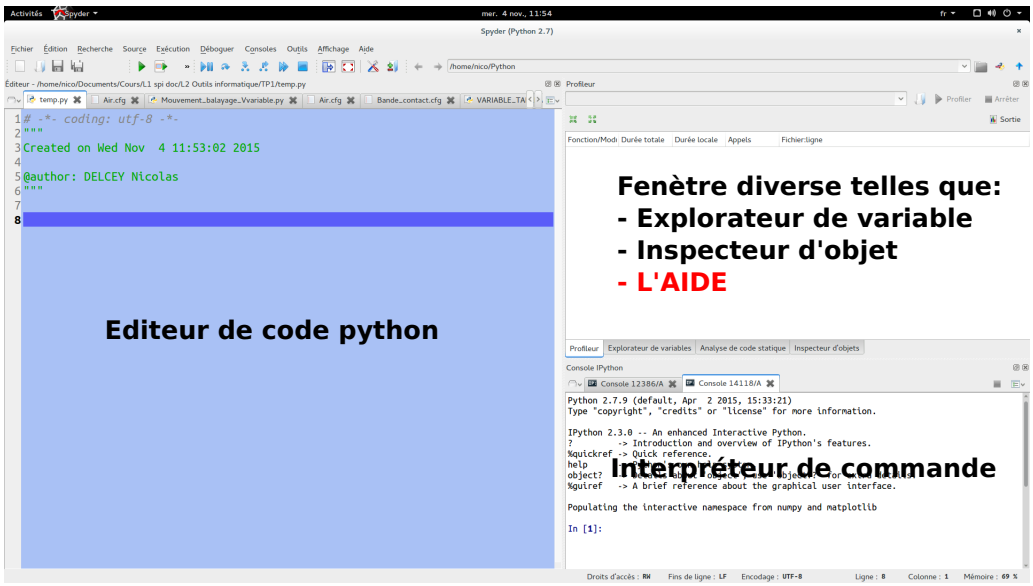

FIGURE 1 – Spyder

L'Editeur de code :

Cette fenêtre vous permet de taper des programme informatiques long et complexe, pour finalement exécuter l'intégralité de votre programme après que vous l'ayez terminer.

L'interpréteur de commande :

Cette fenêtre et une console IPython (Interactive Python) qui permet :

- Les calculs simples
- L'affichage des résultats
- Les développement python courts
- etc....

Les lignes de code tapées dans l'interpréteur IPython sont exécutées séparément et non pas sous la forme d'un programme comme avec l'Editeur de code.

2 Premier pas avec python.

2.1 Premier pas dans l'interpréteur :

L'interpréteur est comme une calculatrice avec des fonctionnalités amplement plus évoluées. Vous pouvez donc effectuer :

- Des additions : $2 + 3 = 5$
- Des soustractions : $3 - 2 = 1$
- Des multiplications : $3 * 2 = 6$
- Des divisions : $3/2 = 1$
- Des divisions à résultat flottant : $3./2. = 1.5$
- Des divisions avec partie entière : $3//2 = 1$
- Obtenir le reste d'une division : $3\%2 = 0.5$
- Des puissances : $3 * *2 = 9$

Exercice 1 :
Tapez l'équation suivante en langage python et donnez le résultat :

$$\left(\frac{5+2}{3\times 18}\times\left(18^5\right)\right)^2 - Reste\,(5/2)\times 500^2 \tag{1}$$

2.2 Les variables dans python :

2.2.1 Les bases

Une variable est une donnée de votre programme, stockée dans votre ordinateur. C'est un code alpha-numérique que vous allez lier à une donnée de votre programme, afin de pouvoir l'utiliser à plusieurs reprises et faire des calculs un peu plus intéressants avec.

Ainsi, si je veux associer mon âge à une variable, la syntaxe sera :

$$\texttt{mon\_age} = 24$$

- Nom de la variable : `mon_age`
- Valeur de la variable : 24

Exercice 2 :
Déclarer les variables : `a, b, c, d` et `e` comme étant respectivement égale à 5, 2, 3, 18 et 500. Puis reprenez l'équation de l'exercice 1 mais avec ces variables.

L'explorateur de variables (Fenêtre en haut à droite vous permet de visualiser l'ensemble de vos variables)

De plus vous pouvez affecter à vos variables la valeurs d'autres variables :

$$\texttt{mon\_age\_new} = \texttt{mon\_age} + 1$$

Ici, la variable `mon_age_new` est égale à la valeur de `mon_age` c'est à dire 24 + 1, donc `mon_age_new` =25.

2.2.2 Les types de variable

Les nombres entiers Python différencie les entiers des nombres à virgule flottante.

Initialement, c'est pour une question de place en mémoire mais, pour un ordinateur, les opérations que l'on effectue sur des nombres à virgule ne sont pas les mêmes que celles sur les entiers, et cette distinction reste encore d'actualité de nos jours.

Le type entier se nomme `int` en Python (qui correspond à l'anglais « Integer », c'est-à-dire entier). La forme d'un entier est un nombre sans virgule.

$$\texttt{a} = 3 \Longrightarrow \texttt{a est un integer}$$

Les nombre flottant Les flottants sont les nombres à virgule. Ils se nomment `float` en Python. La syntaxe d'un nombre flottant est celle d'un nombre à virgule (Sur python les nombres à virgule se forme avec un point : 2.05 et non pas $2,05$). Si ce nombre n'a pas de partie flottante mais que vous voulez qu'il soit considéré par le système comme un flottant, vous pouvez lui ajouter une partie flottante de 0 (exemple : 52.0 est donc un flottant).

$$\texttt{a} = 3.152 \Longrightarrow \texttt{a est un flottant}$$

Les nombres après la virgule ne sont pas infinis. En informatique nous sommes limités par la mémoire, mais globalement la précision est suffisante pour la majorité des cas.

Les chaînes de caractères Les types de données disponibles en Python ne sont pas limités aux seuls nombres. Le dernier simple type est la chaîne de caractères. Ce type de donnée permet de stocker une série de lettres, une phrase.

On peut écrire une chaîne de caractères de différentes façons :

- entre guillemets (`"ceci est une chaîne de caractères"`) ;
- entre apostrophes (`'ceci est une chaîne de caractères'`) ;
- entre triples guillemets (`"""ceci est une chaîne de caractères"""`).

On peut, à l'instar des nombres stocker une chaîne de caractères dans une variable

- `a="Ma chaine"`

Fonction `type` et `print` : La fonction `type` dans python vous fournira le type de la variable que vous utilisez :

- `type(a)` ⤇ `str`
- `b=2.0`
- `type(b)` ⤇ `float`

La fonction `print` vous permet d'écrire vos différente variable :

- `print("la variable a est égale à:",a)`
- `print("la variable a est égale à:",2.0)`

Fonction `input` : La fonction `input` est une fonction qui va caractériser nos premières interactions avec l'utilisateur : le programme réagira différemment en fonction du nombre saisi par l'utilisateur. Elle vous permet de déclarer une variable sans valeur. Cette valeur l'utilisateur la rentrera une fois votre programme en fonctionnement.

Exemple :

```
Ch=input("Veuillez entrer un nombre positif quelconque :")
```

Résultat dans l'interpréteur de commandes :

```
Veuillez entrer un nombre positif quelconque :
```

Pour obtenir une valeur numérique il faudra utiliser `eval(input(' ... '))`

2.3 Exercice récapitulatif :

A faire dans l'éditeur de code.

Prenez votre nombre de cheveux blancs et déclarer le comme variable (normalement égale à 0). Globalement il est dit que le stress augmente le nombre de cheveux blanc de 1,5/an.

Déclarer la variable `cheveux_blanc` comme étant une entrée (`input`) Effectuer une opération entre les variables qui permette en fonction du nombre d'années ou vous avez stressé de connaître votre nombre de cheveux blancs. Afficher votre nombre de cheveux blancs

2.4 Les structures conditionnelles

Les fonctions `if else` et `elif` vont vous permettre d'établir des structures conditionnelles :

Forme en If :

```
a=5
if a>0 : # a est positif
    print('a est positif')
elif a<0 : # a est négatif
    print('a est négatif')
else : # a est nul
    print('a est nul')
```

Le code ci-dessus se comprend de la façon suivante :

- **Si** `a` est strictement supérieur à 0, alors j'écris dans l'interpréteur IPython : `a est positif`
- **Si** `a` est strictement inférieur à 0 alors j'écris dans l'interpréteur IPython : `a est négatif`
- **Sinon** `a` est nul et j'écris dans l'interpréteur IPython : `a est nul`

Le else / elif n'est pas obligatoire :

```
age = 21
if age >= 18:
    print("Vous␣êtes␣majeur.")
else:
    print("Vous␣êtes␣mineur.")
```

ATTENTION : Les tabulations dans python sont très importantes, ce sont elles qui marquent l'appartenance à une structure type conditionnelle, boucle, etc...

Exemple :

```
if a>0 :
    print('a␣est␣positif')
```

L'écriture de `a est positif` dans l'interpréteur python ne se fait que si a est supérieur à 0 !

```
if a>0 :
print('a␣est␣positif')
```

L'écriture de `a est positif` dans l'interpréteur python se fait quoiqu'il arrive !

2.5 Les opérateurs de comparaison

Les conditions doivent nécessairement introduire de nouveaux opérateurs, dits opérateurs de comparaison.

- == : Égal à
- <= : Inférieur ou égal à
- >= : Supérieur ou égal à
- != : Différent de
- > : Strictement supérieur à
- < : Strictement inférieur à

2.6 Exercice récapitulatif :

Déclarer la variable `Moyenne` comme une variable d'entrée (`input`).

Puis réaliser un programme qui :

- si `Moyenne` est supérieur à 10, écrit dans la console IPython `Je passe au semestre suivant`
- si `Moyenne` est inférieur à 8, calcul la différence entre votre moyenne et 10 puis écris `je suis en dessous de la moyenne de` , la différence calculée, `donc je redouble.`
- si vous avez entre 8 et 10, écrit `je vais au rattrapage` et calcule le nombre de points à rattraper pour avoir 10, puis écrit ce nombre.

3 Les boucles :

3.1 La boucle `while`

Elle permet de répéter un bloc d'instructions tant qu'une condition est vraie (while signifie « tant que » en anglais)

```
nb = 7
i = 0
while i < 10:
    print(i + 1, "*", nb, "=", (i + 1) * nb)
    i =i+1
```

1. On déclare `nb` comme étant égale à 7 et `i` comme étant égale à 0.
2. La boucle `while` : elle s'exécute tant que `i` est inférieur à 10.
3. Donc si `i` est inférieur à 10 on écrit dans la console IPython : $\texttt{i} \times \texttt{np} = (\texttt{i} + 1) \times \texttt{nb}$
4. A chaque tour de boucle, `i` s'incrémente de 1.

On est donc en train d'écrire la table du 7.

3.1.1 `break`

Le mot clef `break` permet de stopper la boucle `while` avant que la condition qui la caractérise soit vraie.

```
while True:
    lettre = input("Tapez 'Q' pour quitter : ")
    if lettre == "Q":
        print("Fin de la boucle")
        break
```

1. Comme le test de `while` est toujours vrai (`True`), la boucle `while` ne s'arrête jamais
2. Le programme demande à un utilisateur de taper la lettre Q avec la fonction `input`
3. Si l'utilisateur tape la lettre `Q` le programme écrit dans la console IPython `Fin de la boucle` et stoppe la boucle `while` avec le mot `break`

3.1.2 Continue

La fonction `continue` permet de repartir au début de la boucle sans exécuter la totalité des instructions en son sein.

```
i = 1
while i < 20: # Tant que i est inférieure à 20
    if i % 3 == 0:
        i = i+4 # On ajoute 4 à i
        print("On incrémente i de 4. i est maintenant égale à", i)
        continue # On retourne au while sans exécuter les autres lignes
    print("La variable i =", i)
    i = i+1 # Dans le cas classique on ajoute juste 1 à i
```

- On déclare `i` comme étant égal à 1
- Commencement de la boucle `while`, la boucle tourne tant que `i` est inférieur à 20.
- Si le reste de : i divisé par 3, est nul alors on incrémente i de 4 et on écris dans la console : `On incrémente i de 4. i est maintenant égale à i+4`
- Puis on repart au d&but de la boucle sans exécuter le reste du code.
- Si ce n'est pas le cas alors on écrit `'La variable i =',i` dans la console IPython
- Puis on incrémente i de 1.

3.2 La boucle `for`

L'instruction `for` travaille sur des séquences. Elle est en fait spécialisée dans le parcours d'une séquence de plusieurs données. Nous n'avons pas vu (et nous ne verrons pas tout de suite) ces séquences assez particulières mais très répandues, même si elles peuvent se révéler complexes.

```
chaine = "Bonjour les L2"
for lettre in chaine :
    print(lettre ,end='|')
```

Résultat :

```
B|o|n|j|o|u|r| |l|e|s| |L|2|
```

Ce n'est pas à vous d'incrémenter l'indice de la boucle `for`. De plus cette boucle s'utilise principalement en association avec le terme `range`.

```
for i in range(5):
    print(i)
```

La boucle va opérer avec un `i` qui commence par 0 et qui s'incrémente à chaque tour jusqu'à 4 ; soit un total de 5 tours c.a.d. [0,1,2,3,4]

De même que pour la boucle `while` vous pouvez utiliser les fonctions `break` et `continue` dans une boucle `for`.

3.3 Exercice récapitulatif

On vous donne l'ébauche de programme suivante :

```
x=100*[0] # création d'une liste x formée de 100 zéros
y=100*[0] # création d'une liste y formée de 100 zéros
for i in XXX:
    XXX
    XXX
    XXX # à compléter
    XXX

    x[i]= x # à mettre à l'intérieur de la boucle for
    y[i]= y # à mettre à l'intérieur de la boucle for
```

Compléter ce programme pour avoir `x` qui s'incrémente de 0 à 4π sur 100 valeurs et `y` qui est égal au sinus de x (c.a.d. $y = \sin(x)$.)

- Si `x` est égal à 2π écrire `La première période est passé`
- Sinon continuer la boucle

Aide : Pour accéder à π il faut l'importer depuis le module `math`

```
from math import pi
```

4 Les chaînes de caractères

4.1 Concaténer des chaînes de caractères

On cherche à regrouper deux chaînes en une, en mettant la seconde à la suite de la première. Ceci ce fait avec le terme `+`. Cependant il faut faire attention à ce que le type de variable soit bien le même.

Exemple :

```
a='Coucou'
b='les'
c='licences'
d=2
Phrase=a+" "+b+" "+c # Cette concaténation fonctionne
# Cette concaténation ne fonctionne pas car 2 est du type int
Phrase_complete= a+" "+b+" "+c+" "+d
```

Cependant il vous est possible de convertir vos différentes variables en format texte.

```
# il faut utiliser la conversion en chaîne
Phrase_complete= a+" "+b+" "+c+" "+str(d)
```

La fonction `str` va convertir vos variable en `string` (chaîne en anglais)

Une chaîne se comporte comme un assemblage de lettre, il est alors possible d'accéder à chaque lettre de la manière suivante :

```
a='Bonjour'
b=a[2] # on obtient b='n'
b=a[4] # on obtient b='o'
b=a[0] # on obtient b='B' qui est différent de b
# Les nombre négatif donne accès à la liste dans l'autre sens
b=a[-1] # on obtient b='r'
```

ATTENTION : Les chaînes de caractères sont immuables c.a.d. qu'on ne peut pas les modifier. Ainsi écrire `a[0]='b'` n'est pas possible. Il faut recréer une nouvelle chaîne pour obtenir ce que l'on désire.

4.2 Les fonctions et les méthodes utiles :

- `len` : Affiche la taille de la chaîne
- `min` : diminutif de minimum, retourne le plus petit élément de la séquence (dans notre cas, c'est l'ordre alphabétique qui est considéré)
- `max` : diminutif de maximum, retourne le plus grand élément de la séquence (dans notre cas, c'est l'ordre alphabétique qui est considéré)

Avec `chaine = 'peuh !'`

- `chaine.capitalize()` : retourne la même chaîne mais avec une majuscule pour la première lettre.
- `chaine.center(18)` : retourne la chaîne entourée d'espaces et centrée dans un ensemble dont la taille est donnée en paramètre. Ici, ma chaîne fait 6 caractères et je demande de centrer sur 18, la fonction rajoute alors 6 espaces de chaque côté de ma chaîne.

Résultat : `'      peuh !      '`

- `chaine.count("peu")` : retourne le nombre d'occurrences de la chaîne passée en argument présentes dans la chaîne à laquelle la fonction est appliquée.

Résultat : `1`

- `chaine.find('uh')` : retourne l'indice de la première occurrence de la chaîne cherchée passée en argument.

Résultat : `2`

- `chaine.replace("p","f")` : retourne la chaîne de caractères avec toutes les occurrences de la première chaîne passée en argument remplacée par la deuxième chaîne passée en argument.

Résultat : `'feuh !'`

- `chaine.split()` : retourne une liste de chaînes de caractères correspondant aux différentes parties de la chaîne initiale coupée par la chaîne passée en argument. Par défaut, c'est le caractères ' ' (espace) qui est utilisé.

Résultat : `['peuh', '!']`

- `chaine.upper()` : retourne la chaîne tout en minuscule/majuscule.

Résultat : `'PEUH !'`

5 Exercices

5.1 Exo

Affichez chaque caractère d'une chaîne en utilisant une boucle `for`. Affichez chaque élément d'une liste en utilisant une boucle `for`.

5.2 Exo

Utilisez l'instruction `break` pour interrompre une boucle `for` d'affichage des entiers de 1 à 10 compris, lorsque la variable de boucle vaut 5.

5.3 Exo

Soit la chaine `'Bonjour les licences 2'`, réaliser un programme qui inverse cette chaîne de caractère : `2 secnecil sel ruojnoB`

Réaliser le même programme mais avec la chaîne de caractères qui est une entrée utilisateur (`input`)

5.4 Exo

Soit la chaîne `Les licences deux programment`, réaliser un programme qui détermine le nombre de `e` dans la chaîne

Réaliser le même programme mais avec la chaîne de caractères qui est une entrée utilisateur (`input`)

5.5 Exo

A l'aide des chaînes de caractères et de boucles `for` afficher les formes suivantes dans la console IPython.

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

*
**
***
****
*****
******
*******

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