M1 SIEE
INITIATION ALGORITHMIQUE ET PROGRAMMATION
LANGAGE PYTHON
TRAVAUX PRATIQUES
Brice Mayag

TP0 : INTRODUCTION AU LANGAGE PYTHON

Ce document est largement inspiré du cours de Python de Madame Maude Manouvrier `https://www.lamsade.dauphine.fr/~manouvri/`.

Installation et Prise en main

Cette section vous explique comment exécuter des commandes ou un programme Python depuis un éditeur Python.

Pour installer Python sur votre machine personnelle, vous devez télécharger la dernière version du langage à l'adresse `https://www.Python.org/downloads/`. L'installation de Python génère l'installation d'une interface, appelée IDLE (Python GUI). Cette interface vous permet de saisir des instructions en ligne de commande mais également d'exécuter des programmes Python enregistrés dans des fichiers. L'accès à l'interface IDLE se fait à partir du répertoire où Python a été installé. Il suffit alors de cliquer sur IDLE (voir Figure 1) qui va vous ouvrir l'interface graphique où vous pourrez taper vos instructions Python en ligne de commandes :

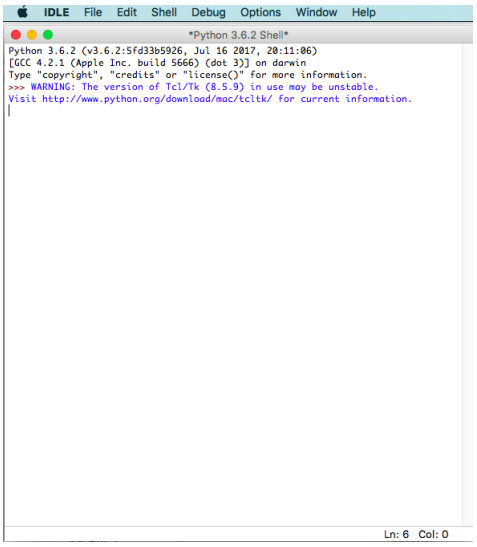

FIGURE 1 – Interface IDLE

Pour écrire un programme dans un fichier, dans le menu *File*, sélectionnez *New File*. Une nouvelle fenêtre s'ouvre. Tapez votre programme Python dans cette fenêtre (attention aux indentations). Pour exécuter votre programme, allez dans le menu Run et faites Run Modules (ou F5). Il va vous être demandé de faire une sauvegarde de votre fichier (qui a généralement l'extension *.py*), puis votre programme s'exécutera (dans la fenêtre en ligne de commande précédemment ouverte). Le début de l'execution de votre programme est indiqué dans la fenêtre en ligne de commande par le mot clé `RESTART`.

D'autres outils intégrant un éditeur de texte peuvent aussi être utilisés pour programmer en Python. Exemples : Canopy `https://www.enthought.com/products/canopy/` et IEP (Interactive Editor for Python) `http://www.pyzo.org/iep.html`.

Pour installer un package dans Canopy, il suffit d'ouvrir dans Outils le terminal Canopy (voir Figure) et de taper la commande `pip install <package name>`.

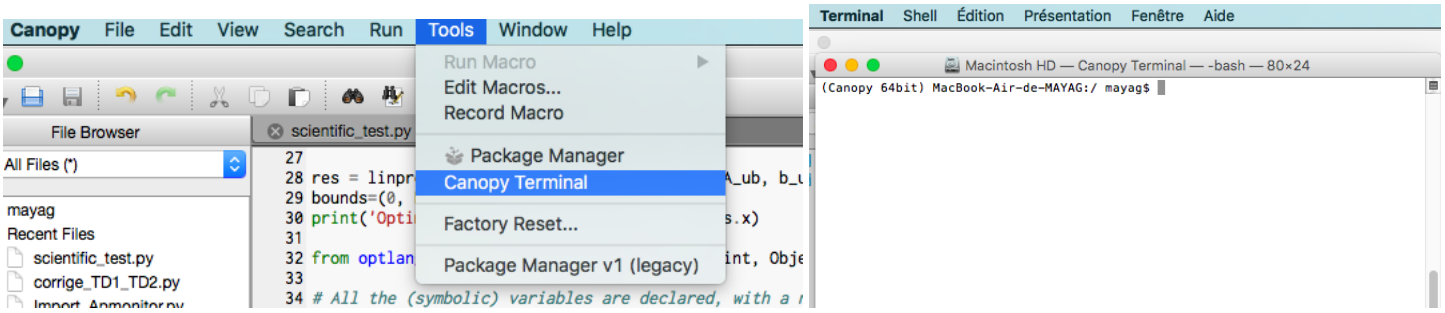

FIGURE 2 – Terminal Canopy et installation package

Premiers pas en Python

Cette section présente quelques exemples de code Python, réalisés avec `Python 3.6` en ligne de commande. Les lignes commençant par >>> correspondent aux instructions. Les lignes situées juste en dessous correspondent à l'affichage après exécution de l'instruction (i.e. après avoir tapé `<Enter>`). En Python, les commentaires commencent par le symbole #.

Vous êtes invités à taper les exemples ci-dessous pour vous entaîner et à répondre à chaque question associée aux exemples.

Exercice 1 : Quelques exemples de calcul

Essayez, en les exécutant, de comprendre ce que fait chaque instruction (non commentée) de l'exemple ci-dessous. Cet exemple est valable en ligne de commande uniquement.

```
>>> 5+3
8
>>> 5*3
15
>>> 5**3
125
>>> x=1 # declaration d'un variable x de valeur 1 (# pour le commentaire)
>>> x  # affichage de x
1
>>> a,b,c=3,5,7 # declaration de 3 variables a, b et c de valeurs resp. 3, 5 et 7
>>> a-b/c
2.2857142857142856
>>> (a-b)/c
-0.2857142857142857
>>> b/c
0.7142857142857143
>>> b//c
```

```
0
>>> b%c
5
>>> d=1.1
>>> d/c
0.15714285714285717
>>> d//c
0.0
>>> from math import * # Pour importer la librairie de fonctions mathematiques
>>> sqrt(4) # Pour calculer la racine carree
2.0
>>>pi
3.141592653589793
```

NB : La liste des fonctions de la librairie `math` est disponible à l'adresse : `http://docs.python.org/library/math.html?highlight=math#math`.

Exercice 2 : Utilisation de la fonction d'affichage print()

```
>>> print(a+b) # a et b sont les variable de l'exercice 1
8
>>> print('la valeur de', a,'+',b,'est :', a+b)
('la valeur de', 3, '+', 5, 'est :', 8)
```

Exercice 3 : Déclaration et initialisation de variables et types

```
>>> print(type(a)) # a est la variable de l'exercice 1
<class 'int'>
>>> pi=3,14
>>> print(type(pi))
<class 'tuple'>
>>> pi=3.14
>>> print(type(pi))
<class 'float'>
>>> s='exemple de chaine de caracteres'
>>> type(s)
<class 'str'>
7
>>> 2+'1.5'
Traceback (most recent call last):
File "<console>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> 2+eval('1.5') # Pour \'eliminer l'erreur pr\'ec\'edente
3.5
```

Exercice 4 : Manipulation des chaînes de caractères et exemples de fonctions sur les chaînes de caractères

```
>>> s='un exemple de chaine'
>>> s2="un autre exemple"
>>> s[1] # Acces au caractere d'indice 1 (les indices commencent a zero)
```

```
'n'
>>> print(s[0],s2[0])
u u
>>> print(s[4],s2[0])
x u
>>> print(s + ' et ' + s2) # Concatenation de chaines
un exemple de chaine et un autre exemple
>>> s3=s + ' et ' + s2
>>> s3
'un exemple de chaine et un autre exemple'
>>> s2*2
'un autre exempleun autre exemple'
>>> print('La taille de s est :', len(s))
La taille de s est : 20
>>> s3[0:3] # Recuperation des caracteres de position entre les 0 et 3e
'un '
>>> s3[4:8]
'xemp'
>>> print(s3[:3]) # Recuperation des 3 premiers caracteres
un
>>> print(s3[3:]) # Recuperation des caracteres a partir de la position 3
exemple de chaine et un autre exemple
>>> s3[::-1]
'elpmexe ertua nu te eniahc ed elpmexe nu'
>>> s3.find("exemple")
3
>>> s3.replace("chaine","str")
'un exemple de str et un autre exemple'
>>> help(str) # pour afficher l'aide
>>> sentence = 'It is raining cats and dogs'
>>> words = sentence.split()
>>> print(words)
['It', 'is', 'raining', 'cats', 'and', 'dogs']
```

Exercice 5 : Boucles et conditions

A partir de cet exercice, vous pouvez saisir vos instructions dans un fichier avec l'extension `.py` et exécuter ensuite ce fichier.

Attention : En Python il n'y a pas, comme dans certains langages, d'accolade ouvrante ou fermante pour délimiter un bloc d'instructions. Les blocs d'instructions en python sont délimités par " :" puis des tabulations : toutes les instructions consécutives à un " :" et débutant par un même nombre de tabulations appartiennent à un même bloc d'instructions.

Tapez les codes suivants et observez le résultat.

1. Boucle for

```
for i in range(10): # Ne pas oublier les deux points!!
     x = 2 # Attention ne pas oublier une tab. en debut de ligne sinon erreur!!!
     print(x*i) # Ne pas oublier la tabulation en debut de ligne!!
# Tapez encore une fois <Enter> si vous etes en ligne de commande
```

2. Boucle while

```
a=0
while(a<12): # Ne pas oublier les deux points!!
       a=a+1 # Ne pas oublier la tabulation en debut de ligne!!
       print(a, a**2,a**3) # Ne pas oublier la tabulation en debut de ligne!!
# Tapez encore une fois <Enter> si vous etes en ligne de commande
```

3. Condition If/Then/Else

```
a=0
if a==0: # Ne pas oublier les deux points!!
     print('0') # Ne pas oublier la tabulation en debut de ligne!!
elif a==1: # Ne pas mettre de tabulation et ne pas oublier les deux points!!
     print('1') # Ne pas oublier la tabulation en debut de ligne!!
else: # Ne pas mettre de tabulation et ne pas oublier les deux points!!
     print('2') # Ne pas oublier la tabulation en debut de ligne!!
# Tapez encore une fois <Enter> si vous etes en ligne de commande
```

Exercice 6 : Récupérer des saisies claviers

```
>>> s=input() # taper <Enter> puis saisir quelque chose au clavier
>>> s
>>> s=input("Saisir une chaine :") # taper <Enter>
Saisir une chaine :
>>> s
```

Récupérez, analysez et exécutez le fichier SaisieEntier.py téléchargeable sur `http://www.lamsade.dauphine.fr/~mayag/teaching.html`

Exercice 7 : Listes

Tapez chacune des insctructions suivantes (en ligne de commande) et observez le résultat.

```
>>> list=['lundi', 2, 'janvier']
>>> print(list)
>>> list[0] # Mettre print(list[0]) si vous n?etes pas en ligne de commandes
>>> list[-1]
>>> print(list[2])
>>> len(list)
>>> list.append(2010)
>>> list
>>> list[3]=list[3]+1
>>> list[3]
>>> del list[0]
>>> list
>>> list.insert(0,'mardi')
>>> list
>>> 'mardi' in list
>>> 'lundi' in list
>>> list.index("mardi")
```

```
>>> list2=list[1:3]
>>> list2
>>> list3=list[:2]
>>> list3
>>> list4=list[1:]
>>> list4
>>> list5=list[-3:-1]
>>> list5
>>> list6 = list[:-1]
>>> list6
>>> list3=list3 + [2011]
>>> list3
>>> list7=3*list
>>> list7
>>> list.extend([3,4])
>>> list
>>> list=list.pop(0)
>>> list
>>> list=[1,2,3]
>>> list2=list # Attention list et list2 correspondent a la meme
>>> list2=list.copy()#list2estunecopiedelist
>>> list.pop(1)
>>> list
>>> list2
>>> list=[1,"ab",[1,True]] # liste imbriquee
>>> list[2]
>>> print(list(range(10)))
>>> print(list(range(1,10)))
>>> print(list(range(1,10,3)))
>>> help(list) # pour l'aide sur les listes
```