TP d'initiation :

Mathématiques et Python

Le langage Python seul ne sait pas faire grande chose dans le domaine mathématique, comme tracer une fonction, calculer des valeurs de fonctions usuelles, réaliser des opérations matricielles, ...Cependant, de nombreux modules ont été développés pour pallier ce manque, parmi lesquels in convient de citer :
– scipy
– numpy
– matplotlib

L'objectif de ce document est de donner quelques points d'entrée sur ces librairies et de proposer des illustrations par l'exemple de leur utilisation.

Ce que l'on peut faire sans les modules...

Types

Les types de base qui peuvent être utiles dans la suite sont les suivants :

Types numériques

– integer (attention à la division entre entiers !)
– float
– complex : l'imaginaire pur *i* est noté *j* en python.

A tout instant, il est possible d'accéder au type d'une variable a en tapant type(a) Toute variable définie avec un type change de type lors d'une nouvelle affectation. On peut aussi changer de type à l'aide des fonctions int(), float(). L'une des caractéristiques importantes de Python est le typage dynamique. Cependant, si certaines opérations provoquent un changement de type, certaines restent interdites.

Conteneurs

– listes (par exemple a= [1,2,3,4,5])
– index (les indices de listes commencent à 0 : par exemple a[2] donne 3)
– slices (a[1 : 3] donne [2,3])
Le typage dans les listes est faible, on peut combiner différents types numériques (ou non comme des chaînes de caractères, des booléens...)

De nombreuses fonctions sont associées à ces listes (concaténation, recherche de sous-chaînes...).

Opérateurs élémentaires

Les opérateurs classiques suivantes sont disponibles :

1. +,-,*,/
2. modulo : %
3. exposant : **
4. division entière :// (par exemple 9//2=4)
5. opérateurs de comparaison : ==, !=, <>, <,<=,>,>=
6. opérateurs d'affectation : =,+=,-=,*=,/=,%=,**=,//=
7. les opérateurs bit à bit :&(et),|(ou),^(XOR),~(complément à 1),<<(décalage à gauche),>>(décalage à droite)
8. opérateurs logiques : and, or, not
9. opérateurs d'appartenance (sur des types comme des chaînes) : in, not in
10. opérateurs d'identité : is, is not

La librairie standard math

Pour disposer des fonctions mathématiques usuelles, la librairie d'origine du python se nomme math. On peut alors importer juste les fonctions nécessaires par :

from math import cos

from math import log

ou toutes les fonctions mathématiques par :

from math import *

Dans le premier cas l'inconvénient est qu'il faut savoir à l'avance les fonctions utilisées par la suite, dans le deuxième cas on risque de surcharger inutilement la mémoire. A noter que pour manipuler des complexes, il faut importer le module **cmath** en plus du module math (par exemple pour réaliser des produits de complexes)

Exercices

Exercice 0 :

1- Essayer, en les exécutant, de comprendre ce que fait chaque instruction.

Quelques exemples de calcul.

```
>>> 5+3

>>>5*3

>>>5**3

>>> x=1
>>> x

>>> a,b,c=3,5,7
>>> a-b/c

>>> (a-b)/c

>>> b/c

>>> b//c

>>> b%c

>>> d=1.1
>>> d/c

>>> d//c

>>> from math import *
>>> sqrt(4)

>>>pi
```

Affichage

```
>>> print(a+b)

>>> print('la valeur de', a,'+',b,'est :', a+b)
```

Déclaration et initialisation de variables et types

```
>>> print(type(a))

>>> pi=3,14
>>> print(type(pi))

>>> pi=3.14
>>> print(type(pi))

>>> s='exemple de chaine de caractere
>>> type(s)

>>> 2+'1.5'



>>> 2+eval('1.5')
```

Boucles et conditions

```
for i in range(10):
    x = 2
    print(x*i)


a=0
while(a<12):
   a=a+1
   print(a, a**2,a**3)


a=0
if a==0:
  print('0')
elif a==1:
  print('1')
else:
  print('2')
```

2- Définir un tableau d'une ligne contenant les valeurs (1,2,3,4).

3- Définir une matrice de deux lignes.

Exercice 1 :

1- Proposer un code qui définit la fonction f1 :

$f1 : x \mapsto 3\sqrt{x} + 2\,.$

2- Donner une valeur de x et calculer f1(x).

Exercice 2 :

Écrire un programme qui demande à l'utilisateur les coordonnées de deux points dans le plan et qui calcule puis affiche la distance entre ces deux points selon la formule :

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Exercice 3 :

Écrire un exemple de code Python permettant de visualiser les représentations graphiques de fonctions de variable réelle

$$x \mapsto \sum_{k=0}^{n} \frac{1}{2^k} x^k$$

pour différentes valeurs de n.