Introduction à Python

Pour réaliser tous les TP nous allons utiliser le langage python **dans sa version 3**.

Installation

Dans un premier temps nous allons créer un espace de travail adéquat.

- Placez-vous sur votre espace de stockage de prédilection (le bureau, une clef USB, un disque dur externe, ...) et créer un répertoire `Modelisation`.
- Sélectionner un éditeur de texte [1] et taper les lignes suivantes

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  import sys
5  print(sys.version_info)
```

  que vous enregistrez dans le répertoire `Modelisation` sous le nom `Test.py`
- Rendez-vous dans votre terminal, placez vous dans le répertoire `Modelisation` et exécutez `Test.py`. Vous pourrez observez la version de python installée et utilisée sur votre machine.

Assurez-vous de bien avoir installé la version 3

Quelques rappels sur Python

Pour les TPs de modélisation nous n'utiliserons des variables que de type *entiers*, *flottants*, *chaines de caractère*, *tableaux* sous forme de *dictionnaires* et les *booléens*.

Sur les entiers et les flottants les opérations usuelles sont :

`+` pour l'addition,	`//` pour la division entière,
`-` pour la soustraction,	`%` pour le modulo (pour les entiers),
`*` pour la multiplication,	`**` pour l'exponentiation (puissance),
`/` pour la division réelle,	`int()` **ou** `float()` pour le cast.

Sur les chaines de caractères les opérations usuelles sont :

`+` pour la concaténation,	`len` pour la longueur de la chaine,
`*` pour l'itération,	
`[]` pour la récupération d'un caractère,	`str()` pour le cast.

Sur les tableaux (dictionnaires) les opérations usuelles sont :

`[]` pour la récupération d'une case,	Une chaine de caractère peut être également vu comme un tableau.
`len` pour la taille du dictionnaire,	
`dict()` pour le cast.	

1. Une préférence pour `NotePad++` gratuit et sous licence libre, qui repère les langages et colorie le texte en conséquence

Sur les booléens les opérations usuelles sont :

- `==` pour un test d'égalité,
- `<, <=, >, >=` pour un test d'inégalité,
- `and` pour le `et` de la logique,
- `or` pour le `ou` de la logique,
- `not` pour le `non` de la logique,

- `if..elif..else` pour les exécutions conditionnelles,
- `while` pour l'exécution en boucle,
- `break` interrompt une boucle,
- `continue` fait repartir au début de la boucle.

On préférera `n!=0` à `not x==0`

Le python gère automatiquement les types. Une variable `n` est entière dès que l'on tape `n=0` et peut devenir flottante à ligne suivante en saisissant `n=12.3` ou en réalisant un cast.

Affichage

Pour l'affichage on utilise la commande `print` comme dans l'exemple suivant :

```
1  n=1.23
2  m=-666
3  print("La valeur de n est", n, "et la valeur de m est ", m)
```

Saisie

La saisie se fait à l'aide d'un `input`. ATTENTION ! Le saisie en python 3 n'enregistre que des chaines de caractère. Ainsi la commande `n=input("Saisir un entier")` suivit de la saisie de `1983` fera de `n` une chaine de caractère initialisée à `"1983"`. Pour récupérer l'entier correspondant il est nécessaire de faire un cast. Les deux codes suivant donnent la même chose

```
1  n=input("Saisir un entier : ")
2  n=int(n)
```

```
1  n=int(input("Saisir un entier : "))
```

A propos des tableaux

On prendra garde à l'utilisation des tableaux. Il est **nécessaire** de les déclarer (par exemple `A=dict()`) pour les utiliser. Une fois déclaré, le tableau peu avoir une taille quelconque. On utilisera la fonction `len` pour la déterminer. La taille d'un tableau peu changer au cours du programme.

Commentaires

Enfin on prendra soin de commenter le code, facilitant ainsi le raisonnement algorithmique et la relecture. Un commentaire sur une ligne sera précédé par le symbole `#` tandis qu'un commentaire sur plusieurs lignes sera entre `"""` et `"""` (trois fois le double guillemet ").

A propos des blocs d'instructions

En `Python` , un bloc d'instruction est identifié par une tabulation (en `C++` par exemple, le bloc d'instruction est repéré par un paire d'incolade {...}). Attention ! Il s'agit du caractère de tabulation et non de 3 ou 5 espaces ! Voici un exemple

```
1  i=0
2  Ha="Ha"
3  while(i<10) :
4     i=i+1 #En Python, le i++ n'existe pas ;
5           #cependant le i+=1 fonctionne
6     print(Ha)
7     Ha=Ha*i
8  print(Ha)
```

Quelques exercices d'entrainement

Exercice 1

Pour chacun des bloc d'instruction suivant, anticiper le résultat puis le vérifier sur machine (en les tapant dans `Test.py` et en exécutant dans le terminal).

```
1.1  print(2+3)
2    print(2-3)
3    print(1-(2-5)*8)
4    print(14/7)
5    print(14//7)
6    print(13//7)
7    print(13%7)
8    print(3**2)
9    print(2**(3**2))
10   print((2**3)**2)
```

```
2.1  x="Bonjour"
2    print(x[1])
3    print(x[5])
4    print(x[7])
5    print(x+" le monde")
6    print(x+x)
7    print(3*x)
8    print("3"*x)
9    print(str(123))
10   print("0123")
11   print(0123)
12   print(int("0123"))
13   print(int(0x123))
14   print(int("0x123"))
```

```
3.1  A=dict()
2    A[0]="Oui"
3    A[1]="123"
4    A[2]=123
5    A[3]=dict()
6    A[3][0] = "Non"
7    A[3]["Oui"] = "noN"
8    print(A[0])
9    print(A[0][2])
10   print(A[2][0])
11   print(len(A))
12   print(len(A[0]))
13   print(A[3]["Non"])
14   print(len(A[3][A[0]]))
```

Exercice 2

Dans chacun des cas suivant, combien de fois la lettre `A` s'affiche-t-elle ? Vérifier vos réponses en essayant le code.

```
1.1  i=7
2    while i==7 :
3       print("A")
4       i=i+1
```

```
2.1  i=1
2    while i<7 :
3       print("A")
4       i=i+3
```

```
3.1  i=0
2    while i<7 :
3       print("A")
4       i=i+3
```

```
4.1  i=3
2    while not i==0 :
3       print("A"*i)
4       if(i<2):
5          break
6       i=i-1
```

```
5.1  i=1
2    while True :
3       print("A"*i)
4       if i%2==0 :
5          break
6       else :
7          i=i+1
8          continue
9          i=0
```

```
6.1  i=2
2    print("A"*i)
3    i=i+3
4    while i>0 and 2*i**2+1<10 :
5       print("A"*i)
6       i=-i
```

```
7.1  i=2
2    print("A"*i)
3    i=i+3
4    while i>0 or 2*i**2+1<10 :
5       print("A"*i)
6       i=-i
```

```
8.1  i=1
2    while not i!=int(i%2==1):
3       if(i%2==1) :
4          i=i+2
5       else :
6          continue
7       print("A")
```

```
9.1  i=2
2    print("A"*i)
3    i=i+3
4    while not(i>0 and 2*i**2+1<10) :
5       print("A"*i)
6       i=-i
```

Bibliothèque mathématiques

On rajoutera en en-tête du document `.py` la ligne suivante

```
1  from math import *
```

qui chargera la bibliothèque des fonctions mathématiques usuelles.

`fabs(x)` retourne $\lvert x\rvert$	`log10(x)` retourne $\log(x)$	`e` retourne $e = e^1$
`factorial(n)` retourne $n!$	`pow(x,y)` retourne x^y	`sin(x)` retourne $\sin(x)$
`exp(x)` retourne e^x	`sqrt(x)` retourne $\sqrt{x}$	`cos(x)` retourne $\cos(x)$
`log(x)` retourne $\ln(x)$	`pi` retourne π	`tan(x)` retourne $\tan(x)$

Les fonctions

Pour définir une fonction en `Python` , il suffit d'utiliser le mot clef `def`. Comme le langage n'est pas typé, il n'est pas nécessaire de préciser les types d'entrées ou de sorties. Pour la valeur de retour, on utilise le mot clef `return`. On peut retourner simultanément plusieurs valeurs en `Python` .

```
1  def binom(n, p) :
2     if(n<=0 or p>=n or p<=0) :
3        return 1
4     if(p==1) :
5        return n
6     return binom(n-1, p)+binom(n-1, p-1)
7
8  print("Le coef binomial C_30^15
9         vaut", binom(30, 15))
```

```
1  def bla(n, a, b) :
2     x=a**n+b**n
3     y=a+b
4     return x, y
5
6  a=int(input("Saisir a : "))
7  b=int(input("Saisir b : "))
8  n=int(input("Saisir n : "))
9
10 u, v = bla(a,b,n)
11
12 print(u, v)
```