Université de Strasbourg
Première année de licence
Semestre 1

Algorithmique et Programmation 1

TP1 : Découverte de *Python*

1 Prise en main

1.1 Installation

L'interpréteur *python* est déjà installé sur les machines de l'UFR de math-info. Vous pouvez, si vous le souhaitez, l'installer sur votre machine personnelle. Si votre machine fonctionne avec un système *Unix* (exemple : *Linux* ou *Mac OS*) il est très probable que l'interpréteur *python* soit déjà installé. Mais il peut s'agir d'une version antérieur à la version 3. Dans ce cours, nous travaillerons avec la version 3.0 ou une version plus récente.

Pour vérifier si la version 3 de python est installée sur votre machine, ouvrir un terminal (une invite de commandes sous *windows*) et entrer *python3*. Vous devriez voir apparaître quelque chose ressemblant à ceci :

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 19 2015, 20:38:52)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more info
>>>
```

Si ce n'est pas le cas, le site `www.python.org` vous permettra d'installer *python3*.

Attention ! ! ! Si vous installez *python3* je vous recommende de ne pas désinstaller les versions précédentes. Un grand nombre de programmes et de packages peuvent dépendre de ces versions antérieures.

1.2 Votre première instruction *python*

Les trois caractères >>> constituent une invite de commandes (*prompt* en anglais). Ils indiquent que l'interpréteur *Python* est prêt à exécuter les instructions que vous lui soumettez. Écrivez `print("Bonjour!!!")` et valider. Vous devriez avoir le résultat suivant :

```
>>> print("Bonjour!!!")
Bonjour !!!
```

Pour quitter *python*, vous pouvez soit écrire `exit()` soit presser `Ctrl-D`.

2 À la découverte de la syntaxe de *Python*

Il y a deux façons de découvrir l'usage d'un langage :

1. Lire un cours, un livre, regarder un tutorial qui vous explique tout ;
2. Expérimenter vous-même.

Nous aurons recours aux deux méthodes. Dans ce premier TP, nous commençerons pas la seconde méthode. Sur la plateforme *Moodle*, ouvrir le *TP1 découverte expérimentale QCM*. Pour répondre à ces questions, vous devrez faire des expériences. On ne peut pas répondre à toutes les questions de façon expérimentale, mais pour celles qui vous sont proposées ici, c'est toujours possible.

À titre d'exemple, on vous demande de déterminer si l'expression `a+b*c` est interprétée par python comme `(a+b)*c` ou comme `a+(b*c)`. Pour répondre à cette question de façon expérimentale, il est nécessaire de choisir des valeurs pour `a`, `b` et `c`. Par exemple respectivement 3, 5 et 6. Mais il faut surtout les choisir pour que `(a+b)*c` et `a+(b*c)` n'aient pas la même valeur, sinon l'expérience ne nous apportera aucune information. En l'occurrence :

— `(3+5) * 6` vaut 48

— `3 + (5*6)` vaut 33

Or quelle valeur affiche python lorsque vous exécutez l'instruction :

```
>>> 3 + 5 * 6
```

Est-ce que le résultat est 48 ? ou 33 ? Ainsi, est-ce que python a d'abord évalué la multiplication ou l'addition ?

3 Le mode script

Ouvrir un éditeur de texte simple comme *gedit, emacs* ou *vim* (sous linux) ou *TextEdit* (sous MacOS). N'utilisez pas *word* ou *libreoffice*. Dans ce fichier, écrire :

```
print("Bonjour!!!")
```

Sauvegarder ce fichier sous le nom que vous souhaitez, par exemple `salut.py`. Ceci est votre premier programme *python*. Pour l'exécuter, ouvrir un terminal et assurez-vous que votre répertoire courant est celui qui contient le fichier `bonjour.py`. Entrer la commande :

```
python3 salut.py
```

Et vous devriez voir apparaître le message :

```
Bonjour !!!
```

dans votre terminal.

Dans les sections précédentes, vous avez utilisé le mode *console*, alors que dans la présente section, vous avez utilisé le mode *script*.

— Le mode console est très adapté pour des tests ponctuels. Il est rassurant aussi pour les débutants car à chaque instruction, il affiche le résultat des expressions évaluées. Par contre, il n'est pas du tout adapté à un programme conséquent, d'une part parce qu'il ne permet pas de conserver la trace de toutes les commandes entrées pour pouvoir les réutiliser dans une session ultérieure. D'autre part, il ne facilite pas de remonter à une instruction particulière, de la corriger et d'exécuter toutes les suivantes ;

— Dans toute la suite, nous utiliserons le mode script.

4 Utilisation d'un *IDE*

FIGURE 1 – Chercher *Geany* sur Ubuntu

Un *IDE (Integrated Development Environment)* est un logiciel qui facilite la programmation en mode *script*. Il n'est pas du tout indispensable. Nous avons vu à la section précédente, qu'un simple éditeur de texte est suffisant pour programmer en *python*. Mais l'*IDE* est un élément de confort.

Vous pouvez utiliser l'*IDE* de votre choix. Celui que nous décrirons ici est *Geany*. Il est installé sur les machines de l'UFR. Suivre les étapes suivantes :

1. Presser le bouton de recherche d'ubuntu (en haut à gauche de l'écran) et entrer *Geany*. Cliquer sur l'icône de *Geany* pour lancer l'application (cf figure 1) ;
2. Dans le menu *Fichier*, choisir *Ouvrir* et, dans la boîte de dialoguq qui s'ouvre, choisir le fichier `salut.py` que vous avez créé à la section précédente.
3. Dans le menu *Construire*, choisir *Définir les commandes de construction* ;
4. Dans la boîte de dialogue qui apparaît, assurez-vous que les commandes de construction font appel à `python3` et non à `python` (cf figure 2).

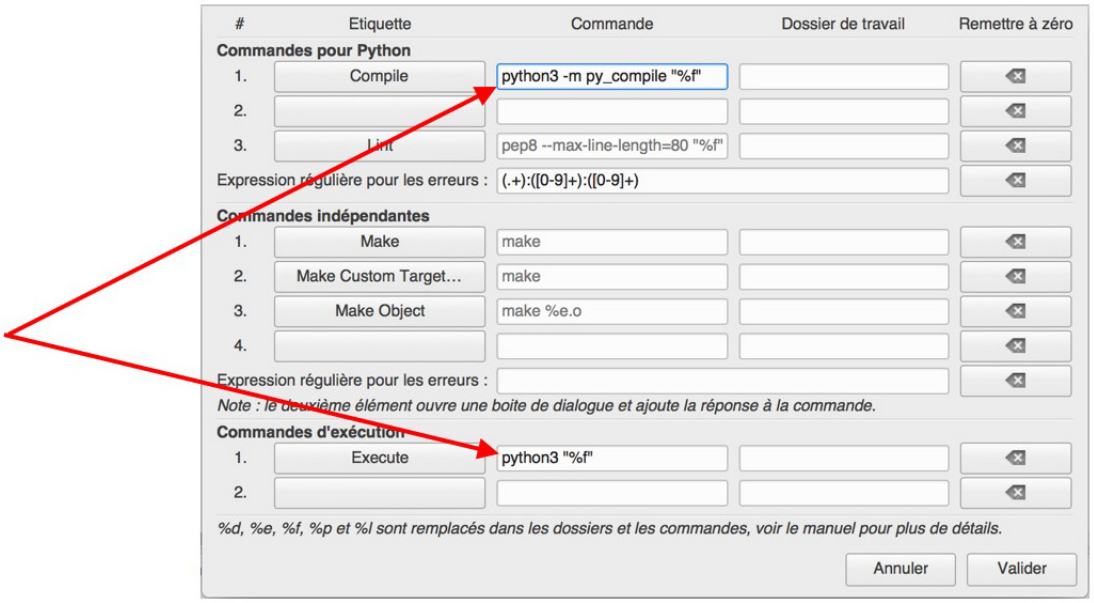

FIGURE 2 – Définir les commandes de construction (en particulier s'assurer que la version de python est bien la version 3 ou une version plus récente).

À partir de ce point, pour exécuter un script sous *Geany*, il suffira de presser la touche `F5` (ou `fn+F5` sous MacOS). Vous êtes libre d'utiliser l'éditeur de votre choix ou tout IDE de votre choix.

5 Entrée utilisateur

Écrire le programme suivant et l'exécuter.

```
saisi = input("Entrer un nombre : ")
print(saisi)
```

La fonction `input` permettra au programme d'interroger l'utilisateur. Essayez le programme suivant :

```
saisi1 = input("Entrer un nombre : ")
saisi2 = input("Entrer un autre nombre : ")
print("La somme de ces deux nombres est de : ")
print(saisi1 + saisi2)
```

D'après le résultat de cette exécution, quel est le type de `saisi1` et de `saisi2` ? (entier ? flottant ? ou chaîne de caractères ?)

1. Écrire un programme qui demande à l'utilisateur d'entrer deux nombres (comme ci-dessus) et qui affiche la somme numérique de ces nombres. Autrement dit, si l'utilisateur entre 5 et 7, la valeur affichée par le programme devra être 12.
2. Écrire un programme qui demande à l'utilisateur d'entrer un montant en euros (non nécessairement entier). Le programme affichera alors le montant correspondant en dollars américains. On suppose qu'un euro vaut 1.17 dollars américains.
3. Écrire enfin un programme qui demande à l'utilisateur d'entrer un montant en euros (comme dans l'exercice précédent), mais au lieu d'afficher simplement le montant en dollars, il devra intégrer ces données dans un message texte. Par exemple si l'utilisateur a entré `5.20`, alors le programme devra afficher un message :

   ```
   5.2 euros valent 6.084 dollars.
   ```

   Je ne vous demande pas d'essayer d'afficher deux chiffres après la virgule.