Exercice 1

1.1 L'écosystème Python

1. Définir un environnement virtuel que l'on placera dans le répertoire `~/option_python`.
2. Charger ce nouvel environnement et s'assurer que la commande `python` pointe bien vers le répertoire `bin` du répertoire `~/option_python`.
3. Lister les modules actuellement installés puis suivre les recommendations issues de la précédente commande.
4. Installer la version `1.5.2` de la librarie `matplotlib` puis mettre à jour cette dernière.
5. Installer l'interpréteur `ipython` et l'utiliser pour vérifier que l'installation de `matplotlib` a été correctement réalisée en important la librairie *via* la commande

   ```
   import matplotlib
   ```

6. Afficher la chaîne de caractères `"Bonjour tout le monde"`
7. Définir une première variable `x1` égale à `10` et une seconde `x2` égale à `10.0`. Afficher le statut des variables.
8. Afficher la partie réelle puis la partie imaginaire de `x1`. Afficher la taille en bits de `x1`. Lors des différentes opérations, on pourra se servir de l'auto-complétion et de la commande d'aide `?` pour accéder à la documentation de chaque commande.
9. Afficher la documentation de la fonction `input` puis se servir de cette fonction pour demander à l'utilisateur la saisie d'un nombre `y`.

Dans toute la suite de cette option et notamment lors des prochains exercices, on prendra bien garde de systématiquement charger l'environnement virtuel défini lors de ce premier exercice.

correction

1. Dans l'invite de commande du terminal, répérée ici par le signe `$`, on tape les commandes suivantes

   ```
   $ python3.5 -m venv ~/python.d/my_python_env
   ```

2. ```
   $ source ~/python.d/my_python_env/bin/activate
   $ which python
   ~/python.d/my_python_env/bin/python
   ```

3. Liste de l'ensemble des modules installés

   ```
   $ pip freeze
   ```

   puis mise à jour de `pip`

   ```
   $ pip install --upgrade pip
   ```

4. ```
   $ pip install matplotlib=1.5.2
   $ pip install --upgrade matplotlib
   ```

5. ```
   $ pip install ipython
   $ ipython
   ```
   ```

correction

```
Python 3.6.0 (default, Jan 16 2017, 12:12:55)
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import matplotlib
```

6.
```
In [2]: print("Bonjour tout le monde")
```

7.
```
In [3]: x1 = 10

In [4]: x2 = 10.0

In [5]: %whos
```

8.
```
In [6]: x1.real
Out[6]: 10

In [7]: x1.imag
Out[7]: 0

In [8]: x1.bit_length()
Out[8]: 4
```

9. On pourra utiliser soit la commande `input?`, propre à l'interpréteur `ipython`, soit l'aide intéractive *i.e.* `help(input)`

```
In [9]: input?
Signature: input(prompt=None, /)
Docstring:
Read a string from standard input.  The trailing newline is stripped.

The prompt string, if given, is printed to standard output without a
trailing newline before reading input.

If the user hits EOF (*nix: Ctrl-D, Windows: Ctrl-Z+Return), raise EOFError.
On *nix systems, readline is used if available.
Type:      builtin_function_or_method
```

Comme l'indique l'aide, la fonction `input` lit une chaîne de caractère et on prendra donc bien garde à convertir la valeur saisie en nombre (flottant)

```
In[10]: n = float(input("Saisissez un nombre "))
```

1.2 La calculatrice Python

1. Dans l'interpréteur `ipython` réaliser les opérations arithmétiques d'addition, soustraction, multiplication et division sur des nombres entiers ainsi que sur des nombres flottants.
2. Comparer le résultat de la division de deux entiers lorsque vous utilisez l'opérateur `/` et `//`. Dans le second cas, afficher le reste de la division.
3. À l'aide de la fonction `type` dont on cherchera le fonctionnement à l'aide de l'opérateur `?` de `ipython`, afficher la nature de nombres entier et flottant.
4. Déclarer deux nombres `i = 10` et `x = 10.0` et tester leur égalité *via* l'opérateur ==. Stocker ce résultat dans une variable `test` et retourner son type.
5. Calculer le nombre de valeurs pouvant être encodées sur `12` bits.
6. Importer le module mathématiques de Python à l'aide de la commande

```
import math
```

En vous servant de l'aide intéractive fournit par `ipython`, déterminer la valeur de factoriel 13. Calculer la valeur du cosinus d'un angle mesurant 666°.

correction

1.
```
In [1]: x1 = x2 = 10.0

In [2]: x1+x2, x1-x2, x1*x2, x1/x2
Out[2]: (20.0, 0.0, 100.0, 1.0)

In [3]: i1, i2 = 3, 4

In [4]: i1+i2, i1-i2, i1*i2, i1/i2
Out[4]: (7, -1, 12, 0.75)
```

2.
```
In [5]: i1/i2, i1//i2
Out[5]: (0.75, 0)
```

Le reste de la division s'obtient *via* l'opérateur modulo `%`

```
In [6]: i1%i2
Out[6]: 3
```

3.
```
In [7]: type(i1), type(x1)
Out[7]: (int, float)
```

4.
```
In [8]: i, x = 10, 10.0

In [9]: i == x
Out[9]: True

In [10]: test = i == x

In [11]: type(test)
Out[11]: bool
```

5.
```
In [12]: 2**12
Out[12]: 4096
```

6.
```
In [13]: import math

In [14]: math.factorial(13)
Out[14]: 6227020800

In [15]: math.cos(math.radians(666))
Out[15]: 0.5877852522924728
```