Initiation à Python

Guide de survie

```
>>> import string
>>> key = string.ascii_lowercase
>>> code = "10 0 0 12 4 11 14 19 19"
>>> for i in code.split():
...     print(key[int(i)].upper())
```

Contact :

pierre-antoine.jean@mines-ales.org

Vue d'ensemble

- Python est un langage de programmation *interprété* (un "interprète" analyse, traduit et exécute un programme)
- L'interprète se distingue d'un compilateur car il réalise les opérations d'analyse et de traduction à chaque exécution du programme plutôt qu'une fois pour toutes comme le fait le compilateur

Roi Burgonde

"Interprèèète ? Interprèèète, couhillère ?"

⊖ Vitesse

⊕ Flexibilité et rapidité de mise en oeuvre

- On distingue un programme dit **script**, d'un programme **compilé**
  - Un programme script est exécuté à partir d'un fichier source via un interprète
  - Un programme compilé est exécuté à partir d'un bloc en langage machine issu de la traduction du fichier source

Les bases du langage - Exécution d'un programme

```
$ which python
/usr/bin/python
$ which python3
/usr/bin/python3
$ python --version
Python 2.7.15+
$ python3 --version
Python 3.6.8
$ echo 'print("Sire on en a gros!")' > premier_pas.py
$ python3 premier_pas.py
Sire on en a gros!
$ python3.6 premier_pas.py
Sire on en a gros!
$ python3 premier_pas.py | tr [a-z] [A-Z]
SIRE ON EN A GROS!
```

Les bases du langage - Les variables

- La déclaration et l'initialisation d'une variable se font en même temps

```
>>> age_merlin = 884
>>> age_merlin
884
```

- Python est un langage dont le typage est dynamique, il reconnaît certains types de variable automatiquement *e.g.* entier et *float* (une chaîne de caractères nécessite des guillemets (simple ou double)

```
>>> type(age_merlin)
<class 'int'>
>>> age_merlin = 88.4
>>> type(age_merlin)
<class 'float'>
>>> profession_merlin = "enchanteur"
<class 'str'>
```

Les bases du langage - Les listes

Python autorise la construction de liste contenant des valeurs de types différents (*e.g.* entier et chaîne de caractères)

```
>>> sloubi = [1,2,3,4,5,6,7]
>>> sloubi[0]
1
>>> sloubi[-1]
7
>>> sloubi[1:-2]
[2, 3, 4, 5]
>>> len(sloubi)
7
>>> sloubi.append(8)
>>> sloubi
[1, 2, 3, 4, 5, 6, 7, 8]
>>> del sloubi[1]
>>> sloubi
[1, 3, 4, 5, 6, 7, 8]
```

Les bases du langage - Les boucles

```
>>> sloubi = [1,2,3]
>>> for i in sloubi:
...     print("sloubi", i)
sloubi 1
sloubi 2
sloubi 3
>>> for i in range(len(sloubi)):
...     print(i,sloubi[i])
0 1
1 2
2 3
>>> i = 0
>>> while i < len(sloubi):
...     print(sloubi[i])
...     i += 1
1
2
3
```

Les bases du langage - Les conditions

Les opérateurs de comparaison

==	égal à
!=	différent de
>	strictement supérieur à
>=	supérieur ou égal à
<	strictement inférieur à
<=	inférieur ou égal à

```
>>> sloubi = [1,2,3,4]
>>> for i in sloubi:
...     if i % 2 == 0:
...         print(i)
2
4
```

Les bases du langage - Les fichiers

```
>>> fin = open("episodes.tsv", "r")
>>> content = fin.readlines()
>>> for i in content:
...     print(i.strip())
Arthur
Merlin
>>> fin.close()
>>> fout = open("episodes.tsv", "a")
>>> fout.write("Perceval")

$ more episodes.tsv
Arthur
Merlin
Perceval

>>> fout = open("episodes.tsv", "w")
>>> fout.write("Perceval")
>>> fout.close()

$ more episodes.tsv
Perceval
```

Les bases du langage - Les fonctions

```
>>> def carre(x):
...     return x**2
>>> carre(4)
16
>>> def add_mul(x, y = 0):
...     return x+y, x*y
>>> add_mul(3,3)
(6,9)
>>> a,b = add_mul(3)
>>> a
3
>>> def add_mul(x, *args):
...     results = [x]
...     for i in args:
...         results.append(x+i)
...     return results
>>> add_mul(3,4,5,6)
[3, 7, 8, 9]
>>> def factorielle(n):
...     if n < 2:
...         return 1
...     else:
...         return n * factorielle(n-1)
>>> factorielle(4)
24
```

Les bases du langage - Les dictionnaires

```
>>> valeurs = {"perceval": [1,2], "arthur": [3,4]}
>>> valeurs["perceval"]
[1, 2]
>>> valeurs["perceval"][0]
1
>>> valeurs["perceval"].append(3)
>>> valeurs["perceval"]
[1, 2, 3]
>>> if "perceval" in valeurs:
...     print("Clé présente.")
Clé présente.
>>> valeurs["merlin"] = {"age": 884, "profession":
"enchanteur"}
>>> valeurs
{'perceval': [1, 2, 3], 'arthur': [3, 4], 'merlin':
{'age': 884, 'profession': 'enchanteur'}}
```

Les bases du langage - Les paquets

```
>>> import os
>>> os.listdir(".")
['Musique', 'Modèles', 'Images', 'Vidéos', 'Bureau',
'Documents', 'Téléchargements', 'Public']
>>> from os import path
>>> filepath, filename =
path.split("/home/perceval/episodes.tsv")
>>> filepath, filename
>>> ('/home/perceval', 'episodes.tsv')
```

- PyPI est un dépôt de paquets sous Python, il centralise les différents paquets mis à disposition par la communauté et en gère les différentes versions
- Le gestionnaire de paquets `pip` utilise par défaut ce dépôt

```
$ sudo apt update
$ sudo apt install python3-pip
$ pip3 install numpy
>>> import numpy as np
```

Les bases du langage - Les environnements virtuels

- Permet des installations de Python isolées de l'OS, et séparées les unes des autres pour chaque projet
- L'installation des paquets sera propre à l'environnement virtuel
- Utilisation du module `venv` (installation de `pip` par défaut depuis la version 3.4 de Python)

```
$ sudo apt install python3-venv
$ python3 -m venv venv_3.6
```

- Activer l'environnement virtuel et installer les dépendances à partir du fichier `requirements.txt`

```
$ source venv_3.6/bin/activate
$ pip install -r requirements.txt
```

- Désactiver l'environnement virtuel

```
$ deactivate
```

Références

https://www.commentcamarche.net/contents/1143-unix-les-fichiers
https://python.doctor/page-apprendre-listes-list-tableaux-tableaux-liste-array-python-cours-debutant
http://sametmax.com/les-environnement-virtuels-python-virtualenv-et-virtualenvwrapper/