IFT3830 - SYSADMIN.

PROGRAMMATION PYTHON

DANIEL OUIMET - STEVE LEVESQUE

- Qu’est-ce Python?
  - Un langage de programmation orienté “scripting”.
  - Un **langage de script** est un langage de programmation interprété qui permet de manipuler les fonctionnalités d'un système informatique configuré pour fournir à l'interpréteur de ce langage un environnement et une interface qui déterminent les possibilités de celui-ci. Le langage de script peut alors s'affranchir des contraintes des commandes de bas niveau.
  - https://fr.wikipedia.org/wiki/Langage_de_script
- Le sujet peut être large!
  - Pour des détails précis, il est recommandé d’aller chercher sur Internet.

INTRODUCTION

- Multi-paradigme : (Orienté objet, Fonctionnel, etc.)
- Typage dynamique : nul besoin de spécifié le type (i.e. comme Java)
- Typage fort : conversion explicite (i.e. il faut le spécifier, par str() ou int())
- Multi-usage :
  - Administration système, Scripting
  - Web, Flask (IFT3225)
  - Programmation traditionnelle
  - Apprentissage Machine (Machine Learning – ML), Data Science
  - Etc.

CARACTÉRISTIQUES

- Forces :
  - Code concis et lisible
  - Rapide à rédiger
  - Facile à apprendre
- Faiblesses :
  - Des paradigms mélangés entre-eux, Python ne force pas une structure précise
  - Même idée, plusieurs usages mélangé dans un fichier possible (Web + ML …, Sysadmin + Maths …)
  - Pas de commentaire lignes multiples (à part “documentation comments”)
- Les deux?
  - Typage fort = attention aux conversions
    - Sécuritaire
    - Ralenti le développement et la concision, à prendre en compte

FORCES ET FAIBLESSES

CODE - INTRODUCTION

- Nomenclature
  - Pas de parenthèses pour délimiter un bloc/fonction.

```
def test(nb):
    print(1+nb)

test(2)

#test(3)
```

```
[root@localhost Python]# python3 test.py
3
```

- Variables
  - L'aspect dynamique nous permet de declarer directement
  - Attention lorsque 2 types différents entrent en conflis…

```
nb1 = 1
nb2 = 2
text = "testing"
total = nb1 + nb2
final = total + text # incorrect
#final = str(total) + " " + text
print(final)
```

```
[root@localhost Python]# python3 test.py
3 testing
```

```
[root@localhost Python]# python3 test.py
Traceback (most recent call last):
  File "test.py", line 10, in <module>
    final = total + text # incorrect
TypeError: unsupported operand type(s) for +: 'int' and
'str'
```

CODE - INTRODUCTION

- Importer un module est simple et permet d'utiliser les fonctionnalités de celui-ci :

```
#!/usr/bin/python

import sys

print(sys.version)
```

```
[root@localhost Python]# python3 test.py
3.6.8 (default, datexyz)
[GCC 8.5.0 20210514 (Red Hat 8.5.0-3)]
```

CODE - MODULES ET STRUCT. FICHIER

- On peut aussi importer un script python (un module revient généralement à un ensemble de scripts)
- ATTENTION : addition.py peut ne pas avoir les permissions nécessaires pour exécuter son contenu.

```
#!/usr/bin/python

# test.py

import sys
from addition import add

print(sys.version)

add(2, 4)
```

```
#!/usr/bin/python

# addition.py

def add(nb1, nb2):
    print(nb1 + nb2)
```

```
[root@localhost Python]# python3 test.py
3.6.8 (default, Nov 17 2021, 16:10:06)
[GCC 8.5.0 20210514 (Red Hat 8.5.0-3)]
6
```

CODE - MODULES ET STRUCT. FICHIER

- Il est intéressant de pouvoir traiter des fichiers de données :
  - Txt, csv, etc.
- « with open » permet de le faire
  - Il y a plusieurs paramètres
- Il existe plusieurs autres techniques / modules pour le faire

```
#!/usr/bin/python
# readfile.py

with open('data.txt') as f:
    for line in f:
        print(line, end='')

print('.')
f.close()
```

```
# data.txt
daniel
steve

[root@localhost Python]# python3 readfile.py
daniel
steve.
```

CODE – IMPORT. DES DONNÉES

https://stackoverflow.com/questions/56980955/how-to-execute-a-perl-script-through-python-subprocess-call

- Nous pouvons démarrer des scripts.
  - Un script perl peut être utilisé pour accomplir une tâche en sous-processus.

```
#!/usr/bin/perl

sub bar
{
    return "test on ip ", $_[0], " ...\n"
}

print bar($ARGV[0]);
print("Sleeping for 5 sec\n");
sleep(5);
print("Done\n");

exit 0;
```

CODE - SOUS PROCESSUS PERL

https://stackoverflow.com/questions/56980955/how-to-execute-a-perl-script-through-python-subprocess-call

CODE - SOUS PROCESSUS PERL

```
#!/usr/bin/python
# runperl.py

import subprocess
import sys

def foo():
    IPs = ["198.168.1.2", "198.168.3.4"]
    procs = []
    for ip in IPs:
        proc = subprocess.Popen("perl check.pl
'%s'" %ip, shell=True)
        procs.append(proc)

    # wait for all the processes to finish
    for proc in procs:
        proc.wait()
        print("Proc done")

foo()
print("Script over")
```

https://stackoverflow.com/questions/56980955/how-to-execute-a-perl-script-through-python-subprocess-call

- Le script python prends en compte le temps d'attente des traitements du script perl.

```
[root@localhost Python]# python3 runperl.py
test on ip 198.168.1.2 ...
Sleeping for 5 sec
test on ip 198.168.3.4 ...
Sleeping for 5 sec
Done
Proc done
Done
Proc done
Script over
```

CODE – SOUS PROCESSUS PERL

- Un langage qui touche tous les domaines.
- Compatible avec beaucoup d’environnements.
- Un bon atout à connaître.
    - Plus d’opportunités d’emploi.

CONCLUSION

BIBLIOGRAPHIE

- Installer Python sur AlmaLinux :
  - https://www.linuxcapable.com/how-to-install-python-3-10-on-almalinux-8/
- AlmaLinux 8
  - https://almalinux.org/fr/
- VisualStudio Code
  - https://code.visualstudio.com/
- Python
  - https://www.python.org/