Le langage Python

Version 3

Systèmes d'information et de connaissances

//
Unité d'Enseignement d'Expertise
Code : PA10
//

Module M2

Document 1 :

Notions de base et découverte de l'éditeur IDLE

Année 2016-2017, Centre de PARIS
rene.boulaire@ensam.eu

ARTS ET MÉTIERS ParisTech

Préambule

- Ce document s'adresse à des personnes ayant des notions de programmation au niveau algorithmique.
- Le cours sera constitué de plusieurs documents dans le but d'aborder graduellement les différentes notions du langage.
- La compréhension de chaque notion sera dans une présentation orientée exemple.
- Un langage de programmation est assimilé par la pratique.
- La conception d'une application est basée sur la réflexion et la modélisation.

Historique

- Guido Van Rossum crée le langage Python.

  https://www.python.org/~guido/

- Le langage Python a été conçu au début des années 90.
- C'est un langage de programmation orienté objet et interprété. Il est portable et gratuit.
- Pour exécuter un programme Python, l'interpréteur Python doit être installé sur la machine.
- L'interpréteur Python existe pour les différentes plateformes actuelles (Windows, Linux, Mac)
- C'est un langage riche en modules

Le Langage

- Les domaines d'utilisation du langage Python :
  - Applications multiplateformes (avec interface graphique)
  - Calcul scientifique
  - Embarqué (robotique par exemple)
  - Langage de scripts pour certains logiciels
  - Site web

Quelques remarques sur le Langage Python

- Langage de haut niveau (gères ses propres ressources, mémoire par exemple)
  - Mémoire
  - Ramasse-miettes (comptage de références)
  - Etc …
- Langage dynamiquement typé
- Dans Python tout est objet y compris les types
- Interaction forte entre C/C++ et Python

Quelques remarques sur le Langage Python

- Existence de Python en version 2 et 3
- Les 2 versions ne sont pas totalement compatibles
- Le site du langage Python : https://www.python.org/
- Python est un langage pérenne et en évolution constante
- Quelques noms utilisant Python et proposant des outils
  - Google
  - Microsoft
  - Mozilla
  - Cisco
  - ……..

Installer Python

- Installer Python : https://www.python.org/download/
- Installer des bibliothèques externes avec le gestionnaire de paquets **pip** disponible automatiquement à partir de Python version 3.4
- Installer un IDE (pas indispensable mais conseillé)
  Python propose une console par défaut
  https://wiki.python.org/moin/IntegratedDevelopmentEnvironments
- Les principaux environnements de développements proposent le support de Python : QT, Visual Studio, Eclipse, etc …

Installer Python

- Python peut être aussi installer à partir de la suite Anaconda : http://www.continuum.io/
- Cette suite contient les différents packages pour le calcul scientifique.
- Intègre un éditeur performant : Spyder

Premiers pas avec Python

- Mode interactif depuis la ligne de commande (shell ou fenêtre DOS mode Terminal). La commande « python » lance l'interpreteur Python.

- A partir de l' éditeur standard Python (IDLE), il sera possible d'écrire des scripts et de les sauvegarder sous forme de fichiers textes ayant l'extension **.py**

- Le prompt ou signal d'invite est composé de 3 caractères **>>>**

- A partir du prompt, la machine Python est utilisable telle une machine à calculer avec les possibilités du langage Python

Quelques règles de syntaxe Python

- Particularité première du langage Python : La présentation du code. La mise en page définit les limites des instructions et des blocs (: + indentation)
- Une instruction par ligne, pas de caractère spécial de fin de ligne. Si le caractère \ est utilisé en fin de ligne, l'écriture de l'expression continue à la ligne suivante.
- Une ligne correspondant à un commentaire commence par un #
- Une fin de ligne peut être un commentaire (# …….)
- Un bloc de commentaires est encadré par la séquence : """

Python, un outil à base de modules

- Des modules disponibles à l'installation de Python.
- Ces modules correspondent à des bibliothèques d'objets.
- Le module sys permet de fournir les modules disponibles.

```
sys.builtin_module_names
```

- La commande help() peut donner de l'aide sur un module chargé en mémoire

```
help(math)
```

Interface IDLE Python

- L'interface IDLE est un éditeur de code (en mode texte) installée par défaut avec l'interpréteur de commandes.
- Ce sont 2 outils standards séparés contenus dans des fenêtres graphiques.
- Ces 2 outils permettent d'écrire des programmes enregistrés dans des fichiers (.py) et d'exécuter ces programmes .
- L'éditeur se comporte comme une machine à calculer et peut exécuter du code à la volée.

Variables dans Python

- Python utilise des variables pour mettre en mémoire une valeur.
- Minuscules et majuscules sont des caractères différents pour former le nom d'une variable.
- Il existe des mots réservés ou mots clés qui ne peuvent être utilisés comme variables.
- Le signe = assigne une valeur à la variable.

```
Mavariable = 1
Mavariable="1"
```

Variables dans Python

- La commande type() donne le type des données contenues dans la variable passée en paramètre.
- Une variable est typée dynamiquement par Python, contrairement à un typage explicite.
- Toute expression utilisant les variables déclarées dans Python effectuera une opération suivant le type de la variable.
- En Python, les variables sont des références à des valeurs. La commande id() retourne l'adresse de la valeur en mémoire.

Variables dans Python

Il est possible d'assigner une valeur à plusieurs variables en 1 seule opération

Exemple :

```
>>> x = y = 7
>>> x
7
>>> y
7
```

Effectuer des affectations parallèles à l'aide d'un seul opérateur

```
>>> a, b = 4, 8.33
>>> a
4
>>> b
8.33
```

Les variables **a** et **b** prennent les valeurs 4 et 8,33.

Variables dans Python

- Python propose un type None pour signifier qu'une variable ne contient rien.

```
>>> a=None
>>> print(a)
None
```

- Le contenu d'une variable peur être converti d'un type dans un autre type (opération appelée un cast). Il suffit de passer en paramètre la valeur ou la variable à convertir au type désiré.

```
>>> a=3.5
>>> a
3.5
>>> b=int(a)
>>> b
3
>>>
```

Les chaines de caractères dans Python

- Python permet une manipulation simple des chaines de caractères à partir de fonction s'appliquant sur les chaines de caractères. Une chaine de caractère est analogue à un tableau non modifiable dont le premier caractère est à l'indice 0.

```
>>> s='un exemple de chaine'
>>> s1="un autre exemple"
>>> s[1]
'n'
>>> s1[0]
'u'
>>> s2=s + ' et ' +s1
>>> s2
'un exemple de chaine et un autre exemple'
>>> len(s2)
40
>>> s2[0:3]
'un '
```

Les structures de données dans Python

- Les Tuples correspondent à un tableau d'objets pouvant être de tout type.
  - Ils sont non modifiables.
  - Opérations possibles sur les Tuples sous forme de fonctions.

```
>>> x=(4,5,6,'sept')      # Tuple composé de 4 objets
>>> x
(4, 5,6 ,'sept')
>>> len(x)
4
```

Les structures de données dans Python

- Le type « list » est une structure de données de tout type dont le premier élément possède l'indice 0.
  - Les éléments peuvent être de plusieurs types.
  - Cette structure peut être manipulée telle un tableau à 1 ou plusieurs dimensions.
  - Le type tableau de certains langages est intégré dans la notion de « list » Python.
  - Elle peut être parcourue dans les sens en utilisant l'indice.
  - Elle est modifiable

```
>>> x=[4,5,'sept']
>>> x
[4, 5, 'sept']
>>> x[1]=89
>>> x
[4, 89, 'sept']
>>>
```

Les structures de données dans Python

- Le type « dict » (dictionnaire) stocke les données sous la forme clé -> valeur.
  - Les valeurs peuvent être de tout type.
  - Une clé est unique et n'est pas nécessairement un entier comme dans le cas de l'indice d'une liste. La clé est immuable.

```
>>> x={'cle1':'valeur1','cle2':'valeur2','cle3':'valeur3',10:'valeur10'}
>>> x
{'cle3': 'valeur3', 10: 'valeur10', 'cle1': 'valeur1', 'cle2': 'valeur2'}
>>>
x['cle2']='Bonjour'
>>> x
{'cle3': 'valeur3', 10: 'valeur10', 'cle1': 'valeur1', 'cle2': 'Bonjour'}
>>>
```

Connaître le type des variables dans Python

La commande type() return le type de la variable

```
>>> pi
3.141592653589793
>>> x=1
>>> type(x)
<class 'int'>

>>> type(pi)
<class 'float'>

>>> flag=True
>>> type(flag)
<class 'bool'>
>>>
```

```
>>> s='bonjour'
>>> s
'bonjour'
>>> type(s)
<class 'str'>

>>> s=5
>>> type(s)
<class 'int'>
>>>
```

Expressions et opérateurs dans Python

Les valeurs et les variables, qui les référencent, sont utilisées en les combinant avec des **opérateurs** pour former des **expressions**.

Exemple :

```
a, b = 7.3, 12
y = 3*a + b/5
```

Certaines fonctions mathématiques sont disponibles après importation d'une librairie « math » :

```
>>> from math import *
>>> pi
3.141592653589793
>>> sqrt(4)
2.0
>>>
```

Opérateurs

les opérateurs Python sont classiques (langage C++) :

+	Addition
-	Soustraction
*	Multiplication
/	Division (avec chiffre après le symbole des décimales)
//	Partie entière d'une division
%	Reste d'une division (modulo)
**	Puissance
==	Test d'égalité
!=	Test de différence
>=	Test supérieur ou égal
<=	Test inférieur ou égal
>	Test supérieur à
<	Test inférieur à

Il existe en plus de ces opérateurs les instructions « and », « or » et « not » qui peuvent servir dans des tests.

Les tests conditionnels

- if, elif, else

```
if <test1>:
    <instructions>
    ……..
elif <test2>:
    <instructions>
    .....
else:
    <instructions>
    ……
```

Exemple

```
if a > 0:
        #code 1
elif a < 0:
        #code 2
else: #a vaut 0
        #code 3
```

La boucle while

La boucle while permet d'effectuer des opérations tant que la condition de la boucle est remplie

```
while <condition> :
    <instructions>
```

Exemple

```
>>> a=0
>>> while (a<4):
    a=a+1
    print(a,a**2,a**3)
```

```
1 1 1
2 4 8
3 9 27
4 16 64
```

La boucle for

La boucle for en Python n'a pas le même fonctionnement que pour d'autres langages (tel que le C). Elle permet de parcourir le contenu d'une variable, d'un objet

```
For <cible> in <objet> :
    <instructions>
```

Exemple

```
>>> ma_chaine="Hello"
>>> for letter in ma_chaine:
    print(letter)

H
e
l
l
o
```

La boucle for …in range …

L'instruction range permet de parcourir une liste croissante d'entiers depuis le paramètre 1 jusqu'au paramètre 2 suivant le pas du paramètre 3.

Exemple

```
>>> for i in range(10):
    x=2
    print(i,x*i)
```

```
0 0
1 2
2 4
3 6
4 8
5 10
6 12
7 14
8 16
9 18
```

```
>>> for i in range(2,10,3):
        print(i)
```

```
2
5
8
```

Les entrées – sorties standards

- print() permet d'afficher sur la console texte le contenu d'une variable ou le résultat d'une expression. Des options de paramétrages sont possibles sur cette instruction.

- input() permet de saisir des caractères au clavier en passant un libellé. La valeur saisie est une chaine de caractères (str)

- Pour saisir une valeur numérique, il est nécessaire de convertir (cast) à la saisie les caractères.

```
>>> a=input()
5
>>> type(a)
<class 'str'>
>>> a=int(input())
5
>>> type(a)
<class 'int'>
```

```
>>> a=float(input())
5
>>> type(a)
<class 'float'>
>>> a
5.0
```

Les fonctions

- Python dispose de plusieurs fonctions prédéfinies dans le langage comme print, input, etc …

- L'instruction « def » permet de définir sa ou ses propres fonctions. Dans le cas ou l'instruction return est utilisée au sein de la fonction, la valeur retournée (qui peut être de tout type) est assignée à la fonction appelante.

Exemple

```
def nom_fonction(arg1, arg2, ...):
        instruction1
        instruction2
        .................
        return valeur
```

Les fonctions, passage de paramètres

Les paramètres sont passés par référence, ces références sont associées aux noms des paramètres lors de l'appel.

Exemple

```
a=2
b=4
def f(c):
    print("c : ",c)
    print("id c : ",id(c))
    c=5
    print("c : ",c)
    print("id c : ",id(c))

print("a : ",a)
print("id a : ",id(a))
f(a)
print("a : ",a)
print("id a : ",id(a))
```

Les fonctions, paramètres avec valeurs par défaut

Les paramètres peuvent avoir des valeurs par défaut. Dans ce cas il est possible de ne pas passer de valeurs sur ce paramètre qui prendra la valeur par défaut.

Exemple

```
def func(x, y, z=-1, t=-2):
    print x, y, z, t

u, v, x, y = 0, 1, 2, 3

func(x, y)
# 2 3 -1 -2
func(y=10, x=3)
# 3 10 -1 -2
func(1, 2, 3, 4)
# 1 2 3 4
func(x=1, t=5, y=3)
# 1 3 -1 5
func(u, v, x, y)
# 0 1 2 3
```

Les fonctions, nombre de paramètres indéfinis

Les paramètres peuvent ne pas être définis et sont remplacés par le caractère * suivi d'un argument. Ceci correspond à une liste de valeurs.

Exemple : def f (*arg): # arg représente la liste

```
def fonction(*arg) :
    for element in arguments :
    print(element)

fonction(43, 38, « Bonjour'', True)

43
38
Bonjour
True
>>>
```

Les fonctions, nombre de paramètres indéfinis

Les paramètres peuvent ne pas être définis et sont remplacés par le caractère ** suivi d'un argument. Ceci correspond à un dictionnaire de clés:valeurs.

Exemple : def f (**arg): # arg représente le dictionnaire

```
>>> def fct(**arg):
        for elt in arg:
            print(elt,' ',arg[elt])


>>> fct(a=89,b=56)
a   89
b   56
>>> fct(a=1,b=5,toto='bonjour')
toto   bonjour
a   1
b   5
```

Python et les fichiers textes

- Les fichiers textes de type séquentiel sont des suites de caractères organisés en une suite de lignes et enregistrés sur un support pouvant être physique (disque dur, clé USB, etc …). les fichiers HTML ou XML sont des fichiers textes.

- Avant toute opération, il est nécessaire d'ouvrir le fichier et de configurer son mode d'utilisation.

  Exemple :

```
f=open('fichier1.txt','w')  # ouverture en mode écriture
f.write('Bonjour\n')    # \n pour le retour à la ligne
f.write ('Mon premier fichier texte')
f.close()   # fermeture du fichier
```

- Les différents modes

  'w' Ecrire dans le fichier
  'a' Ajouter
  'r' Lire

Python et les fichiers textes

- Un fichier texte peut être lu au niveau caractère, ligne ou dans sa totalité

- Au niveau caractère

```
f=open('fichier1.txt','r')
c=f.read(1)            # lit un caractère
while c!='':
   print(c)
   c=f.read(1)
f.close()   # fermeture du fichier
```

- Au niveau ligne

```
f=open('fichier1.txt','r')
for l in f:
print(l.strip('\n\r')) # supprime le retour chariot
f.close()   # fermeture du fichier
```

- Au niveau du fichier

```
f=open('fichier1.txt','r')
print(f.readlines())
f.close()   # fermeture du fichier
```

Python et les fichiers binaires

- Les fichiers binaires sont utiles lorsqu'il faut une écriture ou lecture rapide des données (correspond à une suite d'octets).

- Un fichier binaire peut être lu octet par octet ou par bloc d'octets. Un accès indicé est possible dans ce cas.

- Les différents modes
  - 'wb' Ecrire dans le fichier
  - 'ab' Ajouter
  - 'rb' Lire

Python et la gestion des exceptions

- Une exception est un objet qui indique que le programme ne peut continuer son exécution.

- Python propose un mécanisme de protection de codes au moyen des instructions suivantes try, except, else

```
a,b=12,4
try:
        resultat = a/ b
except NameError as nerr:
        print("Erreur", nerr)
except TypeError as err:
        print("Erreur",err)
except ZeroDivisionError as err1:
        print("Erreur", err1)
else:
        print("Le résultat obtenu est", resultat)
```

Le mode Debug avec IDLE Python

L'éditeur IDLE permet en mode débogage de suivre le déroulement du programme et de connaitre les contenus des différentes variables.
Pour lancer le débogueur à partir de IDLE, choisir dans le menu Debug ->Debugger
Cocher les cases Source et Globals dans la fenêtre de Debug Control.
Lancer le programme Run (F5) sur le fichier source à exécuter.

Le débogueur possède 5 commandes

Go	exécution jusqu'à un point d'arrêt
Step	exécution pas à pas
Over	exécution sans rentrer dans les fonctions
Out	pour sortir d'une fonction en cours
Quit	termine le programme