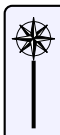
Variables et tests

Spé ISN

À partir d'une version originale de M.Duclosson



Introduction

Dans ce chapitre, on étudie un premier programme écrit en Python qui permettra d'avoir un aperçu de ce langage. On étudiera ensuite la notion de variable et leurs affectations puis les tests.

I Un premier programme en Python

Définition 1 ***(Langage de programmation)***

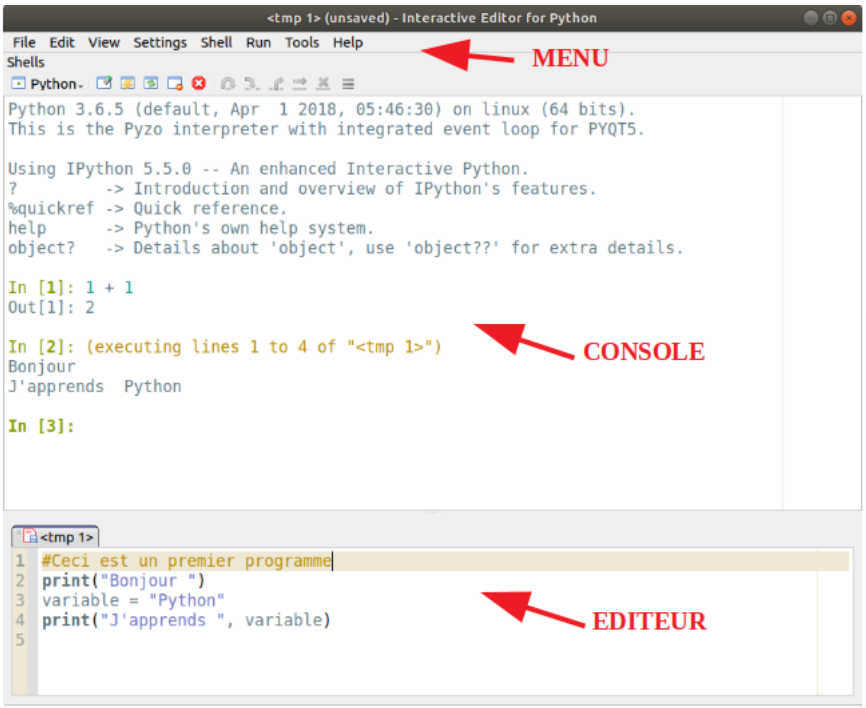
Un **programme informatique** est un texte qui décrit les opérations que l'on souhaite faire exécuter par un ordinateur. Ce texte est écrit en suivant les règles d'un **langage de programmation**. Dans ce cours, nous utiliserons le langage **Python** (version 3).

Un langage de programmation fait interface entre l'homme et la machine, il est compréhensible par le premier et exécutable par la seconde. Un langage dont la structure est proche du fonctionnement du processeur [1] est dit de bas niveau. Réciproquement, un langage de haut niveau doit être plus facile à comprendre. Python est un langage de haut niveau.
Pour rédiger et mettre au point un programme on utilise généralement un Environnement de Développement Intégré : EDI (ou IDE en anglais). Pour Python il y en existe plusieurs : IDLE, Spyder, Pyzo etc. Nous utiliserons l'environnement Pyzo.

L'interface de Pyzo, comporte trois zones :

- La **console** permet de saisir une instruction Python après le prompt constitué de trois chevrons »> ou d'une étiquette `In [1]:` . On marque la fin de la saisie avec un retour chariot puis l'interpréteur Python donne sa réponse en-dessous.
- **L'éditeur de texte** permet d'écrire un programme constitué de plusieurs instructions et de l'enregistrer sur le disque.
- **La barre de menus**, permet de dérouler différents menus : nous utiliserons principalement les menus File, Edit et Run. Le menu Run propose différentes formules et les résultats de l'exécution du programme s'affichent dans la console.

1. Le processeur est le composant de l'ordinateur qui exécute les instructions du langage machine.



Dans votre répertoire personnel, créez un dossier `ISN` puis à l'intérieur, un dossier pour le premier chapitre : `Ch1_Variables_et_tests`.
On veillera à toujours respecter la règle suivante : les noms de fichier ou de dossier ne doivent pas comporter d'espace, d'accents ou de caractères spéciaux. Pour séparer les mots, vous pouvez utiliser - (tiret) ou _ (tiret-bas ou underscore).
Dans ce dossier, créer un fichier `Devinette.py` et reproduire le code qui suit.

```
1  from random import randint
2
3  n = randint(1, 100)
4  print('Python vient de choisir un nombre entre 1 et 100.')
5  r = int(input('Devinez-le : '))
6  while r != n:
7      if r < n:
8          print('Trop petit !')
9      else:
10         print('Trop grand !')
11     r = int(input('Essayez encore : '))
12 print('Bravo, vous avez trouvé !')
```

Sauver puis exécuter le code à l'aide du menu `Run` (noter le raccourci clavier au passage).

Que fait ce programme ? .....
.....
.....

Commenter la ligne 1 en la faisant précéder du caractère #. Exécuter le programme, que se passe-t-il ?
.....
.....
.....

Que se passe-t-il à l'exécution de l'instruction `n = randint(1, 100)` ? De quel type est-elle ?

.....

.....

.....

Quel est le rôle de l'instruction `print('Python vient de choisir un nombre entre 1 et 100.')` ?

.....

.....

Que permet l'instruction `r = int(input('Devinez-le : '))` ?

.....

.....

.....

Une traduction de ce programme en langage C++ serait :

```
#include <iostream>
#include <stdlib.h>
using namespace std;

int main()
{
    int n = rand() % 100 + 1;
    int r;
    cout << "C++ vient de choisir un nombre entre 1 et 100." << endl;
    cout << "Devinez-le : ";
    cin >> r;
    while (r != n){
        if (r < n) {
            cout << "Trop petit" << endl;
        }
        else {
            cout << "Trop grand" << endl;
        }
       cout << "Essayez encore" << endl;
       cin >> r;
    }
    cout << "Bravo vous avez trouvé !";
    return 0;
}
```

Quelles différences de syntaxe entre `Python` et `C++` apparaissent de façon significative ?

.....

.....

.....

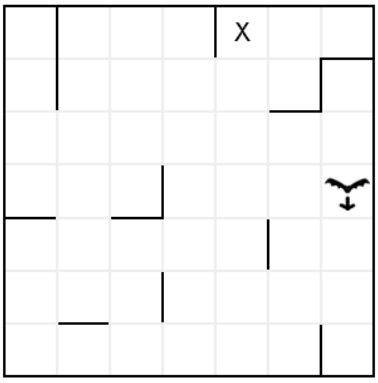
.....

.....

.....

.....

Exercice 1 *Suite d'instructions*



Vous avez mis au point un petit robot chauve-souris pour explorer des labyrinthes.
Malheureusement son système de vision n'est pas encore au point : il lui est impossible de voir un mur avant de rentrer dedans !
Vous décidez cependant d'en faire une démonstration à l'un de vos amis, dans un labyrinthe dont le plan est représenté ci-dessus.

La chauve-souris répond aux trois instructions suivantes :

- ☞ `droite()` : elle fait un quart de tour sur sa droite.
- ☞ `gauche()` : elle fait un quart de tour sur sa gauche.
- ☞ `fonce()` : elle va tout droit jusqu'à rencontrer un mur.

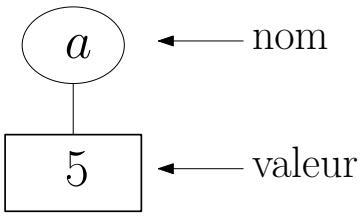
Écrire une suite d'instructions qui lui permette d'atteindre la case d'arrivée, marquée d'un « X » sur le plan.

II Variables et affectation

Pour stocker et manipuler les informations élémentaires on utilise des **variables** qui associent à un nom une valeur : une variable est une boîte qui porte un nom.

Pour donner une valeur à une variable on utilise une instruction fondamentale : **l'affectation**. Sa syntaxe utilise le symbole = sous la forme :

```
nom_variable = expression
```



Attention, le symbole = n'a rien à voir avec le égal des mathématiques, il n'est pas symétrique. En pseudo-code il s'écrit par exemple `a` ← `5`. Permuter les deux parties donne lieu à une erreur :

```
>>> 5 = a
SyntaxError: can't assign to literal
```

• Le **nom de la variable** peut comporter plus d'une lettre mais il faut respecter quelques règles :

- On peut utiliser les lettres minuscules ou majuscules (a → z et A → Z) qui sont différentiées ainsi que les 10 chiffres (0 → 9). Les autres caractères sont interdits sauf le tiret bas _ (mais pas le tiret usuel - car c'est le symbole de la soustraction).
- Le premier caractère est obligatoirement une lettre.
- Les mots réservés du langage sont interdits :

```
  >>> else = 5
SyntaxError: invalid syntax
```

- Utiliser un nom de fonction ne provoquera pas d'erreur directe mais c'est absolument à éviter :

```
>>> print('Hello Word')
Hello Word
>>> print = 1789
>>> print('Hello word')
Traceback (most recent call last):
  File "<pyshell#28>", line 1, in <module>
    print('Hello word')
TypeError: 'int' object is not callable
```

Expliquer l'erreur précédente :

.....

.....

• L'expression est évaluée et donne une **valeur** qui est affectée à la variable en écrasant une éventuelle valeur précédente.

```
>>> a = 5
>>> a
5
>>> b = a*(a + 1)
>>> b
30
>>> a = max(1, a, 50)
>>> a
50
```

Définition 2 ***(Expression)***

Une **expression** est une combinaison de valeurs explicites (appelées littéraux), de variables, d'opérateurs et de fonctions qui est évaluée pour donner lieu à une valeur.

Lors de l'exécution d'une instruction d'affectation (après la vérification de la syntaxe), il y a d'abord évaluation de l'expression puis la valeur obtenue est attribuée à la variable.

Des erreurs classiques :

```
>>> a - 1 = 5
SyntaxError: can't assign to operator
```

.....

.....

```
>>> u = v
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    u = v
NameError: name 'v' is not defined
```

.....

.....

L'incrémentation :

```
>>> a = 1789
>>> a
1789
>>> a = a + 1
>>> a
1790
```

Exercice 2

Quelle instruction provoquera la décrémentation de la variable a ? Son doublement ?

.....

.....

.....

.....

Essayons un premier code pour permuter les valeurs de deux variables :

```
>>> x = 1
>>> y = 100
>>> y = x
>>> x = y
>>> x
1
>>> y
```

```
1
```

Expliquer ce qu'il s'est passé : .....

.....

.....

.....

.....

La bonne manière de procéder est d'utiliser une variable auxiliaire :

```
>>> x = 1
>>> y = 100
>>> aux = x
>>> x = y
>>> y = aux
>>> x
100
>>> y
1
>>> (x, y)    # Pour afficher les valeurs de plusieurs variables
(100, 1)
```

Entraînement 1

Comment obtenir la permutation circulaire des valeurs des trois variables x, y et z?

Exercice 3

Déterminer si les suites de symboles sont des expressions ou des instructions.

* $x + 1$
* $x = y + 1$
* $x < y + 1$

Exercice 4

Prévoir le résultat des séquences d'instructions :

```
>>> a = 10
>>> b = 7
>>> c = a + b
>>> a = 3
>>> c = a - b
>>> (a, b, c)
```

```
>>> a = 1
>>> b = a
>>> b = 10
>>> c = a
>>> a = b
>>> (a, b, c)
```

.....

.....

.....

Définition 3 ***(État courant d'un programme)***

Lors de l'exécution d'un programme, l'ensemble des variables et plus généralement l'ensemble des données en mémoire à un instant donné s'appelle l'état d'exécution ou l'état courant du programme.

Suivre l'évolution de l'état courant d'un programme permet souvent de le comprendre en profondeur et de le mettre au point. On peut utiliser un tableau comme ci-contre.

ligne	état courant après		
	a	b	c
1			

Le site http://www.pythontutor.com/ vous permet de visualiser l'état courant de vos programmes.

Entraînement 2

Que font les instructions suivantes :

```
>>> x = x + y
>>> y = x - y
>>> x = x - y
```

III Type

Définition 4 ***(Type)***

Chaque variable possède un type qui est déterminé au moment de l'affectation mais il peut changer (on parle de typage dynamique). Le type indique la nature des données qu'une variable contient.

Les types Python les plus courants en ISN seront :

type Python	"traduction"	exemple
`int`	entier	1871
`float`	flottant (décimal)	3.1415
`bool`	booléen (`True` ou `False`)	`True`
`tuple`	n-uplet	`(2, 0, 1, 7)`
`str`	chaîne de caractères (string)	`'ISN'`
`list`	liste	`[9, 8, 3.14, 'Pi']`
`function`	fonction	`max`

On obtient le type d'une variable par la fonction `type` :

```
>>> x = 101
>>> type(x)
<class 'int'>
>>> x = 101.
>>> type(x)
<class 'float'>
>>> y = 101
>>> type(x == y)
<class 'bool'>
>>> x == y
True
>>> type(False)
<class 'bool'>
```

```
>>> a = (1, 3.14159)
>>> type(a)
<class 'tuple'>
>>> ch = 'Hello word'
>>> type(ch)
<class 'str'>
>>> L = [1, 'abcdefgh', 2.71]
>>> type(L)
<class 'list'>
>>> def f(x):
    return x + 1

>>> type(f)
<class 'function'>
```

Noter l'utilisation du caractère ' pour délimiter les chaînes de caractères.

Quelques opérateurs usuels

On notera que certains peuvent s'appliquer à des objets de types différents ; ils sont polymorphes.

Opérateur	**Type**	**Action**
+	int, float	Addition
-	int, float	Soustraction
*	int, float	Multiplication
**	int, float	Puissance
/	float	Division décimale
//	int	Quotient de la division euclidienne
%	int	Reste de la division euclidienne
+	str	Concaténation de deux chaines de caractères
not	bool	Négation (False ↔ True)
and	bool	Et logique
or	bool	Ou logique

Les fonctions mathématiques usuelles sont utilisables moyennant leur importation :

```
>>> cos(4)
Traceback (most recent call last):
  File "<pyshell#99>", line 1, in <module>
    cos(4)
NameError: name 'cos' is not defined
>>> from math import cos
>>> cos(4)
-0.6536436208636119
```

IV Tests

Dans l'exemple du jeu de devinette, nous avons déjà rencontré un test. La syntaxe générale est :

```
if condition1:
    instruction     #
      ...           # bloc d'instructions 1
    instruction     #
elif condition2:
    instruction     #
      ...           # bloc d'instructions 2
    instruction     #
    instruction
elif ...
    .....
else :
    instruction     #
      ...           # dernier bloc d'instructions
    instruction     #
```

Seul le premier mot clé `if` est obligatoire, les autres `elif` et `else` sont optionnels. Noter que `else` n'est pas suivi d'une condition.
Les conditions sont des variables de type booléen ou des expressions qui sont évaluées sous forme de booléen. Attention, dans cette situation, Python évalue en booléen des types qui n'en sont pas a priori.
Si `condition1` a pour valeur `True` alors le bloc d'instructions 1 est exécuté.
Si ce n'est pas le cas, `condition2` est évaluée. Si sa valeur est `True` alors le bloc d'instructions 2 est exécuté.
Sinon on passe au `elif` suivant et ainsi de suite.
Si aucune des conditions présentes derrière un des mot-clé `elif` n'est évaluée à True alors le dernier bloc est exécuté.
Remarque : Un seul des blocs d'instructions peut être exécuté : le premier possible ceci même si l'état courant rend plusieurs des conditions valides (`True`).
Les conditions sont souvent construites à l'aide des opérateurs de comparaison.

Opérateur	test effectué
==	égalité
!=	différent
<=	inférieur ou égal
>=	supérieur ou égal
<	inférieur strict
>	supérieur strict

Exercice 5

Expliquer la différence entre les deux codes. On retrouvera cette difficulté avec les boucles.

```
if bool:
    instruction1
else:
    instruction2
    instruction3
```

```
if bool:
    instruction1
else:
    instruction2
instruction3
```

.....

.....

.....

Entraînement 3

On suppose que les variables x, y, et z ont respectivement pour valeur 2, 10 et 10. Quel état produit l'exécution des instructions suivantes :

```
if x == 2:
    y = 5
else:
    y = 6
    z = 7
```

```
if x == 2:
    y = 5
else:
    y = 6
z = 7
```

Tester ou ne pas tester un booléen ?

Supposons que l'on dispose d'une fonction `est_premier(n)` qui renvoie le booléen True si n est premier et False sinon. Que pensez-vous des deux codes suivants :

```
n = input('Entrez n : ')
if est_premier(n) == True:
    print('n est premier')
else:
    print('n est composé')
```

```
n = input('Entrez n : ')
if est_premier(n):
    print('n est premier')
else:
    print('n est composé')
```

L'écriture de gauche est une maladresse (coupable ?) qui revient à la formulation : « Si il est vrai que n est premier alors ... ». On préfère simplement « Si n est premier alors .... ».

Exercice 6

Écrire un programme qui affiche le plus grand de deux nombres saisis par l'utilisateur (sans utiliser la fonction `max`).

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Entraînement 4

La conjecture de Syracuse étudie la fonction qui à un entier n associe $n/2$ si n est pair et $3*n+1$ s'il est impair. Écrire un programme qui affiche l'image d'un nombre n saisi par l'utilisateur.

V Ressources en ligne et installation de Python

Quelques liens vers des ressources :

- Site officiel : https://www.python.org/ https://www.python.org/download/ https://www.python.org/doc/
- Les bases de Python pour le lycée : http://www.frederic-junier.org/PythonSeconde/Python_Seconde_Parc/accueil_python_2nde.html
- Tutoriel sur Python : http ://www.tutorialspoint.com/python3/index.htm
- Le site de l'Olympiade française d'informatique (France-ioi) : http://www.france-ioi.org/algo/chapters.php

Python est installé par défaut sur la distribution Linux Ubuntu, sous Windows ou MAC OSX on peut télécharger les installateurs à partir de http://www.python.org/download/ et on disposera d'un environnement graphique Idle avec console et éditeur de texte. Anaconda est une distribution plus complète (avec tous les module et bibliothèques nécessaires) qui se télécharge depuis https://www.anaconda.com/distribution/. Anaconda intègre plusieurs IDE comme Idle et Spyder mais aussi le notebook Jupyter.
Tout interpréteur Python peut être associé à l'IDE Pyzo qui se télécharge et s'installe très facilement depuishttps://pyzo.org/.
Il existe aussi des interpréteurs en ligne comme https://repl.it/languages/python3.