02 | Les structures de données

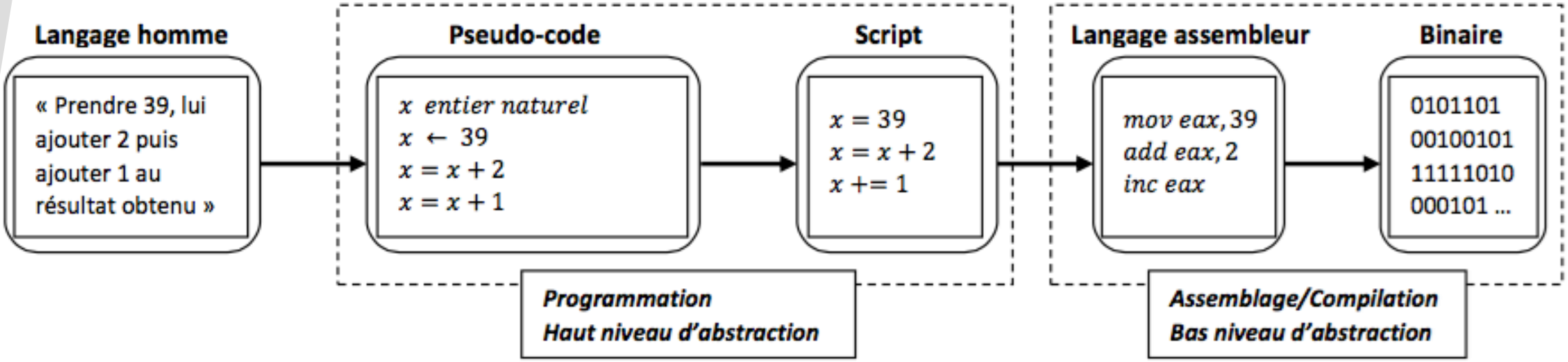
I - Introduction sur les niveaux d'abstractions

Python est un langage:

- de **très haut niveau d'abstraction**
  ⇒ simplicité mais perte d'efficacité
- **impératif**
  ⇒ le programme exécute des instructions successives
- **orienté objet**
  ⇒ on agit sur des structures de données
- permettant une **approche fonctionnelle**
  ⇒ il est possible de décomposer un programme principal en fonctions

Ce langage peut être utile en phase de prototypage et être suivi d'une reprogrammation en C.

I - Introduction sur les niveaux d'abstractions



II - Les composants et leurs connexions

II.1 Les expressions

Une expression est une combinaison de valeurs, de variables, d'opérateurs et de fonctions, qui est évaluée en suivant les règles de calcul du langage de programmation pour retourner une nouvelle valeur.

Exemple :

- 3 + 2 est une expression dont la valeur est 5
- True or False est une expression dont la valeur est True

II - Les composants et leurs connexions

II.2 Les types de variables

Nombre			Booléen	Chaîne de caractère	Tableaux			Autres objets	
Entier	Flottant	Complexe			n-uplet	Liste		Ensemble	Autre
int	*float*	*complex*	*bool*	*str*	*tuple*	*list*	*array*	*set*	Dict, class, file, function, …
42	42.0	1+2j	True/False	"Hello word!" "42"	(4,2)	[4,2]	array([4,2])	{4,2}	…

Remarques :

- le type *array* est un type importé de *numpy*
- L'imaginaire i est remplacé par j sous Python

II - Les composants et leurs connexions

II.3 Opérations numériques, Opérateurs booléens et de comparaisons

Python	Résultat	Exemple	Sortie (output)
x + y	Somme de x et y	2+3	5
x – y	Différence de x et y	2-3	-1
x * y	Produit de x et y	2*3	6
x / y	Division de x par y	3/2.	1.5
x //y	Quotient de la division entière de x par y	7//2	3
x % y	Reste de la division entière de x par y	7%2	1
-x	Opposé de x		
abs(x)	Valeur absolue de x	abs(-3)	3
int(x)	Conversion de x en l'entier le plus proche de 0	int(-4.9)	-4
float(x)	Conversion de x en flottant	float(2)	2.0
divmod(x,y)	Le doublet (x//y , x % y)	divmod (7,3)	(3,1)
x**y	x puissance y	2**7	128

II - Les composants et leurs connexions

II.3 Opérations numériques, Opérateurs booléens et de comparaisons

Python	Résultat	Exemple	Sortie (output)
a.conjugate( )	Conjugué d'un nombre complexe	(3-4j).conjugate	3+4j
a.real	Partie réelle du complexe a	(3-4j).real	3.0
a.imag	Partie imaginaire du complexe a	(3-4j).imag	-4.0
abs (a)	Module du complexe a	abs (3-4j)	5.0

Python	Résultat	Exemple	Sortie (output)
x or y	Vrai si x ou y sont vrais (ou les deux)	0 or 1	True
x and y	Vrai si x et y sont vrais tous les deux	0 and 1	False
not x	Vrai si x est faux et inversement	not (0)	True

II - Les composants et leurs connexions

II.3 Opérations numériques, Opérateurs booléens et de comparaisons

Python	Résultat	Exemple	Sortie (output)
<	Strictement inférieur à	3 < 3	False
<=	Inférieur ou égal à	3 <= 3	True
>	Strictement supérieur à		
>=	Supérieur ou égal à		
==	Egal à	4 == 2	False
!=	Différent de	2 != 3	True

III - Les variables

Les variables ne sont rien d'autre qu'une réservation d'espace mémoire pour stocker des valeurs. L'opérateur d'affectation est le = avec **à gauche le nom de la variable** et **à droite la valeur à stocker dans la variab**le.

L'ensemble des variables ainsi que les valeurs stockées dans ces variables à un instant t définissent le « **contexte** ».

Remarques :

- Pour faciliter la lecture d'un programme il est conseillé de donner des noms explicites aux variables.
- On peut affecter à une variable le résultat d'une expression dans laquelle cette variable intervient. (a = a + 2)
- La syntaxe Python introduit des opérateurs comme += (-= , *= , /= , …) afin de simplifier ce type d'expression (a = a + 2 devient alors a+=2).

IV - Types Séquences

IV.1 Introduction

Les séquences sont des types informatiques qui permettent de stocker plusieurs valeurs d'une manière efficace et organisée.

Nous allons voir quatre types séquences de Python :

- les list
- les array (tableau)
- les tuples
- les str (chaînes de caractères)

Une séquence est dite **immuable** si les valeurs stockées dans la séquence ne peuvent pas être modifiée après création (exemple : les tuples et les str). On parle de séquence **variable** dans le cas contraire (exemple : les list).

IV - Types Séquences

IV.2 Les listes

Les listes sont des **séquences variables**. Pour créer une liste on utilise des crochets : [ ] (L=[] donne la liste vide). Les valeurs à l'intérieur de la liste peuvent être de types quelconques et sont séparés par des virgules.

Exemple : L=[“soleil”, 2.5 , 4 ,”informatique”]

IV - Types Séquences

IV.2 Les listes

Opération	Résultat
x in s	True si un élément de s est égal à x, False sinon
x not in s	False si un élément de s est égal à x, True sinon
s + t	Concaténation de s et t
s* n ou n * s	n copies de s concaténées
len(s)	Nombre d'éléments contenus dans s
min(s)	Le plus petit élément de s
max(s)	Le plus grand élément de s
s[i]	i-ème élément de s (attention les indices commencent à 0 !!)
s[i :j]	Séquence composée des éléments de s du i-ème inclus au j-ème exclus
s[i :j :k]	Séquence composée des éléments de s du i-ème inclus au j-ème exclus avec un pas k.
s[i] = x	i-éme élément remplacé par x
del s[i :j :k]	Les éléments de s[i :j :k] sont supprimés
s.append(x)	Ajoute x à la fin de la liste s
s.clear()	Supprime tous les éléments de s
s.insert(i,x)	Insère x à la i-ème place dans s
s.pop[i]	Renvoie s[i] et le supprime de s
s.remove(x)	Supprime le premier élément de s tel que s[i] == x
s.reverse()	Inverse la place des éléments de s

IV - Types Séquences

IV.3 Les tableaux

Pour utiliser les tableaux il est nécessaire d'importer le module **numpy** (**import numpy as np**).
Les tableaux sont des séquences dont la taille est fixée à la création et dont **tous les éléments contenus doivent être de même type**.

Un tableau peut être créé à partir d'une liste, d'un n-uplet ou à l'aide de la fonction **arange(début inclus, fin exclus, pas)**.

IV - Types Séquences

IV.3 Les tableaux

```
>>> a = np.array([3,4,5,6])
>>> b = np.array((5.,4.,6.))

>>> a.dtype
dtype('int64')

>>> b.dtype
dtype('float64')

>>> np.arange(1,10,2)
array ([1, 3, 5, 7, 9])
```

IV - Types Séquences

IV.3 Les tableaux

Autres outils :

- **linspace**

```
>>> np.linspace (0.1 , 1.3, 7)                    # 7 nombres de 0.1 à 1.3 inclus
array ([ 0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3 ])
```

IV - Types Séquences

IV.3 Les tableaux

Autres outils :

- **linspace**
- **shape**

```
>>> a = np.arrange(12)
>>> a
array ([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 ])

>>> a.reshape(3,4)
array (   [0, 1, 2, 3] ,
          [4, 5, 6, 7],
          [8, 9, 10, 11] ] )

>>> b = np.zeros ((3,4))          # np.ones((,)) fonctionne de la même façon avec des 1.
>>> b
array (   [0., 0., 0., 0.] ,
          [0., 0., 0., 0.],
          [0., 0., 0., 0.] ] )
```

IV - Types Séquences

IV.4 Les tuples

Les tuples (n-uplets) sont des **séquences immuables** qui peuvent être construite de différentes manières :

- En utilisant des parenthèses pour le tuple vide ( )
- En utilisant une virgule pour un tuple constitué d'un élément a, ou (a,)
- En séparant les éléments par une virgule a, b, c ou (a, b, c)

IV - Types Séquences

IV.4 Les chaînes de caractères

Ce sont des séquences immuables de caractères, qui peuvent être définies avec des guillemets.

```
>>> a = '' Informatique ''
>>> a[4]
'r'

>>> a[2 :6]
'form'

>>> a[3] = 'e'
TypeError: 'str' object does not support item assignment
```