Introduction à la programmation en Python

Thierry Lecroq

Université de Rouen
FRANCE

Plan du cours

1. Généralités sur le traitement de l'information
2. Programmation en Python

UNIVERSITÉ DE ROUEN

Références

Gérard Swinnen,
Apprendre à programmer avec Python,
2e édition, O'Reilly, 2005.

Plan

1 Généralités sur le traitement de l'information

2 Programmation en Python

Généralités sur le traitement de l'information

Les ordinateurs sont utilisés pour

- le traitement d'informations ;
- le stockage d'informations.

Généralités sur le traitement de l'information (2)

Le schéma global d'une application informatique est toujours le même :

réception		traitement des		émission
	$\Longrightarrow$		$\Longrightarrow$	
d'informations		informations		d'informations déduites

Exemple

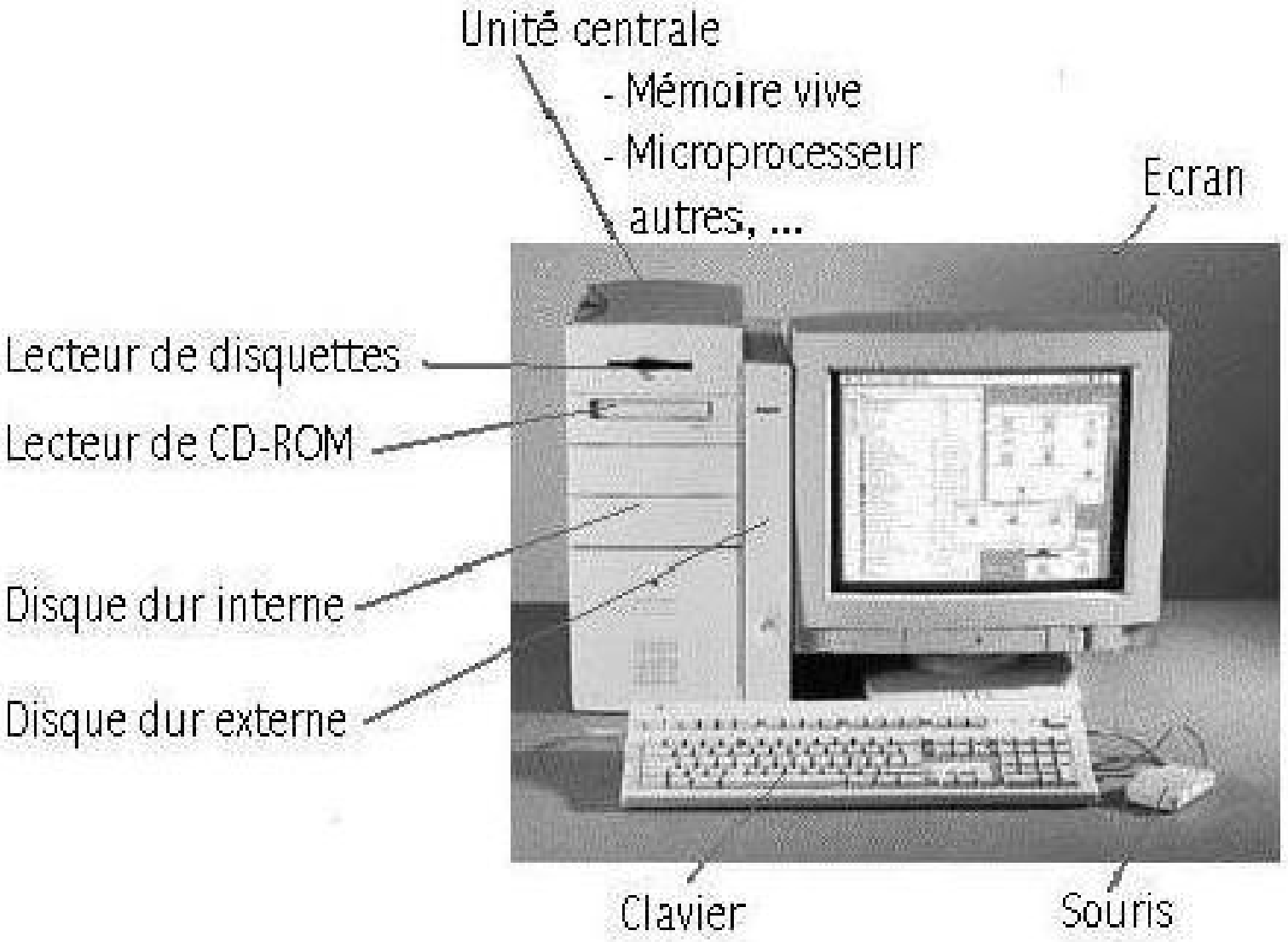
réception de		somme de ces n valeurs		émission de la
n valeurs	$\Longrightarrow$	et division	$\Longrightarrow$	moyenne
numériques		du résultat par n		arithmétique

Tout traitement demandé à la machine, par l'utilisateur, se traduit par l'exécution séquencée d'opérations (**instructions**).

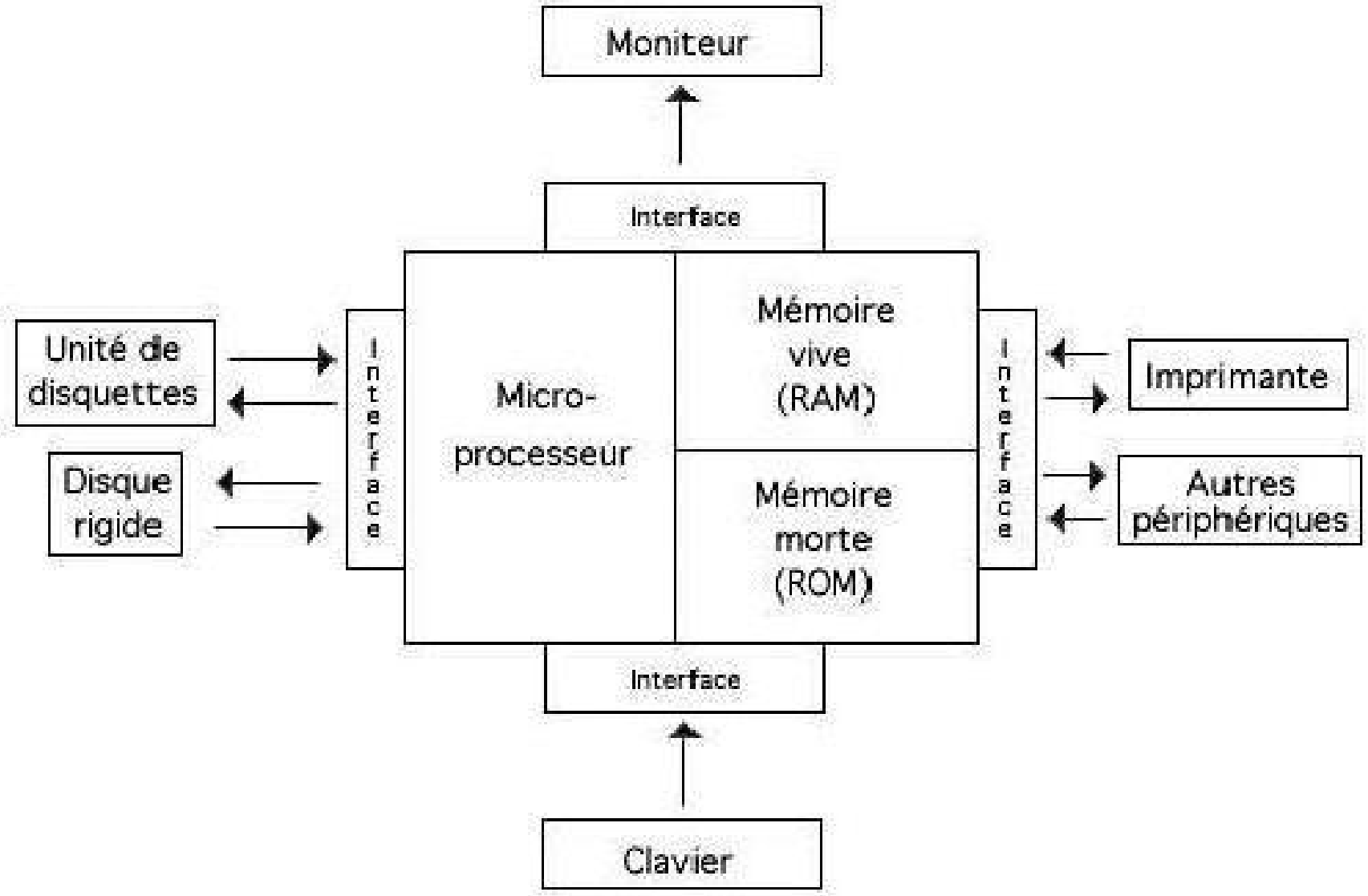
La notion de programme

Données $\Longrightarrow$ Programme $\Longrightarrow$ Résultats

Organisation matérielle



Organisation matérielle



architecture conceptuelle d'un ordinateur.

L'unité centrale

Elle contient le ou les micro-processeurs.

- unité arithmétique et logique ;
- unité de commande ;
- mémoire centrale ;
- bus ;
- horloge.

L'unité arithmétique et logique

- unité de traitement arithmétique ;
- unité de traitement logique ;
- registres.

L'unité de commande

- coordonne l'ensemble des tâches ;
- est en relation avec la mémoire principale ;
- est associée au registre à instruction.

La mémoire centrale

La mémoire centrale est directement reliée à l'unité centrale et contient le ou les programmes à exécuter.

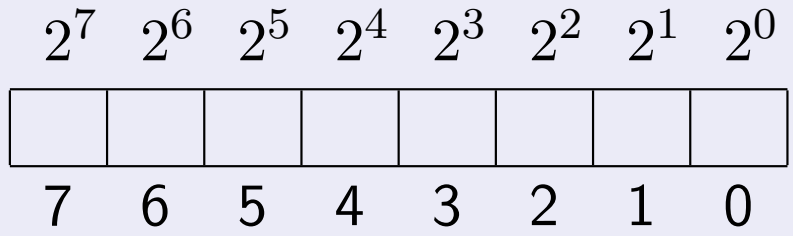
L'organisation de la mémoire

- plus petite information : bit (contraction de *binary digit*)
- 0 ou 1

L'organisation de la mémoire (2)

On a l'habitude de regrouper les bits :

- par groupe de 8 bits : **octet** (**byte**)



  Le bit en position 0 est le bit de poids faible et le bit en position 7 est le bit de poids fort.

- en **mots mémoires**.
  La taille d'un mot mémoire est généralement une puissance de 2, cela correspond à la taille du **bus**.
  Cette taille varie suivant les machines et les constructeurs.

La taille des mémoires

- kilooctet : 1 ko $= 1\,024$ $(2^{10} \approx 10^3)$ octets ;
- mégaoctet : 1 Mo $= 1\,048\,576$ $(2^{20} \approx 10^6)$ octets ;
- gigaoctet : 1 Go $= 1\,073\,742\,824$ $(2^{30} \approx 10^9)$ octets ;

Les différents types de mémoire

- RAM (*Random Access Memory*) ;
- ROM (*Read Only Memory*).

Le bus

- cable électrique ;
- transporte les données d'un organe vers un ou plusieurs autres ;
- composé de plusieurs fils ;
- chaque fil transporte une information qui peut prendre deux valeurs : 0 ou 1.

L'horloge

cadence la communication à l'intérieur de l'ordinateur.

Exemple

Considérons l'opération qui consiste à amener une donnée de la mémoire jusqu'au processeur :

1. le processeur place, sur le bus, le numéro (adresse) de la case dans laquelle se trouve la donnée ;
2. le co-processeur gestionnaire de la RAM prend ce numéro sur le bus ;
3. le co-processeur gestionnaire de la RAM lit le contenu de la case dont il vient de recevoir l'adresse, et le place sur le bus ;
4. le processeur récupère la donnée sur le bus.

UNIVERSITÉ DE ROUEN

L'horloge (suite)

En général, la cadence de l'horloge est donnée en GHz (gigahertz). On parle, par exemple, d'ordinateurs $1,5$ GHz, ce qui signifie que la durée d'un cycle est de $\frac{1}{1\,500\,000\,000}$ seconde soit $0,0006$ μ-seconde.

Les organes d'entrées/sorties (E/S ou I/O)

- les unités de visualisation (visuel, visu, moniteur) ;
- le clavier (*keyboard*), la souris (*mouse*), le crayon optique ;
- les imprimantes (*printers*), les traceurs de courbes ;
- les modems (modulateur–démodulateur) ;
- un robot, l'alarme de sa maison, un feu tricolore, ...

- disques durs ;
- lecteurs de disquettes ;
- supports de mémoire amovibles (*stick USB*) ;
- lecteurs/graveurs de CD-ROM, DVD-ROM ;
- lecteurs de bandes magnétiques.

Des processeurs spécialisés (co-processeurs) sont associés à chaque périphérique.

Les systèmes d'exploitation des ordinateurs

Les principales fonctions d'un système d'exploitation sont :

- la gestion et la conservation des informations par l'intermédiaire d'un système de **fichiers** ;
- la gestion de l'ensemble des ressources (processeurs, mémoires, registres, imprimantes, ...) permettant l'exécution d'un programme ;
- fournir à l'utilisateur un langage de commande facile et efficace.

Exemple

Unix, Windows, Linux, MacOS, BeOS, ...

Les différents types de systèmes d'exploitation

- mono-tâche ;
- multi-tâche ;
- mono-utilisateur ;
- multi-utilisateur.

Multimédia et hypertexte

- un ordinateur est **multimédia** s'il peut stocker et traiter des textes, des sons, des images fixes et des images vidéos ;
- un document **hypertexte** est constitué par un ensemble de pages (fichiers) reliées par des liens (renvois) placés dans le texte.

Réseaux

Permettent de connecter plusieurs ordinateurs entre eux.

On peut distinguer au moins deux types de réseaux :

- les réseaux locaux permettent de relier des ordinateurs dans un même lieu à l'aide de câbles ou liaisons optiques ;
- les réseaux distants permettent de relier des ordinateurs distants à l'aide de liaison téléphoniques, satellites, ...

Intérêts

- partage des ressources et des logiciels ;
- communication ;
- transfert d'informations.

UNIVERSITÉ DE ROUEN

Internet

- le réseau des réseaux ;
- les ordinateurs du monde entier sont connectés entre eux à l'aide de câbles, de lignes téléphoniques et de satellites.

Des logiciels spécifiques permettent d'accéder aux services principaux :

- l'accès distant (`telnet`, `ssh`) ;
- la messagerie électronique (`mail`) ;
- le transfert d'informations (`ftp`, `sftp`) ;
- la consultation de forums (`news`) ;
- la consultation de la Toile (*Web* ou *WWW*)

UNIVERSITÉ DE ROUEN

WWW

La Toile (ou *World Wide Web* ou *Web*) est constituée par un ensemble d'informations multimédia contenant du texte, des images, des vidéos, des sons, ...

Les logiciels permettant de *surfer* sur la Toile sont appelés des navigateurs (*browsers*). Les principaux sont *Mozilla* et *Internet Explorer*.

Le langage principal d'écriture de pages *Web* est HTML (*Hyper Text Mark-up Language*).

Il existe des moteurs de recherche (*AltaVista, Google, Kartoo, Yahoo*, ...) pour rapidement localiser des informations à partir de mots clés.

UNIVERSITÉ DE ROUEN

La notion d'algorithme

Pour résoudre un problème de manière informatique il y a un cheminement méthodique à respecter :

$$\text{Problème} \rightarrow \begin{matrix}\text{Énoncé} \\ \text{algo}\end{matrix} \rightarrow \text{Algo} \rightarrow \begin{matrix}\text{Programme} \\ \text{(instructions)}\end{matrix} \rightarrow \text{Résultats}$$

La notion d'algorithme (2)

Algorithme

- description formelle d'un procédé de traitement qui permet à partir d'un ensemble d'informations initiales d'obtenir des informations déduites ;
- succession finie et non ambiguë d'opérations clairement posée ;
- se termine donc toujours.

Programme

- suite d'instructions définies dans un langage donné ;
- décrit un algorithme.

UNIVERSITÉ DE ROUEN

La programmation

- langage machine : directement compréhensible par la machine ;
- langage d'assemblage (ou **assembleur**) : très facilement traduisible pour être compris par la machine ;
- langage de programmation : doit être compilé ou interprêté pour être compris par la machine.

Trois types de langages de programmation : les langages compilés, les langages interprêtés et les langages à *bytecode*.

UNIVERSITÉ DE ROUEN

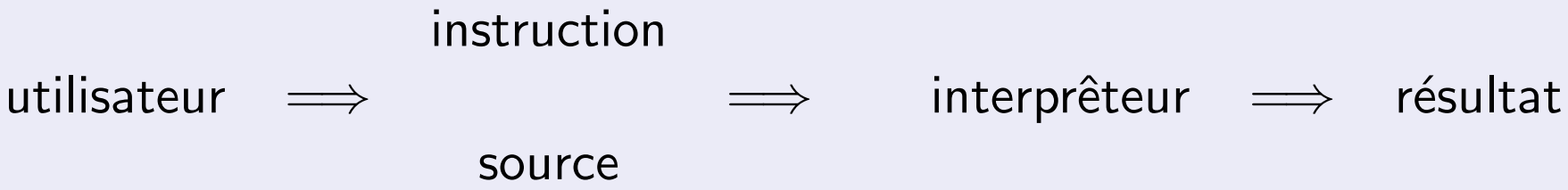
Les langages compilés

utilisateur → programme source → compilateur → programme objet → exécution → résultats

Exemple

Pascal, C, ADA, FORTRAN, ...

Les langages interprêtés



Exemple

Basic, LISP, Perl, ...

Les langages à *bytecode*

utilisateur → programme source → compilateur → *bytecode* → interprêteur → résultats

Exemple

Java, Python, ...

Les différentes couches

- une couche matériel ;
- une couche système d'exploitation ;
- une couche logiciel d'applications (traitements de textes, tableurs, ...) ;
- une couche programme utilisateurs.

Plan

1 Généralités sur le traitement de l'information

2 Programmation en Python

Le langage Python

- créé en 1989 par Guido van Rossum ;
- portable ;
- dynamique ;
- extensible ;
- gratuit ;
- modulaire ;
- orienté objet.

UNIVERSITÉ DE ROUEN

La notion de variable

- on doit être capable de stocker des informations en mémoire centrale durant l'exécution d'un programme ;
- on veut éviter d'avoir à manipuler directement les adresses ;
- on manipule des **variables** ;
- le programmeur donne aux variables des noms de son choix ;
- les variables désignent une ou plusieurs cases mémoires.

UNIVERSITÉ DE ROUEN

La notion de variable (2)

Une variable possède quatre propriétés :

- un nom ;
- une adresse ;
- un type ;
- une valeur.

variables : boites spécifiques dans la mémoire contenant une suite de 0 et de 1.

Règles de formation des identificateurs

Les noms des variables (ainsi que les noms des fonctions) sont appelés des **identificateurs**.
Règles de formation :

- suite de lettres (minuscules `'a'..'z'` ou majuscules `'A'..'Z'`), de chiffres (`'0'..'9'`) et de caractères de soulignement (`'_'`) ;
- premier caractère doit être une lettre ;

Règles de formation des identificateurs (2)

Exemple

- `c14_T0` est un identificateur ;
- `14c_T0` n'est pas un identificateur ;
- `x*y` n'est pas un identificateur.

- donnez des noms significatifs aux variables ;
- évitez le caractère de soulignement ;
- pour former des identificateurs à l'aide de plusieurs mots, écrivez le premier mot en minuscule et pour les mots suivants écrivez les initiales en majuscule et les autres lettres en minuscule.

UNIVERSITÉ DE ROUEN

Les types et les opérations

Les principaux types en Python sont :

- Les entiers ;
- les réels ou flottants ;
- les chaînes de caractères ;
- les listes ;
- les dictionaires.

Les entiers

- représentés sur un mot machine pour les entiers courts ;
- sur une machine à n bits on peut représenter 2^n entiers, soit les entiers compris entre -2^{n-1} et $+2^{n-1}-1$.

Des opérations sur les entiers

- l'opposé (opération unaire, notée $-$) ;
- l'addition (opération binaire, notée $+$) ;
- la soustraction (opération binaire, notée $-$) ;
- la multiplication (opération binaire, notée $*$) ;
- la division entière (opération binaire, notée $/$) ;
- le reste de la division entière (opération binaire, notée $\%$) ;

Attention la multiplication n'est pas implicite, le symbole $*$ doit toujours être indiqué explicitement entre les deux opérandes.

Des opérations sur les entiers (2)

Exemple

opération	résultat
17 + 5	22
17 - 5	12
17 * 5	85
17 / 5	3
17 % 5	2

Les réels

La représentation des réels varie suivant les langages de programmation, les machines et les normes utilisées.
Des opérations possibles sur les réels sont :

- l'opposé (opération unaire, notée $-$) ;
- l'addition (opération binaire, notée $+$) ;
- la soustraction (opération binaire, notée $-$) ;
- la multiplication (opération binaire, notée $*$) ;
- la division (opération binaire, notée $/$) ;

UNIVERSITÉ DE ROUEN

Des opérations sur les réels

Exemple

opération	résultat
`13.6 + 6.9`	$20,5$
`13.6 - 6.9`	$6,6999999999999993$
`13.6 * 6.9`	$93,840000000000003$
`13.6 / 6.9`	$1,9710144927536231$

L'affectation

Permet de donner une nouvelle valeur à une variable.

Syntaxe

nomDeVariable = expression

Sémantique

Calcul (ou évaluation) de la valeur de l'expression et rangement de cette valeur dans la case mémoire associée à cette variable.

Exemple

opération	instruction	valeur de la variable
affecter la valeur 1 à la variable x	x = 1	x : 1
affecter la valeur 3 à la variable y	y = 3	y : 3

UNIVERSITÉ DE ROUEN

L'affectation (2)

- le symbole d'affectation est =
- ce qui figure à gauche est obligatoirement un identificateur de variable ;
- la partie droite est une **expression**.

Les expressions

Une expression peut être :

- une valeur constante (exemples : `2`, `56.7` ou `'u'`) ;
- une variable ;
- toutes combinaisons d'opérations valides mettant en œuvre des constantes et/ou des variables.

Ordre de priorité

PEMDAS

P : parenthèses ;

E : exposant ;

M et D : multiplication et division ;

A et S : addition et soustraction.

À priorité égale les opérations sont évaluées de la gauche vers la droite.

Exemple

opérations	valeurs
5 + 4 * 2	13
(5 + 4) * 2	18

La manipulation des variables (2)

On peut modifier la valeur des variables tout au long du programme.

Exemple

opération	instruction	valeur
affecter la valeur `x + 1` à la variable `x`	`x = x + 1`	`x` : 2
affecter la valeur `y + x` à la variable `y`	`y = y + x`	`y` : 5

La saisie (ou lecture)

Permet d'affecter à une variable une valeur tapée sur le clavier.

Syntaxe

```
nomDeVariable = input(message)
nomDeVariable = raw_input(message)
```

L'affichage (ou écriture)

Permet d'écrire une valeur sur l'écran.
Cette valeur peut être le contenu d'une variable comme le résultat du calcul d'une expression.

Syntaxe

```
print expression
print expression,
```

L'exécution de l'instruction `print` consiste à placer le symbole de fin de ligne sur le flux de sortie (ce qui provoque un passage à la ligne lorsque le flux de sortie est l'écran).
L'utilisation de la virgule permet de ne pas faire passer le curseur à la ligne après son exécution.

UNIVERSITÉ DE ROUEN

Un exemple de programme Python

Première version

```
unEntier = input("Entrez un entier")
sonCarre = unEntier * unEntier
sonCube = sonCarre * unEntier
print "Le cube de", unEntier, "est", sonCube
```

Deuxième version

```
unEntier = input("Entrez un entier")
print "Le cube de", unEntier, "est",unEntier * unEntier * unEntier
```

UNIVERSITÉ DE ROUEN

Les commentaires

- annotez les programmes de commentaires ;
- tous les langages de programmation permettent de placer du texte dans un programme sans qu'il agisse sur l'exécution ;
- servent à faciliter la lecture du programme ;
- tout ce qui suit le symbole #.

Exemple

```
v = 4/3*pi*r*r*r  # Calcul du volume d'une sphère de rayon r
```

Les fonctions

En Python, une fonction est un objet qui doit être déclaré.

Syntaxe

```
def nomDeLaFonction(liste de paramètres formels):
    bloc d'instructions
```

Retour de résultat

La fonction peut contenir une instruction de la forme : `return` expression. Cette instruction permet de fournir le résultat de la fonction au programme appelant.

UNIVERSITÉ DE ROUEN

L'indentation

- l'**indentation** consiste à espacer les lignes de code par rapport au bord gauche de la fenêtre de saisie de texte ;
- cette indentation est obligatoire en Python ;
- la taille de l'espacement doit être proportionnelle au niveau d'imbrication des instructions du programme ;
- la plupart des éditeurs de texte offrent des facilités pour réaliser une bonne indentation.

UNIVERSITÉ DE ROUEN

Un exemple de fonction

Exemple

```
def f(x, n) :
    xcarre = x*x
    return 3*xcarre + 4*x/n + 5*n
```

Appel d'une fonction

Pour calculer le résultat d'une fonction pour certaines valeurs, le programme appelant doit appeler la fonction en lui transmettant ces valeurs.

La syntaxe est nomDeLaFonction(liste des paramètres réels).

La liste des paramètres réels est constituée d'une liste d'expressions séparées par des virgules. Il est important de respecter l'ordre des paramètres formels. La valeur de chaque expression est calculée et devient ainsi la valeur du paramètre formel correspondant.

UNIVERSITÉ DE ROUEN

Exemples d'appel

Exemple

```
a = f(b, 4)
a = f(14.5, n) + 4*f(b, m + 12)
print 'La valeur est : ', f(55.12, 42)
```

Fonctions prédéfinies

Il existe une bibliothèques de fonctions prédéfinies appelables dans n'importe quel programme.

Exemple

abs(x) retourne la valeur absolue de x
round(x) retourne l'entier le plus proche de x

L'instruction alternative

- souvent utile d'effectuer un choix en fonction du résultat d'un test ou d'une condition ;
- possible dans tous les langages de programmation d'effectuer un groupe (ou **bloc**) d'instructions en fonction du résultat d'un test ;
- l'instruction alternative (ou conditionnelle) prend généralement trois formes.

UNIVERSITÉ DE ROUEN

L'instruction alternative (2)

Syntaxe

```
if condition :
    bloc d'instructions 1
```

```
if condition :
    bloc d'instructions 1
else :
    bloc d'instructions 2
```

L'instruction alternative (3)

Syntaxe

```
if condition :
    bloc d'instructions 1
elif condition :
    bloc d'instructions 2
else :
    bloc d'instructions 3
```

Les conditions

- La condition est en fait une expression de type booléen.
- Cette expression est évaluée, si sa valeur est vraie alors le bloc d'instructions 1 est exécuté.
- Si sa valeur est faux, le bloc d'instructions 2 est exécuté lorsqu'il est présent.
- La condition résulte dans la majorité des cas d'une ou plusieurs comparaisons.

UNIVERSITÉ DE ROUEN

Les comparaisons

symbole Python	symbole mathématique
<	$<$
<=	$\leq$
==	$=$
!=	$\neq$
>=	$\geq$
>	$>$

Les blocs d'instructions

Un bloc d'instructions est :

- soit une seule instruction ;
- soit plusieurs instructions au même niveau d'indentation.

Exemple d'instruction alternative

Exemple

Calcul du maximum entre deux variables `x` et `y`. L'instruction suivante permet de stocker $\max\{\texttt{x}, \texttt{y}\}$ dans `maximum`.

```
if x > y :
    maximum = x
else :
    maximum = y
```

Exemple d'instruction alternative (2)

Exemple

Calcul du maximum et du minimum de `x` et `y`

```
if x > y :
    maximum = x
    minimum = y
else :
    maximum = y
    minimum = x
```

Tests imbriqués

Il est bien sur possible d'imbriquer des instructions `if`.

Exemple

```
if x >= y :
   if x == y :
      print x,"=",y
else
   print x, "<",y
```

L'instruction itérative

L'instruction itérative permet de répéter un certain nombre de fois l'exécution d'une suite d'instructions sous une certaine condition. De façon imagée, on appelle **boucle** cette méthode permettant de répéter l'exécution d'un groupe d'instructions.

Syntaxe

```
while condition :
    bloc d'instructions
```

L'instruction itérative (2)

Exemple

Affichage des dix premiers entiers strictement positifs

```
print 1
print 2
print 3
print 4
print 5
print 6
print 7
print 8
print 9
print 10
```

L'instruction itérative (3)

Exemple

```
nombreDEntiers = input("Entrez le nombre d'entiers : ")
compteur = 1
while compteur <= nombreDEntiers :
    print compteur
    compteur = compteur + 1
```

L'instruction itérative (4)

Si la valeur de la condition est fausse dès le départ alors le bloc d'instructions ne sera jamais exécuté.

Exemple

```
compteur = 1
while compteur < 0 :
    print compteur
    compteur = compteur + 1
```

Boucle infinie

Par contre si la valeur de la condition est vraie et que le bloc d'instructions ne permet pas d'altérer cette valeur alors le bloc d'instructions sera exécuté à l'infini : on a alors affaire à une boucle infinie.

Exemple

```
compteur = 1
while compteur != compteur + 1 :
    print compteur
    compteur = compteur + 1
```

Accumulateur

Une technique classique consiste, on vient de le voir, à utiliser un compteur. Une autre technique classique consiste à utiliser un accumulateur. Prenons l'exemple d'une fonction qui calcule la somme des n premiers entiers positifs.

Exemple

```
def Somme(n) :
    compteur = 0
    accumulateur = 0
    # accumulateur = somme de 0 à compteur et compteur = 0
    while compteur < n :
        # accumulateur = somme de 0 à compteur et compteur < n
        compteur = compteur + 1
        # accumulateur = somme de 0 à compteur-1 et compteur ≤ n
        accumulateur = accumulateur + compteur
        # accumulateur = somme de 0 à compteur et compteur ≤ n
    # accumulateur = somme de 0 à compteur et compteur = n
    return accumulateur
```

UNIVERSITÉ DE ROUEN

Les listes

Les listes sont des collections ordonnées d'éléments séparés par des virgules, l'ensemble étant délimité par des crochets.

Exemple

```
liste = ['lundi', 1, 'mercredi', '3.0']
```

Les listes (2)

On peut accéder individuellement à chaque élément d'une liste.

Exemple

```
jour = liste[0]
```

On peut connaître le nombre d'éléments d'une liste.

Exemple

```
print len(liste)
```

Les listes (3)

On peut supprimer un élément d'une liste.

Exemple

```
del(liste[2])
```

On peut ajouter un élément à la fin d'une liste.

Exemple

```
liste.append('vendredi')
```

Les chaînes de caractères

Les chaînes de caractères sont des suites de caractères délimitées par des apostrophes (’), des quotes (") ou des triple quotes (""").

Exemple

```
a = 'ok'
b = '"Oui"'
c = "j'aime bien"
d = """
sur deux
lignes"""
```

Les chaînes de caractères (2)

On peut concevoir les chaînes de caractères comme des liste de caractères : il est possible d'accéder à chaque lettre du mot en donnant sa position dans le mot en commençant la numérotation à 0.

Exemple

```
>>> a = 'bioinfo'
>>> print a[0]
b
>>> print a[3 :6]
inf
>>> print a[ :3]
bio
>>> print a[3 :]
info
```

Les chaînes de caractères (3)

Les chaînes de caractères peuvent être concaténées avec l'opérateur + et répétées avec l'opérateur *.

Exemple

```
a = 'abc' + 'cde'
b = a * 3
```

L'instruction `for`

L'instruction `for` a `in` s`:` attribue successivement à a les valeurs des éléments de s.
s peut être une liste ou une chaîne de caractères.

Exemple

```
for i in liste :
   print i
```

La fonction range

La fonction `range(`n`)` renvoie une liste composées des entiers de 0 à $n-1$.

Exemple

```
for i in range(i) :
   print i
```

Les fichiers

L'utilisation de fichiers permet de lire et écrire des informations sur un support de mémoire secondaire.

Les fichiers texte

Ouverture

fichier = open('MonFichier', 'm')

où m est

- a : pour ajout ;
- w : pour création ;
- r : pour lecture.

Les fichiers texte (2)

La lecture

- t = fichier.readline() lit une ligne dans le fichier et la transfère dans une chaîne de caractères.
- t = fichier.readlines() lit toutes les lignes restantes dans le fichier et les transfère dans une liste de chaîne de caractères.

L'écriture

fichier.write(txt) écrit txt dans le fichier.

La fermeture

fichier.close()

UNIVERSITÉ DE ROUEN