UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Python

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Rappels - Un programme ?

- Ensemble d'opérations exécutées par un ordinateur
- Code source ou binaire
- Code source interprété ou compilé

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Rappels - Interprétation

- Code source interprété par un programme appelé **interpréteur**.
- L'interpréteur lit et analyse chaque opération écrite dans le code source,
- Évalue ou exécute cette opération.
- Avantage : cycle de développement rapide et débogage simple
- Points faibles : optimisation et temps d'exécution

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Rappels - compilation

- Code source interprété par un programme appelé **compilateur**.
- Le compilateur traduit le code source vers un autre langage, en général, plus bas niveau.
- Avantage : optimisation et temps d'exécution
- Points faibles : cycle de développement plus lent et débogage plus complexe

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Variable

- Stockage en mémoire d'une valeur qui peut changer et manipulée grâce à un nom symbolique
- Une variable a un nom et un type.
- Le type définit les valeurs possibles : un nombre, une chaîne de caractères, un booléen…
  - Dans certains langages dits **statiquement typés** le type est attaché à la variable et est donc définitif.
  - Dans d'autres dits **dynamiquement typés**, le type est seulement attaché aux valeurs et donc le type d'une variable peut changer.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Structure de données

- Type qui regroupe plusieurs variables.
- Selon les langages, certains types de structures de données existent nativement : les listes, les vecteurs, les dictionnaires…
- On peut aussi créer ses propres structures de données :
  - Par exemple, une structure représentant un étudiant comprendra son prénom, son nom, son numéro d'étudiant, son adresse email, son UFR, …
  - En créant ainsi un type *étudiant* on peut utiliser une seule variable de ce type là où on a besoin de toutes ces valeurs.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Python

- Créé en 1991 par Guido van Rossum.
- Très répandu, il existe énormément de bibliothèques.
- Dynamiquement typé.
- Plutôt prévu pour la programmation impérative.
- Offre un support de la programmation orientée objet (à base de classes).

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Syntaxe et indentation

- En Python, les indentations et retours à la lignes sont significatifs.
- Une instruction se termine par un retour à la ligne
- Un bloc est déterminé par un niveau d'indentation
- Le symbole **#** indique le début d'un commentaire qui se termine à la fin de la ligne.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Les variables

- Stockage en mémoire manipulé grâce à un nom symbolique
- En Python, tout accès à la mémoire se fait par l'intermédiaire de variables (pas d'adresses, ni pointeurs)
- Un nom de variable doit commencer par une lettre ou par le caractère souligné _ suivi d'un nombre quelconque de lettres, chiffres ou caractères soulignés.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Les variables

- Création de la variable au moment de sa première utilisation qui est toujours une *initialisation* réalisée grâce à l'*opérateur d'affectation* **=**

```
a = 5
```

- Cette affectation crée une variable *a* et lui affecte la valeur entière 5.

- Pour afficher le contenu de *a :*

```
print (a)
```

- On peut utiliser *a* dans une expression

```
b = 5  + a
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Type des variables

- Notion très importante en programmation : aux valeurs contenues en mémoire sont associés des types.
- Permet de savoir ce qu'on peut faire avec les valeurs et comment les manipuler.
- En Python, pas de déclaration de type lors de la création d'une variable.
- Les variables ne sont donc pas associées à un type mais les valeurs qu'elles contiennent le sont !
- C'est au programmeur de connaître le type de la valeur contenue dans une variable pour ne pas faire d'opération incompatible

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Types

- Les valeurs numériques peuvent être entières ou flottantes
- Une valeur booléenne vaut True ou False
- Il existe une valeur None pour *rien*
- Les chaines de caractères sont notées entre simples ou doubles quotes.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Les listes

- Une liste est une collection ordonnée et modifiable d'éléments éventuellement hétérogènes.
- Les éléments séparés par des virgules, et entourés de crochets.
- Pour indexer une liste, on utilise l'opérateur [ ] dans lequel l'index, un entier signé qui commence à 0 indique la position d'un élément :

```
couleurs = ['trèfle', 'carreau', 'coeur', 'pique']
print(couleurs)      # ['trèfle', 'carreau', 'coeur', 'pique']
couleurs[1] = 14
print(couleurs)      # ['trèfle', 14, 'coeur', 'pique']
list2 = [4, 2.718]
print (list2[1])     #2.178
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Opérateurs arithmétiques

- *+* : addition

```
b = 5 + a #affecte à b la valeur 5 + a
```

- *-* : soustraction
- *** : multiplication
- */* : division

```
b = 5 / 2 #affecte 2 à b car ici la division est entière car les deux opérandes le sont.
b = 5.0 / 2 #affecte 2.5 à b. Division décimale.
```

- *%* : modulo (reste de la division entière)
- *+=* : ajout d'une valeur à une variable

```
b += 5 #on ajoute 5 à la valeur contenue dans b
```

- *-=* : retrait d'une valeur à une variable

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Opérateurs logiques

- == : Teste l'égalité des opérandes
- != : Teste l'inégalité
- > : strictement supérieur
- >= : supérieur ou égal
- < : strictement inférieur
- <= : inférieur ou égal

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Les structures de contrôle

```
if expression:                # ne pas oublier les :
    bloc d'instructions1      # attention à l'indentation
else:                         # else au même niveau que if
    bloc d'instructions2
# suite du programme
```

- Si l'expression est vraie alors le bloc d'instructions 1 est exécuté.
- Si l'expression est fausse alors c'est le bloc d'instructions 2 qui est exécuté.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Les structures de contrôle

```
if expression1:
    bloc d'instructions1
elif expression2:
    bloc d'instructions2
elif expression3: # autant de elif que nécessaire
    bloc d'instructions3
else:  #optionnel
    bloc d'instructions4
# suite du programme
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Boucle *while*

```
while expression:
    bloc d'instructions
# suite du programme
```

- Si l'expression est vraie le bloc d'instructions est exécuté
- L'expression est à nouveau évaluée.
- Le cycle continue jusqu'à ce que l'expression soit fausse : on passe alors à la suite du programme.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Boucle *for*

```
for élément in séquence:
    bloc d'instructions
# suite du programme
```

- Les éléments de la séquence sont issus d'une chaîne de caractères ou bien d'une liste.

- Exemple avec une chaîne de caractères :

```
chaine = 'Salut'
for l in chaine:      # l est la variable d'itération
    print(l)
print("Fin de la boucle")
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Fonctions

- Sorte de sous-programme qui permet de réutiliser le même code à différents endroits.
- Une fonction reçoit un ou des arguments (ou paramètres) et renvoie un résultat.
- On utliise le mot-clé réservé *def* pour créer une fonction.
- On indique entre parenthèses les arguments avec lesquels la fonction va travailler. Ces arguments peuvent avoir une valeur par défaut (dans ce cas le paramètre est suivi d'un = et de la valeur à lui attribuer en cas d'absence au moment de l'appel)
- Le mot-clé *return* (optionnel) permet le renvoi d'une valeur au programme appelant. L'utilisation est facultative.

```
def nom_de_la_fonction (argument1, argument2 = valeur_par_défaut, …) :
    instruction1
    instruction2
    …
    return valeur
# suite du programme
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Portée des variables

- On appelle **portée** d'une variable, la partie du code dans laquelle notre variable est accessible.
- Une variable déclarée en dehors d'une fonction est une variable **globale** (accessible partout).
- Une variable déclarée dans une fonction est dite **locale**, elle n'existe que dans cette fonction.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Portée des variables

```
a=4 #variable globale
d=3 #variable globale
def exemple (arg1):
    b=2 #variable locale
    r= b*arg1 + d  #d est accessible, r variable locale
    a=10 # ⚠ l'affectation « a=10 » n'est pas une
    #modification en place mais bien la création d'une
    #nouvelle variable locale
    return r
#suite du programme
#b et r ne sont plus accessibles
#a vaut toujours 4
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Passage d'arguments

- En Python, les arguments des fonctions sont passés par référence.

- Les variables passées en arguments sont donc affectés par les modifications en place opérées dans la fonction.

```
def exemple (l1, l2):
    l1[1]=2 #modification en place
    l2=[13, 35, 34] # nouvelle variable locale
a=[4, 3, 6]
b=[3, 4, 5]
print (a) #[4, 3, 6]
print (b) #[3, 4, 5]
exemple (a, b)
print (a) #[4, 2, 6]
print (b) #[3, 4, 5]
```