Chapitre 1: Types et Instructions de Base

M. DIALLO

Spécialité: L2 GL - IAGE

muustafa.dllo@gmail.com
+221 77 193 52 60

Introduction au Typage Dynamique

- En Python, **tout est un objet**
- Un objet contient des données et des méthodes pour les manipuler
- Les objets ont un type qui détermine leur comportement

Nommer des objets (variables) :

- Les noms peuvent contenir :
  - Lettres (a-z, A-Z)
  - Chiffres (0-9)
  - Caractère souligné (_)

Exemples valides

`age = 20`, `nom_complet = "Jean"`, `PI = 3.14159`

Typage Dynamique

Caractéristique importante

En Python, le type n'est pas lié à la variable mais à l'objet référencé.

Exemple

```
1 x = 10        # x est de type int
2 x = "hello"   # maintenant x est de type str
3 x = 3.14      # maintenant x est de type float
4
```

- Pour supprimer une variable : `del nom_variable`
- **Garbage Collector** : Mécanisme qui libère automatiquement la mémoire des objets non référencés

Types Numériques

Python supporte plusieurs types numériques :

- **int** : Entiers (précision illimitée)
- **float** : Nombres décimaux
- **complex** : Nombres complexes
- **bool** : Booléens (True/False)

Exemples

```
1 entier = 42
2 decimal = 3.14159
3 complexe = 2 + 3j
4 booleen = True
5
```

- `False = 0`, `True = 1` (héritage des entiers)
- Nombres complexes utilisent `j` comme constante imaginaire

Conversions entre Types Numériques

Opérations de conversion

- `float(10)` → 10.0 (entier vers flottant)
- `int(4.9)` → 4 (flottant vers entier, troncature)
- `str(42)` → "42" (nombre vers chaîne)
- `bool(0)` → False, `bool(1)` → True

Attention à la perte d'information

La conversion `int(4.9)` tronque la partie décimale, elle ne l'arrondit pas.

Conversion sécurisée

```
1 age = "25"
2 age_int = int(age)  # Conversion reussie
3
4 age = "25.5"
5 age_float = float(age)  # Conversion reussie
6
```

Chaînes de Caractères (str)

Pour créer une chaîne de caractères :

- Guillemets simples : `'Bonjour'`
- Guillemets doubles : `"Bonjour"`
- Guillemets triples pour multiligne : `"""Ligne1\nLigne2"""`

Immuabilité

Les chaînes sont immuables : chaque modification crée un nouvel objet.

Exemples

```
1 nom = "Alice"
2 message = 'Bonjour ' + nom
3 chemin = "C:\\dossier\\fichier.txt"  # Echappement
4 chemin_brut = r"C:\dossier\fichier.txt"  # Chaine brute
5
```

Méthodes Courantes sur les Chaînes

Quelques méthodes utiles

- `str.title()` : Première lettre de chaque mot en majuscule
- `str.replace(old, new)` : Remplace une sous-chaîne
- `str.isdecimal()` : Vérifie si la chaîne est un nombre décimal
- `str.format()` : Formatage avancé de chaînes

Exemples d'utilisation

```
1  "je suis ingenieur".title()
2  # Resultat: 'Je Suis Ingenieur'
3
4  "ingenieur logiciel".replace("logiciel", "civil")
5  # Resultat: 'ingenieur civil'
6
7  "123".isdecimal()  # True
8  "12.3".isdecimal() # False
9
10 "Bonjour {}, vous avez {} ans".format("Alice", 25)
11
```

Formatage Moderne des Chaînes (f-strings)

Nouveauté Python 3.6+

Les f-strings permettent un formatage plus lisible et concis.

Comparaison

```
# Ancienne methode
nom, age = "Alice", 25
message = "Bonjour {}, vous avez {} ans".format(nom, age)

# Avec f-string (plus simple!)
message = f"Bonjour {nom}, vous avez {age} ans"

# Formatage numerique
pi = 3.14159
message = f"        {pi:.2f}"  # Resultat:         3.14

```

Recommandation

Utilisez les f-strings pour un code plus clair et maintenable !

Les Séquences

Définition

Une séquence est un ensemble ordonné d'éléments indexés de 0 à n-1.

Types de séquences en Python :

- **list** : Listes (mutables)
- **tuple** : Tuples (immuables)
- **str** : Chaînes de caractères (immuables)
- **range** : Intervalles de nombres

Opérations communes sur les séquences

```
1  # Acces a un element
2  sequence = [10, 20, 30, 40]
3  print(sequence[0])    # Premier element: 10
4
5  # Test d'appartenance
6  print(20 in sequence)  # True
7
8  # Longueur
9  print(len(sequence))  # 4
10
```

Listes

Caractéristiques

Les listes sont des séquences **mutables** (modifiables après création).

Création et manipulation

```
# Creation
fruits = ["pomme", "banane", "orange"]
nombres = list(range(5))  # [0, 1, 2, 3, 4]

# Modification
fruits[0] = "kiwi"  # Remplacement
fruits.append("mangue")  # Ajout en fin
fruits.insert(1, "fraise")  # Insertion a position

# Suppression
del fruits[0]  # Supprime le premier element
fruits.remove("banane")  # Supprime par valeur
element = fruits.pop()  # Supprime et retourne le dernier

```

Méthodes Courantes des Listes

Quelques méthodes essentielles

- `list.append(x)` : Ajoute x à la fin
- `list.extend(iterable)` : Ajoute plusieurs éléments
- `list.insert(i, x)` : Insère x à la position i
- `list.remove(x)` : Supprime la première occurrence de x
- `list.pop([i])` : Supprime et retourne l'élément à la position i
- `list.sort()` : Trie la liste (en place)
- `list.reverse()` : Inverse l'ordre (en place)

Conversion Chaîne - Liste

Transformer une chaîne en liste

Utilisez la méthode `split()` pour diviser une chaîne.

Exemples

```
# De chaine a liste
texte = "apprendre python est amusant"
mots = texte.split()  # Division par espaces

# Division par un separateur specifique
donnees = "nom,age,ville"
elements = donnees.split(',')

# De liste a chaine
nouvelle_phrase = " ".join(mots)

# Avec un separateur different
chemin = "/".join(['dossier', 'sous-dossier', 'fichier'])

```

Introduction aux Fonctions

Définition

Une fonction est un bloc de code réutilisable qui effectue une tâche spécifique.

Syntaxe de base

```
def nom_de_la_fonction(parametres):
    """Documentation de la fonction (docstring)"""
    # Corps de la fonction
    instructions
    return resultat  # Optionnel

```

Exemple simple

```
def saluer(nom):
    """Affiche un message de salutation"""
    message = f"Bonjour {nom}!"
    print(message)
    return message

```

Fonctions Intégrées (Built-in)

Quelques fonctions intégrées essentielles

Python fournit de nombreuses fonctions intégrées :

- `print()` : Affiche des valeurs
- `len()` : Retourne la longueur d'un objet
- `type()` : Retourne le type d'un objet
- `input()` : Lit une entrée utilisateur
- `range()` : Génère une séquence de nombres
- `min()`, `max()`, `sum()` : Opérations sur séquences

Exemples d'utilisation

```
1 print("Hello World")  # Affichage
2 longueur = len([1, 2, 3])  # Longueur: 3
3
4 nombres = range(5)  # 0, 1, 2, 3, 4
5
6 maximum = max([10, 20, 5])  # 20
7 somme = sum([1, 2, 3])  # 6
8
```

Introduction aux Modules

Définition

Un module est un fichier Python contenant des fonctions, classes ou variables réutilisables.

Importer un module

```
# Import complet
import math

# Import selectif
from math import sqrt, pi


```

Bonnes pratiques

Préférez les imports explicites plutôt que `import *` pour plus de clarté.

Utilisation des Modules

Accéder aux éléments d'un module

```python
# Apres: import math
racine = math.sqrt(25)        # 5.0
valeur_pi = math.pi           # 3.141592653589793

# Apres: import numpy as np
tableau = np.array([1, 2, 3])

# Apres: from math import sqrt, pi
racine = sqrt(25)             # Plus besoin de math.
valeur_pi = pi

```

Modules utiles pour débutants

- `math` : Fonctions mathématiques
- `random` : Génération de nombres aléatoires
- `datetime` : Manipulation de dates et heures
- `os` : Interaction avec le système d'exploitation

Récapitulatif

Ce que nous avons vu

- Typage dynamique de Python
- Types numériques (int, float, complex, bool)
- Chaînes de caractères et leurs méthodes
- Les séquences (listes, tuples, chaînes)
- Manipulation des listes
- Bases des fonctions
- Import et utilisation des modules

À retenir

- Python utilise le typage dynamique
- Les chaînes sont immuables, les listes sont mutables
- Pratiquez régulièrement pour maîtriser ces concepts !

Prêt pour la séance de travaux pratiques ?