**Initiez-vous à Python pour l’analyse de données**

**Objectifs :**

* Mettre en place votre environnement de travail Python
* Manipuler les fonctions et objets fondamentaux
* Structurer votre code au sein d’un projet
* Utiliser des librairies Python spécialisées

**Variables = Nom + valeur**

Nom Variable clair (ex : compteCourant) explicite et avec convention camel case.

Plusieurs types : Variable entières (int), décimales (float), ou chaînes de caractères (string). Type de variable défini par sa valeur.

Type primitif : ?

Variable de type numérique : entier relatifs et nombres décimaux.

Attribution d’une valeur s’appelle une affectation.

**Opérations : + - \* /**

% modulo

\*\* puissance

// division entière (quotient)

*compteEpargne = compteEpargne + 100 ⬄ compteEpargne += 100*

**Fonctions**

print()

int() str() float() (fonctions de cast)

len()

type()

pow(a, b)

abs()

help()

**Créer une fonction**

Def nomFonctions() :

**POO :**

Classe : catégorie d’objet

Attributs : caractéristiques d’une classe

Méthodes : fonctionnalités d’une classe

Possibilité de créer plusieurs objets différents d’une même classe en changeant les attributs, ce sont les instances de classe.

**Utilisation d’une méthode :**

a = «HELLO WORLD ! »

a.lower() => utilisation : nomVariable.méthode()

**Méthode de chaînes de caractères**

upper() : tout mettre en majuscule

capitalize() : première lettre en maj + suite en min

replace(old, new) : remplace old par new

find(chaîne): retourne soit indice de première occurrence, soit -1 si elle ne le trouve pas .

**Liste**

Classe capable de stocker plusieurs informations.

Liste = objet ordonné, cad que chaque élément est associé à un indice.

Premier indice = 0

Ex : nomClient = [‘Georges Dupont’, ‘Luc Martin’, ‘Lucas Anderson’, ‘Alexandre Petit’]

Changer une valeur : nomClient[0] = ‘Georges Dupond’

Afficher une valeur : print(nomClient[0])

Afficher dernier élément : print(nomClient[-1])

Afficher éléments du 2ème au 4ème : print(nomClient[1 :3])

Afficher éléments du début au 3ème : print(nomClient[ :2])

Liste pouvant rassembler plusieurs objets différents (entiers, décimaux, chaînes de caractères, et même une autre liste). Possibilité de créer une liste vide.

Maliste = []

Méthodes de liste :

liste = []

liste.append(7) # ajouter un élément à la fin de la liste

liste.insert(1, 12) # insérer l’entier 12 à l’indice 1

liste.extend([1, 2, 3]) # ajouter une liste à la fin de la liste

liste.remove(12) # supprimer un élément de la liste

liste.index(2) # affiche le numéro de l’indice de l’élément 2

del liste[3] # supprimer l’élément d’indice 3

Fonction len :

liste = [1, 2, 3]

len(liste) # affiche le nombre d’élément de la liste

**Dictionnaire**

Liste d’éléments organisés via un système de clés.

Éléments non ordonnés contrairement à liste

Chaque clé a une valeur et doit être unique

Ex : comptes = {‘Georges Dupont’ : 10000, ‘Luc Martin’ : 150, ‘Lucas Anderson’ : 300, ‘Alexandre Petit : 1800.74’}

Manipulation des éléments :

comptes[‘Georges Dupont’] -= 2000 # soustrais 2000 au compte de Georges

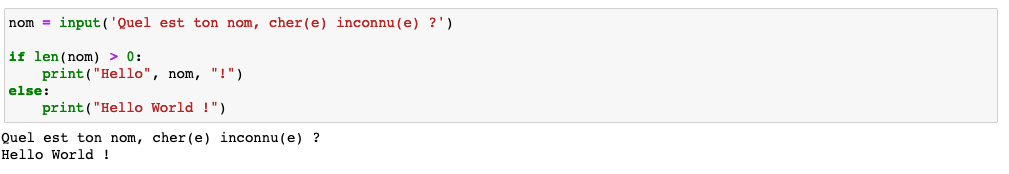
comptes[‘Cyril Andreje’] = 1000 # ajoute un nouvel éléments au dictionnaire

print(comptes[‘Cyril Andreje’]) # affiche valeur du compte de Cyril Andreje

comptes.pop(‘Luc Martin’) # supprime Luc Martin du dictionnaire

Fonction len : len(comptes) # affiche nombre d’éléments du dictionnaire

**Structures conditionnelles**



**Si plusieurs conditions**

****

**Variables de types booléen**

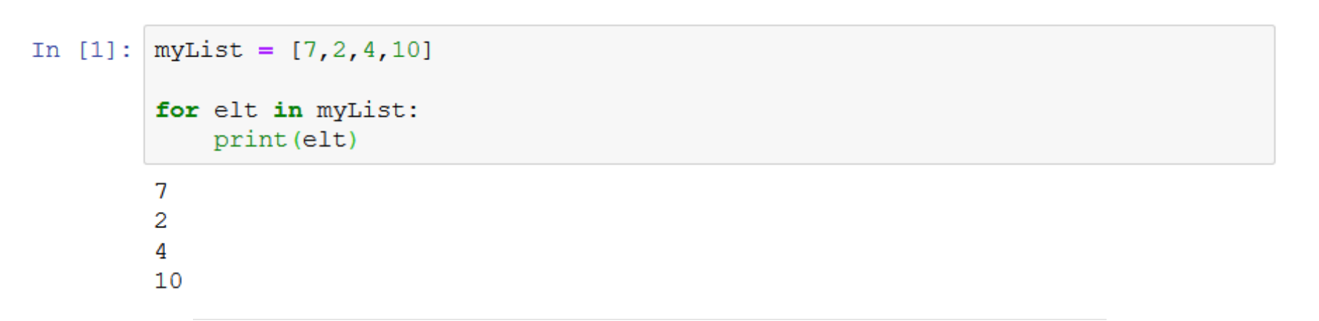
Deux valeurs possibles : True ou False

Utilisation :

* Dans une condition if… else :
  + If True:
* Avec une variable de type booléen :
  + maVariable = True
  + if maVariable :
* Une expression aboutissant à une valeur booléenne;
  + Meteo = « La météo est chouette ! »
  + Meteo.startswith(« La météo ») #True
* Opérateurs de comparaison :
  + ==
  + !=
  + <
  + <=
  + >
  + >=
* Opérateurs logiques :
  + and
  + or
  + not
* Opérateur « in »

**Boucles**

**For** : boucler avec un nombre de fois fixé



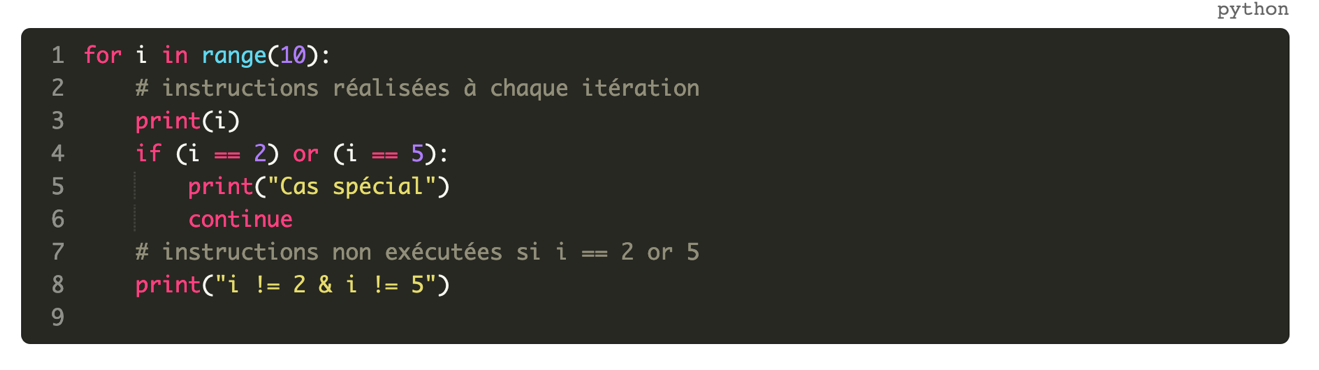




While : Répéter la boucle « tant que » c’est possible



Passer certaines instructions au sein de la boucle avec « continue »



Interrompre la boucle, lorsqu’on cherche un élément particulier par exemple, avec « break »

