Analyse de données

Python

Cours 2 par Nizar Ouarti
(2022-2023)

Plan du cours

Les différentes parties détaillées tout au long du cours sont les suivantes:
Aucun prérequis n'est attendu pour ce cours

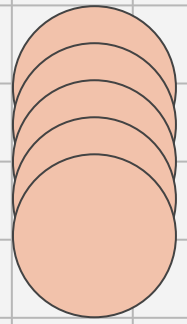
Dictionnaires	Description des dictionnaires
Les Tuples	Description des tuples
Librairies : import	Import, Numpy, Panda

Plus d'informations:
nizar.ouarti@sorbonne.fr

Page internet:
F page nizar

Qu’est ce qu’un

Dictionnaire



Un dictionnaire en Python est un type composé permettant d’**associer des clefs et des valeurs**:

- Contrairement aux liste ou aux tuples l’accès ne se fait pas par un indice
- L’accès d’une valeur se fait fait grâce à une **clef**
- Utilisation de **l’accolade pour créer un dictionnaire**

Exemple de manipulation de dictionnaire

- **Création itérative**
  >>> dico={}
  >>> dico[“triangle”]=3
  >>> dico[“carre”]=4

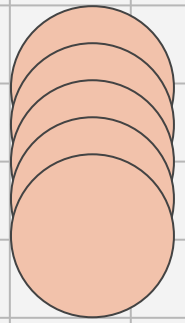
- **Création avec plusieurs entrées**
  >>> dico={“triangle”: 3, “carre”:4}
  Ici la clef et la valeur séparés par :
  Une virgule sépare les couples clef:valeur

- **Parcourir un dictionnaire:**
  >>> for clef, veleur in dico.items():
  >>> ... print (clef,valeur)
  dico.items() retourne une liste de couples (clef:valeur)

- **Accéder à un élément:**
  >>> print(dico[“carre”])

Qu’est ce qu’un

Tuple



Un tuple est un type composé, équivalent à une liste mais immutable:

- L’accès se fait par indice
- Contrairement aux liste on utilise les parenthèse non les crochets
- La fonction append n’est pas disponible car immutable
- Des éléments séparés par des virgules, sans parenthèses sont quand même des tuples

Exemple de manipulation de tuple

- **Initialisation rapide**
  >>> x,y,z = (3.2, 1.5, 0.7)
- **Echange de variables**
  >>> (b,c) = (c, b)
- **Récupération de variables**
  >>> a=(5,2)
  >>> u,v=à
  >>> u=5
  >>> v=2
- **Particularité élément unique:**
  >>> a= (3)
  Va créer un entier à initialisé à 3
  >>> a= (3,)
  Permet de créer une liste a

- **Dans les return de fonction:**
  >>> return a, b, c
  >>> return (a, b, c)
  Les deux notations équivalentes
  Cela permet de retourner plusieurs variables de sorti
- **Parcourir un tuple:**
  >>> for el in montuple :
  >>> ... print (el)

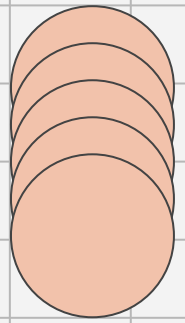
- **Accéder à un élément:**
  >>> print(montuple[0])

Import de library Python

- **From import as**
  - **from file import object as alias**
  - Avec file qui est le nom du fichier contenant la library
  - Avec object qui est l'élément à importer : object, fonction, variable, etc…
  - Avec alias qui est la manière d'en parler dans le code
- **import**
  - On peut aussi écrire **import file as alias**
  - Dans ce cas pour utiliser les objets on doit écrire alias.object
  - example : **import numpy as np**
  - np.array()
- **sys.path.append**
  - On peut importer ses propres library
  - On doit idiquer le chemin ou elle se trouve
  - **sys.path.append(chemindeMaLlibrary) et ensuite import malibrary as ml**

La library

Numpy



NumPy

La library Numpy agit sur les tableaux:

- Les tableaux peuvent être de taille et dimensions arbitraire
- Il est possible de représenter des matrice et des tenseurs
- Numpy est très rapide comparé à du Python natif
- Numpy peut avantageusement remplacer Matlab
- De nombreuses fonctions sont disponibles

Numpy

- **Import de Numpy classique**
  >>> import numpy as np
  >>>
- **Création de tableau vide**
  >>> tab= np.array((5,5))
- **Création de tableau rempli de zéros ou de uns**
  >>> tab= np.zeros((5,5))
  >>> tab= np.ones((5,5))
- **Création de tableau manuelle**
  >>> tab= np.array([1,2,3])
  >>> tab= np.array([[1,2,3],[4,5,6]])

- **Dimension d'un tableau**
  >>> tab.shape
  >>> (2,3)
  Un tableau de 2 ligne et 3 colonnes
  Les tableaux en numpy utilisent la convention du C en mémoire (row first): stockage par ligne

- **Accès aux éléments d'un tableau**
  >>> tab[1,2]
  >>> 6
  Les indices sont de 0 à N-1 (N taille de la dimension)

Numpy

- **Slicing**
  >>> tab[0:1,:]
  >>> array([[1, 2, 3]])
  On peut récupérer une sous partie du tableau
- **Utilisé comme une Matrice (transposé)**
  >>> tab.T
  >>> array([[1, 4],
  [2, 5],
  [3, 6]])
  >>>

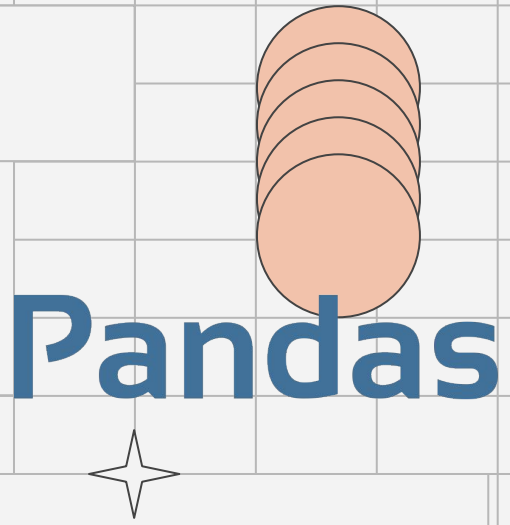
- **Utilisé comme une Matrice (multiplication)**
  >>> tab.dot(tab.T)
  >>> array([[14, 32],
  [32, 77]])

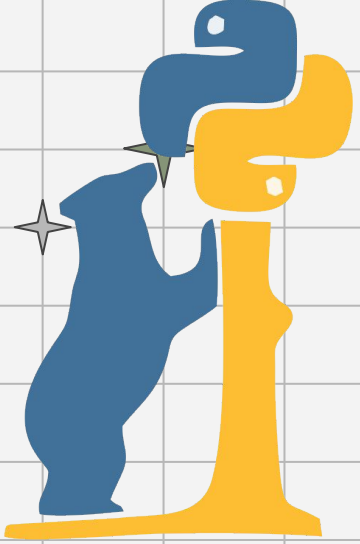
- **Cumsum, min, max, mean**
  >>> tab.cumsum()
  >>> tab.min()
  >>> tab.max()
  >>> tab.mean()

Bien d'autre fonctions sont disponibles

La library

Pandas





La library Pandas sert à l'analyse de données:

- Ouvrir et écrire dans les formats classiques de stockage de données
- csv, xlsx, json, sql, html, hdfs, gbq (Google BigQueries) et parquet
- Utilise la notion de DataFrame, pour traduire tous ces formats
- dataFrame= read_*format* (File) : création de dataFrame en partnat d'un fichier
- dataFrame.to_format (File) : écriture du dataFrame sur le disque
- Panda agit sur des Tables

Qu’est ce qu’une table?

- **Les tables sont des formes de données retrouvées dans les bases de données ou des logiciel avec feuille de calcul (spreadsheet)**
- **Citer des exemples**

State	Total Population
Uttar Pradesh	199,581,477
Maharashtra	112,372,972
Bihar	103,804,637
West Bengal	91,347,736
Andhra Pradesh	84,665,533
Madhya Pradesh	72,597,565

Pandas

- **Import de Pandas classique**
  >>> import pandas as pd
- **Import d'un fichier csv**
  >>> patients= pd.read_csv('in.csv')
- **Import d'un fichier parquet**
  >>> data2= pd.read_parquet('in.parquet', engine='pyarrow')
- **Création d'un fichier excel**
  >>> patients.to_excel('out.xlsx')
- **Création d'un fichier json**
  >>> data2.to_json('out.json')
- **Lister le nom des colonnes**
  >>>patients.keys()

- **Sélectionner certaines colonnes**
  >>> taille_age = patients[["taille", "age"]]
  >>> taille_age.head() #affichage du début de la table
  >>> taille_age.shape # donne la taille
  >>> (2,3)
- **Sélection lignes basé condition**
  >>>taille_age[taille_age["taille"]<1.65]
- **Sélection lignes basé condition en ne conservant que certaines colonnes**
  >>>taille_age.loc[taille_age["taille"]<1.65, "age"]
- **Sélection basé index**
  >>>taille_age.iloc[2:5,3:5]

On s’exerce!