Formation Analyse de données

-Python Pandas-

Manouba - Zaghouan

2023

Plan de la formation

- Introduction
- Installation et configuration
- Les structures de données Pandas
- La manipulation des données
- Le tri des données
- Statistiques
- Graphique

Introduction

NB: Cette formation est dédiée aux enseignants et elle ne doit pas être traitée avec les élèves.

Pandas est une bibliothèque dédiée Python *qui* permet la **manipulation** et **l'analyse** de données.

Installation et configuration

Première méthode : La plateforme Anaconda

Anaconda est une distribution Python, dédiée pour la science de données et elle comporte :

- **Python** (>=3.7)
- Les librairies de Data Science telles que : **Matplotlib**, **Scipy**, **Numpy**, **Pandas**
- Le notebook **Jupyter**
- D'autres utilitaires

Installation

- Lien de téléchargement d’anaconda :

  https://www.anaconda.com/download/

- Il existe aussi une version express d'Anaconda

  https://conda.io/miniconda.html

Installation

Deuxième Méthode : Installer uniquement le notebook Jupyter

- Installer Python
- Lancer l'invite de commandes (**cmd**)
- Mettre à jour la bibliothèque **pip**

  python -m pip install --upgrade pip
- Installer la bibliothèque **jupyter**

  python -m pip install jupyter

Pour vérifier si l'installation s'est bien déroulée, dans l'invite de commandes, lancer **jupyter** en tapant la commande suivante :

jupyter notebook

Installation

- ***Troisème Méthode : Installer les bibliothèques scipy, numpy, pandas et matplotlib.***

python -m pip install scipy

python -m pip install numpy

python -m pip install matplotlib

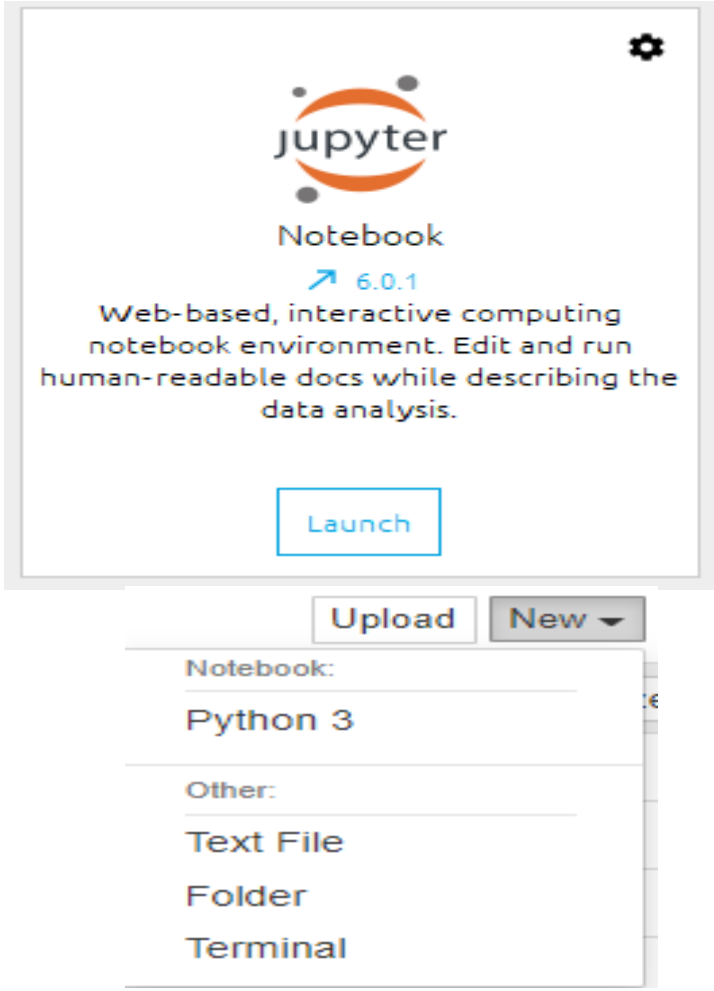
python -m pip install pandas

- ***Quatrième Méthode : Via l'éditeur EduPython/ IdlePython***

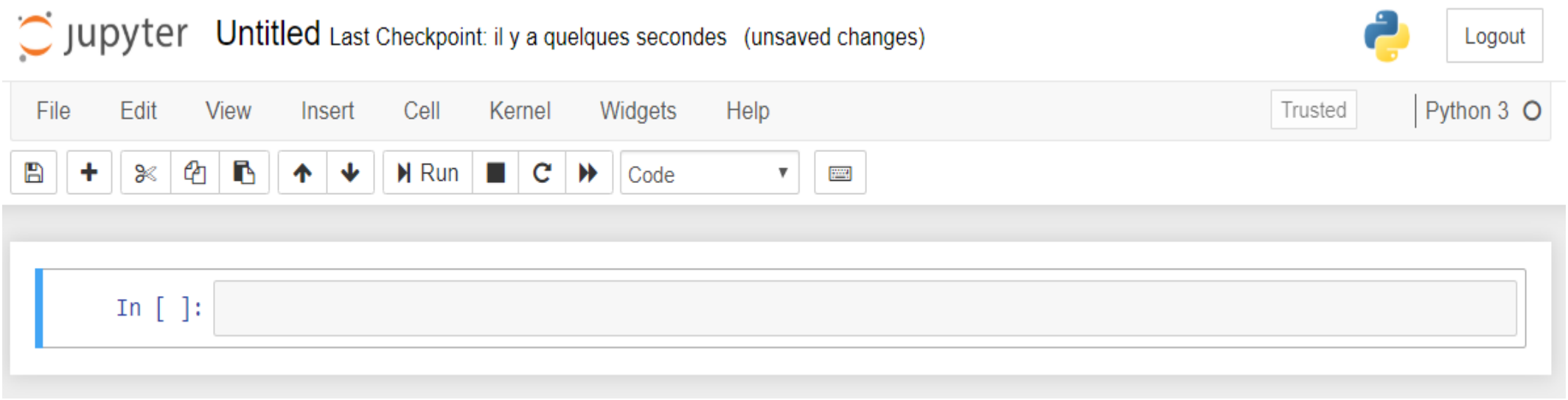
Le Jupyter Notebook

Pour lancer le **Jupyter notebook**, il faut:

- Lancer la plateforme **Anaconda** via l'application **Anaconda Navigator**
- Lancer le **Jupyter Notebook**
- Sélectionner le dossier de travail
- Commencer un nouveau **Notebook** via la commande **New**, tout en spécifiant la version du langage **Python**



Présentation de l'interface Jupyter



La ligne **d'entrée (In [])**, peut comporter plusieurs instructions séparées par un **retour à la ligne**
pour **exécuter** les instructions il suffit de taper simultanément les touches : **Shift** et **Retour à la ligne**.

Les structures de données Pandas

Pandas manipule les structures :

1. **Series**
2. **DataFrame**

N.B. : A chacune de ces structures, est associé un ensemble d'opérateurs et de méthodes.

Les opérateurs relationnels : **<, <=,>, >=, =, !=** et **isin**

Opérateur	Symbole
AND	**&**
OR	**|**
NOT	**~**

Les Séries

- Une Série est un tableau unidimensionnel étiqueté utilisé pour stocker des données de **n'importe** quel type (entier, chaîne, flottant, objets python, **NaN**, etc.).

Series	
index	value
0	12
1	-4
2	7
3	9

Création d'une série

Créer la série **s**=[4 , 8 , 1 , -5 , 3, -10]

s2=[14.25, 17, -5.5, 0.25, -20]

```
from pandas import Series, DataFrame
```

```
s=Series([4 , 8 , 1 , -5 , 3, -10])
```

```
s2=Series([14.25, 17, -5.5, 0.25, -20])
```

Visualiser la série

```
s
```

```
0     4
1     8
2     1
3    -5
4     3
5   -10
dtype: int64
```

```
s2
```

```
0    14.25
1    17.00
2    -5.50
3     0.25
4   -20.00
dtype: float64
```

Création d'une série

Créer une série **s3** comportant les lettres du mot "**Bonjour**"

```
s3=Series(list("Bonjour"))
```

```
s3
```

```
0    B
1    o
2    n
3    j
4    o
5    u
6    r
dtype: object
```

Créer **s4** comportant les entiers entre **5 et 60** avec un pas = **9**

```
s4=Series(list(range(5,60,9)))
```

```
s4
```

```
0     5
1    14
2    23
3    32
4    41
5    50
6    59
dtype: int64
```

Création d'une série

Il est possible de créer une série à partir d'un **dictionnaire**

```
s5 ={"nom":"Ayari","prenom":"Ali","age":17,"fumeur":False}
```

```
s5
```

```
{'nom': 'Ayari', 'prenom': 'Ali', 'age': 17, 'fumeur': False}
```

```
type(s5)
```

```
dict
```

```
s6=Series(s5)
```

```
s6
```

```
nom       Ayari
prenom      Ali
age          17
fumeur    False
dtype: object
```

```
s6[1]
```

```
'Ali'
```

```
s6[1:4]
```

```
prenom      Ali
age          17
fumeur    False
dtype: object
```

```
s6.index
```

```
Index(['nom', 'prenom', 'age', 'fumeur'], dtype='object')
```

```
s6.index[1]
```

```
'prenom'
```

Les indices d'une série

Créer la série **S**= [12, -5, 1.7, True, "Bon"]
et lui associer les indices '**b**', '**k**', '**f**' '**a**' et '**d**'

```
S=Series([12, -5, 1.7, True, "Bon"],
         index=["b","k","f","a","d"])
S
```

```
b       12
k       -5
f      1.7
a     True
d      Bon
dtype: object
```

Afficher les valeurs de la série S

```
S.values
```

```
array([12, -5, 1.7, True, 'Bon'], dtype=object)
```

Afficher les indices de la série S

```
S.index
```

```
Index(['b', 'k', 'f', 'a', 'd'], dtype='object')
```

Les indices d'une série

Afficher le **2ème** élément

```
print(S[1],":",S["k"])
```

```
-5 : -5
```

Afficher les éléments d'indices 0, 2 et 4

```
S[[0,2,4]]
```

```
b     12
f    1.7
d    Bon
dtype: object
```

```
S[['b','f','d']]
```

```
b     12
f    1.7
d    Bon
dtype: object
```

Afficher les éléments d'indices 1, 2 et 3

```
S[1:4]
```

```
k     -5
f    1.7
a   True
dtype: object
```

```
S['k':'d']
```

```
k     -5
f    1.7
a   True
d    Bon
dtype: object
```

```
S[-4:-1]
```

```
k     -5
f    1.7
a   True
dtype: object
```

Les DataFrame

- Un **DataFrame** est une matrice dont les données de chaque colonne sont de même type (nombre, dates, texte), elle peut contenir des valeurs manquantes (**NaN**).
- La matrice est formée par des **variables** (*champs*) en colonnes et des **observations** (*enregistrements*) en ligne.
- Les colonnes et les lignes d'un **dataframe** sont indexées.

DataFrame			
	columns		
index	color	object	price
0	blue	ball	1.2
1	green	pen	1.0
2	yellow	pencil	0.6
3	red	paper	0.9
4	white	mug	1.7

Exemple d'un DataFrame

	City	Edition	Sport	Discipline	Athlete	NOC	Gender	Event	Event_Gender	Medal
0	Athens	1896	Aquatics	Swimming	HAJOS, Alfred	HUN	Men	100m freestyle	M	Gold
1	Athens	1896	Aquatics	Swimming	HERSCHMANN, Otto	AUT	Men	100m freestyle	M	Silver
2	Athens	1896	Aquatics	Swimming	DRIVAS, Dimitrios	GRE	Men	100m freestyle for sailors	M	Bronze
3	Athens	1896	Aquatics	Swimming	MALOKINIS, Ioannis	GRE	Men	100m freestyle for sailors	M	Gold
4	Athens	1896	Aquatics	Swimming	CHASAPIS, Spiridon	GRE	Men	100m freestyle for sailors	M	Silver
5	Athens	1896	Aquatics	Swimming	CHOROPHAS, Efstathios	GRE	Men	1200m freestyle	M	Bronze
6	Athens	1896	Aquatics	Swimming	HAJOS, Alfred	HUN	Men	1200m freestyle	M	Gold
7	Athens	1896	Aquatics	Swimming	ANDREOU, Joannis	GRE	Men	1200m freestyle	M	Silver
8	Athens	1896	Aquatics	Swimming	CHOROPHAS, Efstathios	GRE	Men	400m freestyle	M	Bronze
9	Athens	1896	Aquatics	Swimming	NEUMANN, Paul	AUT	Men	400m freestyle	M	Gold
10	Athens	1896	Aquatics	Swimming	PEPANOS, Antonios	GRE	Men	400m freestyle	M	Silver
11	Athens	1896	Athletics	Athletics	LANE, Francis	USA	Men	100m	M	Bronze

La création d'un dataframe

- Il est possible de créer un dataframe à partir d'une série

```
s=Series([10,20,30,40,50], index=['a','b','c','d','e'])
```

```
s
```

```
a    10
b    20
c    30
d    40
e    50
dtype: int64
```

```
d=DataFrame(s)
```

```
d
```

	0
a	10
b	20
c	30
d	40
e	50

```
d=DataFrame(s,columns=['Age'])
```

```
d
```

	Age
a	10
b	20
c	30
d	40
e	50

La création d'un dataframe

Créer la matrice numpy suivante:

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

```
from numpy import *
m=array([[1,2,3],
         [4,5,6],
         [7,8,9],
         [10,11,12]])
print(m)
```

Créer un dataframe à partir de la **matrice** précédente en nommant les colonnes par **a**, **b** et **c**:

```
dm=DataFrame(m)
print(dm)
```

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9
3	10	11	12

```
dm=DataFrame(m,columns=['a','b','c'])
print(dm)
```

	a	b	c
0	1	2	3
1	4	5	6
2	7	8	9
3	10	11	12

La création d'un dataframe

Renommer les colonnes par **X**, **Y** et **Z**

```
dm.columns=['X','Y','Z']
print(dm)
```

```
    X   Y   Z
0   1   2   3
1   4   5   6
2   7   8   9
3  10  11  12
```

Tester les commandes suivantes:

dm.index	**len(dm)**	**dm.shape**	**dm.dtypes**
dm.columns	**len(dm.columns)**	**dm.shape[0]**	**dm.info()**
dm.values	**len(dm.index)**	**dm.shape[1]**	

La création d'un dataframe

Créer un dataframe à partir du **dictionnaire** suivant :

Hauteur	Largeur
58	25
59	12
60	33
61	14
62	32
63	25
64	55
65	39

```
dico={"hauteur":[58,59,60,61,62,63,64,65],
      "largeur":[25,12,33,14,32,25,55,39]}
df=DataFrame(dico)
print(df,"\n",type(df))
```

```
    hauteur  largeur
0        58       25
1        59       12
2        60       33
3        61       14
4        62       32
5        63       25
6        64       55
7        65       39
 <class 'pandas.core.frame.DataFrame'>
```

La création d'un dataframe

Afficher les noms des colonnes de la matrice.

```
print(df.columns)
```

```
Index(['hauteur', 'largeur'], dtype='object')
```

Créer un dataframe tout en commençant par la colonne **Largeur** puis la colonne **hauteur**

```
df=DataFrame(dico, columns=["largeur","hauteur"])
print(df)
```

```
   largeur  hauteur
0       25       58
1       12       59
2       33       60
3       14       61
4       32       62
5       25       63
6       55       64
7       39       65
```

```
dd=DataFrame(dico, columns=["hauteur"])
print(dd)
```

```
   hauteur
0       58
1       59
2       60
3       61
4       62
5       63
6       64
7       65
```

Il est possible de créer un dataframe avec un nombre limité de colonne.

Statistiques

```
dico={"hauteur":[58,59,60,61,62,63,64,65],
      "largeur":[25,12,33,14,32,25,55,39]}
df=DataFrame(dico)
print(df,"\n",type(df))
```

Statistiques

sum()	Retourne la somme des valeurs
mean()	Retourne la moyenne arithmétique des valeurs
count()	Retourne le nombre des valeurs
std	Retourne l'écarttype des valeurs
average()	Retourne la moyenne géométrique des valeurs
median()	Retourne la médiane des valeurs
min()	Retourne le minimum des valeurs
max()	Retourne le maximum des valeurs
prod()	Retourne le produit des valeurs
var()	Retourne la variance des valeurs
idxmin	Retourne l'indice du minimum des valeurs
idxmax	Retourne l'indice du maximum des valeurs

Statistiques

Syntaxe : dataframe.fonction_stat(axis, skipna)

- **axis** : **0** pour les colonnes, **1** pour les lignes ;
- **skipna** : si **True**, exclue les valeurs manquantes pour effectuer les calculs.

La moyenne de chaque colonne

```
df.mean(0)
```

```
df.mean(axis=0)
```

```
hauteur    61.500
largeur    29.375
dtype: float64
```

La moyenne de chaque ligne

```
df.mean(1)
```

```
df.mean(axis=1)
```

```
0    41.5
1    35.5
2    46.5
3    37.5
4    47.0
5    44.0
6    59.5
7    52.0
dtype: float64
```

Statistiques

Le min, le max et la somme des valeurs de de la colonne ‘hauteur’

```
print(df.hauteur.min(),' :: ',df.hauteur.max(), ' ::: ',df.hauteur.sum())
```

```
58  ::  65  :::  492
```

La moyenne, le nombre et le produit des valeurs de la colonne largeur

```
print(df.largeur.mean(),' :: ',df.largeur.count(), ' ::: ',df.largeur.prod())
```

```
29.375  ::  8  :::  237837600000
```

Statistiques

Ajouter une nouvelle colonne intitulée superficie

```
: df["Superficie"]=(df.hauteur+df.largeur)*2
  print(df)
```

	hauteur	largeur	Superficie
0	58	25	166
1	59	12	142
2	60	33	186
3	61	14	150
4	62	32	188
5	63	25	176
6	64	55	238
7	65	39	208

Statistiques

Ajouter une nouvelle ligne pour stocker le minimum de chaque colonne

```
df.at[8,"hauteur"]=df.hauteur.min()
```

```
df.at[8,"largeur"]=df.largeur.min()
```

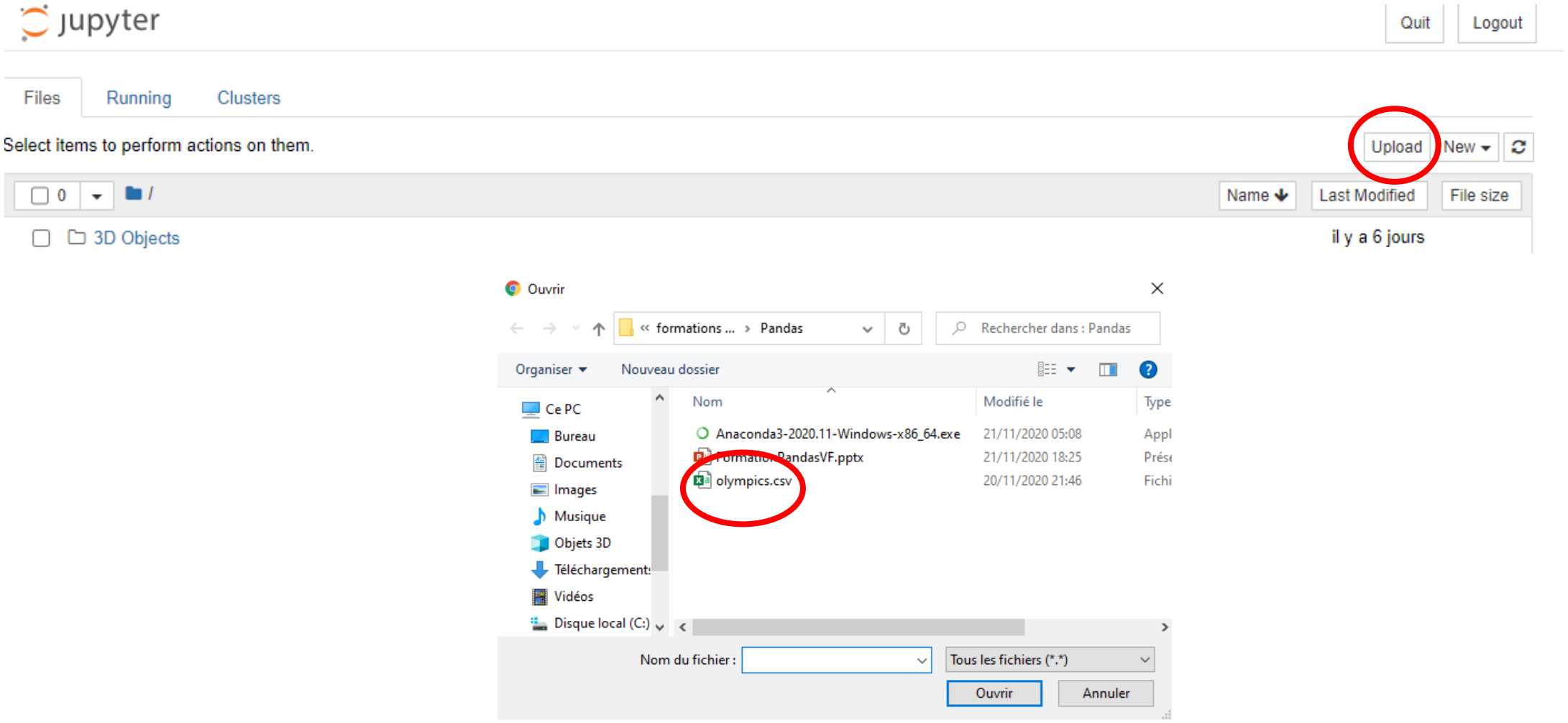
	hauteur	largeur	Superficie
0	58.0	25.0	166.0
1	59.0	12.0	142.0
2	60.0	33.0	186.0
3	61.0	14.0	150.0
4	62.0	32.0	188.0
5	63.0	25.0	176.0
6	64.0	55.0	238.0
7	65.0	39.0	208.0
8	58.0	12.0	NaN

Importation des données

- Ouvrir le fichier **olympics.csv** du dossier de travail.
- Endéduire sa **taille** : Nombre de lignes et Nombre de colonnes
- Comment faire pour analyser ces données via la bibliothèque **Pandas** du langage **Python**.
- En Pandas il est possible de lire des données brutes depuis plusieurs formats: **Excel, Json, SQL, Csv,....** a travers les fonctions : **read_excel(), read_json(), read_csv()**
- **CSV : Comma Seperated Values** → un fichier CSV est un fichier qui comporte des valeurs séparées par des virgules.

Importation des données

- Importer le fichier ‘olympics,csv’ dans le dossier de travail



Importation des données

- Créer un nouveau notebook (python3)
- Importer la bibliothèque Pandas

```
from pandas import *
```

- Stocker dans la variable **df**, les données du fichier **olympics.csv.**

```
df=read_csv('olympics.csv')
```

- Afficher le **type** de la variable **df**

```
type(df)
```

```
pandas.core.frame.DataFrame
```

- Afficher le **contenu** de la variable **df**

Importation des données

- Importer les données du fichier **olympics.csv**, toute en ignorant les 4 premières lignes

```
df=read_csv('olympics.csv',skiprows=4)
```

- Afficher le **contenu** de la variable **df**

Accès aux colonnes

Créer une variable **a** comportant les données de la colonne **City** puis afficher son **type**.

```
a=df['City']
```

```
a=df.City
```

```
type(a)
```

```
pandas.core.series.Series
```

Créer une variables **b** comportant les données des colonnes **City**, **Edition** et **Athlete** puis afficher son **type**.

```
b=df[['City','Edition','Athlete']]
```

```
type(b)
```

```
pandas.core.frame.DataFrame
```

Manipulation d'un DataFrame

Déterminer la taille de la variable df

```
df.shape
```

```
(29216, 10)
```

Déterminer le nombre de lignes du dataframe df

```
df.shape[0]
```

```
29216
```

Déterminer le nombre de colonnes du dataframe df

```
df.shape[1]
```

```
10
```

Manipulation d'un DataFrame

Afficher les 8 premières lignes du dataframe df

```
df.head(8)
```

Afficher les 4 dernières lignes du dateframe df

```
df.tail(4)
```

Testez les deux commandes suivantes:

```
df.head()
```

```
df.tail()
```

Manipulation d'un DataFrame

- La fonction **value_counts()** : Permet de compter le nombre d'occurrences de chaque valeur d'une colonne(Série)

Exemple1 : afficher le nombre de médailles par genre :

```
df.Gender.value_counts()
```

```
Men      21721
Women     7495
Name: Gender, dtype: int64
```

```
df.Gender.value_counts(ascending=True)
```

```
Women     7495
Men      21721
Name: Gender, dtype: int64
```

Manipulation d'un DataFrame

- La fonction **sort_values()**: Permet de renvoyer un dataframe avec les lignes triées selon une ou plusieurs colonnes.

Exemple1 :Trier le dataframe en sens croissant des athlètes

```
df.sort_values(by=['Athlete'])
```

```
df.sort_values('Athlete')
```

Exemple2 : Afficher les éléments du df triés sur les colonnes '**NOC**', puis par '**Gender**' enfin par '**Edition**' des athlètes.

```
df.sort_values(by=['NOC','Gender','Edition'])
```

ou

```
df.sort_values(['NOC','Gender','Edition'])
```

Manipulation d'un DataFrame

Exemple3 :Afficher uniquement les 4 premiers athlètes du df

```
df.sort_values(['NOC','Gender','Edition']).head(4)
```

	City	Edition	Sport	Discipline	Athlete	NOC	Gender	Event	Event_gender	Medal
28965	Beijing	2008	Taekwondo	Taekwondo	NIKPAI, Rohullah	AFG	Men	- 58 kg	M	Bronze
19323	Seoul	1988	Sailing	Sailing	BOERSMA, Jan D.	AHO	Men	board (division II)	M	Silver
17060	Los Angeles	1984	Boxing	Boxing	ZAOUI, Mohamed	ALG	Men	71-75kg	M	Bronze
17064	Los Angeles	1984	Boxing	Boxing	MOUSSA, Mustapha	ALG	Men	75 - 81kg (light-heavyweight)	M	Bronze

Manipulation d'un DataFrame

L'indexation booléenne :

Exemples:

1- Afficher les athlètes qu'ont remporté des médailles d'or

```
df.Medal=='Gold'
```

```
df[df.Medal=='Gold']
```

2- Afficher les athlètes féminins qu'ont remporté des médailles d'or

```
df[(df.Medal=='Gold') & (df.Gender=='Women')]
```

3- Afficher les athlètes féminins qu'ont remporté des médailles d'or dans les éditions <1900 ou les éditions >2000

```
df[(df.Medal=='Gold') & (df.Gender=='Women')&((df.Edition >2000)|(df.Edition<1900))]
```

Manipulation d'un DataFrame

L'indexation booléenne :

4- Afficher uniquement les noms de ces athlètes.

```
df[(df.Medal=='Gold') & (df.Gender=='Women')&((df.Edition >2000)|(df.Edition<1900))]
```

```
df[(df.Medal=='Gold')&(df.Gender=='Women')&((df.Edition>2000)|(df.Edition<1900))].Athlete
```

ou

```
df[(df.Medal=='Gold')&(df.Gender=='Women')&((df.Edition>2000)|(df.Edition<1900))]['Athlete']
```

Manipulation d'un DataFrame

L'indexation booléenne :

5- Afficher uniquement les noms et les nationalités de ces athlètes.

```
df[(df.Medal=='Gold')&(df.Gender=='Women')&((df.Edition>2000)|(df.Edition<1900))][['Athlete','NOC']]
```

6- Afficher les noms et les nationalités des 3 premiers de ces athlètes.

```
df[(df.Medal=='Gold')&(df.Gender=='Women')&((df.Edition>2000)|(df.Edition<1900))][['Athlete','NOC']].head(3)
```

	Athlete	NOC
25180	NEWBERY, Chantelle	AUS
25186	GUO, Jingjing	CHN
25196	LAO, Lishi	CHN

Manipulation d'un DataFrame

Manipulation des chaînes de caractères:

Nom_Colonne.str.contains(), *Nom_Colonne.str.startswith(),*
Nom_Colonne.str.endswith(), *Nom_Colonne.str.isnumeric().*

Exemple 1 : Trouver tous les participants tunisiens aux jeux olympiques.

```
df[df.NOC.str.contains('TUN')]
```

Application

A quel Edition MELLOULI, Oussama a gagné une médaille?

```
df[df.Athlete=='MELLOULI, Oussama']
```

```
df[df.Athlete.str.contains('MELLOULI')]
```

	City	Edition	Sport	Discipline	Athlete	NOC	Gender	Event	Event_gender	Medal
27237	Beijing	2008	Aquatics	Swimming	MELLOULI, Oussama	TUN	Men	1500m freestyle	M	Gold

Application

Quel sont les 3 pays qu'ont remporté le plus de médailles de l'année 1984 à l'année 2008

```
rr=df[(df.Edition>=1984) & (df.Edition<=2008)]
```

```
rr.NOC.value_counts()
```

```
USA    1837
AUS     762
GER     691
CHN     679
RUS     638
       ...
URU       1
ERI       1
MRI       1
ISV       1
MKD       1
Name: NOC, Length: 124, dtype: int64
```

```
rr.NOC.value_counts().head(3)
```

```
USA    1837
AUS     762
GER     691
Name: NOC, dtype: int64
```

Sélection et découpage (Séries)

Générer une série ss comportant uniquement les sports puis afficher son type

```
ss=df.Sport
```

```
type(ss)
```

```
pandas.core.series.Series
```

```
s1
```

```
0          Aquatics
1          Aquatics
2          Aquatics
3          Aquatics
4          Aquatics
             ...
29211     Wrestling
29212     Wrestling
29213     Wrestling
29214     Wrestling
29215     Wrestling
Name: Sport, Length: 29216, dtype: object
```

Générer une série ss comportant les différents sports puis afficher son type

```
ss=df.Sport.unique()
```

```
type(ss)
```

```
numpy.ndarray
```

Merci pour votre attention