Jour-25 Pandas

Pandas

Pandas est un open source, haute performance et des structures de données faciles à utiliser et des outils d'analyse des données pour le langage de programmation Python. Pandas ajoute de s structures et des outils de données conçus pour fonctionner avec des données de type table qui sont *Series* et *Data Frames*. Pandas fournit des outils pour la manipulation des donnée s:

- Reshaper • Merging • tri • Slicing • Aggrégation • Imputation. Si vous utilisez An aconda, vous n'avez pas d'installation de pandas.

Installation de pandas

Pour Mac:

```
PIP Install Conda Conda Install
er Pandas pour Windows: PIP I
nstaller Conda PIP Installer Pan
das
```

La structure de données Pandas est basée sur *Series* et *DataFrames*.

Un *series* est un *column* et un dataframe est un *multidimensional table* composé de collect e de *series*. Afin de créer une série Pandas, nous devons utiliser Numpy pour créer des tablea ux unidimensionnels ou une liste Python. Voyons un exemple d'une série:

Noms Pandas Series

	Name
0	Asabeneh
1	David
2	John

Série de pays

	Country
0	Finland
1	UK
2	Sweden

Séries de villes

	City
0	Helsinki
1	London
2	Stockholm

Comme vous pouvez le voir, la série Pandas n'est qu'une colonne de données. Si nous voulons avoir plusieurs colonnes, nous utilisons des trames de données. L'exemple ci-dessous montre le s frames de données Pandas.

Voyons, un exemple de cadre de données Pandas:

	Name	Country	City
0	Asabeneh	Finland	Helsinki
1	David	UK	London
2	John	Sweden	Stockholm

Data Frame est une collection de lignes et de colonnes. Regardez le tableau ci-dessous; Il a be aucoup plus de colonnes que l'exemple ci-dessus:

	Name	Country	City	Weight	Height
0	Asabeneh	Finland	Helsinki	74	173
1	David	UK	London	78	175
2	John	Sweden	Stockholm	69	169

Ensuite, nous verrons comment importer des pandas et comment créer des séries et des dataf rames à l'aide de pandas

Importation de pandas

```
Importer des pandas sous le nom de PD # Importation de pandas en
tant que PD importent Numpy que NP # importation Numpy comm
e np
```

Création d'une série Pandas avec un index par défaut

```
nums = [1, 2, 3, 4,5]
s = pd.series (nums) print (s)
```

```
    0    1
    1    2
    2    3
    3    4
    4    5
    dtype: int64
```

Création d'une série Pandas avec un index personnalisé

```
nums = [1, 2, 3, 4, 5] s = pd.series (nums, index = [1, 2, 3, 4, 5]
)
imprimer (s)
```

```
    1    1
    2    2
    3    3
    4    4
    5    5
    dtype: int64
```

```
fruits = ['orange', 'banana', 'mango'] fruits = pd.series (fruits, index
= [1, 2, 3]) imprimer (fruits)
```

```
    1    Orange
    2    Banana
    3    Mango
    dtype: object
```

Création d'une série Pandas à partir d'un dictionnaire

```
DCT =
{'name': 'Asabeneh', 'country': 'Finlande', 'City': 'Helsinki'}
```

```
S = Pd.Series (DCT)
imprimer (s)
```

```
    name       Asabeneh
    country     Finland
    city       Helsinki
    dtype: object
```

Création d'une série Pandas constante

```
s = pd.series (10, index = [1, 2, 3])
imprimer (s)
```

```
   1    10
    2    10
    3    10
    dtype: int64
```

Création d'une série Pandas à l'aide de Linspace

```
S = Pd.Series (np.linspace (5, 20, 10)) # lispace (Démarrage, fin, éléments) Impression (s
)
```

```
    0     5.000000
    1     6.666667
    2     8.333333
    3    10.000000
    4    11.666667
    5    13.333333
    6    15.000000
    7    16.666667
    8    18.333333
    9    20.000000
    dtype: float64
```

Dataframe

Les cadres de données Pandas peuvent être créés de différentes manières.

Création de données de données à partir de la liste des listes

```
data = [
    ['Asabeneh', 'Finland', 'Helsink'],
    ['David', 'UK', 'London'],
    ['John', 'Sweden', 'Stockholm']
```

```
]]
df = pd.dataframe (data, colonnes = ['noms', 'country', 'ville']))
Imprimer (DF)
```

	Names	Country	City
0	Asabeneh	Finland	Helsink
1	David	UK	London
2	John	Sweden	Stockholm

Création de données de données à l'aide du dictionnaire

```
Data = {'name': ['Asabeneh', 'David', 'John'], 'Country': ['Finland', 'UK', 'Suède'], 'City': ['Helsi
ki', 'Londres', 'Stockholm']} DF = Pd.Dataframe (Data) Print (DF)
```

	Name	Country	City
0	Asabeneh	Finland	Helsiki
1	David	UK	London
2	John	Sweden	Stockholm

Création de données de données à partir d'une liste de dictionnaires

```
Data = [{'name': 'Asabeneh', 'country': 'Finland', 'City': 'Helsinki'}, {'name': 'David', 'Co
untry': 'UK', 'City': 'London'}, {'name': 'John', 'Country': 'Sweden', 'City': 'Stockholm' P
D.DATAFRAME (DATA) PRINT (DF)
```

	Name	Country	City
0	Asabeneh	Finland	Helsinki
1	David	UK	London
2	John	Sweden	Stockholm

Lire le fichier CSV à l'aide de pandas

Pour télécharger le fichier CSV, ce qui est nécessaire dans cet exemple, la ligne de console / de commande est suffisante:

```
curl -o https://raw.githubusercontent.com/asabeneh/30-days-of- python / maître / data / poids-height.csv
```

Mettez le fichier téléchargé dans votre répertoire de travail.

```
Importer des pandas en tant que PD

df = pd.read_csv ('poids-height.csv')
Imprimer (DF)
```

Exploration des données

Lissons uniquement les 5 premières lignes en utilisant Head ()

```
print (df.head ()) # Donnez cinq lignes, nous pouvons augmenter le nombre de lignes en pas sant l'argument à la méthode de la tête ()
```

	Gender	Height	Weight
0	Male	73.847017	241.893563
1	Male	68.781904	162.310473
2	Male	74.110105	212.740856
3	Male	71.730978	220.042470
4	Male	69.881796	206.349801

Explorons également les derniers enregistrements du DataFrame à l'aide des méthodes Tail ().

```
imprimer (df.tail ()) # que les queues donnent les cinq dernières lignes, nous pouvons augmenter les lignes en passant la méthode de l'argument à la queue
```

	Gender	Height	Weight
9995	Female	66.172652	136.777454
9996	Female	67.067155	170.867906
9997	Female	63.867992	128.475319
9998	Female	69.034243	163.852461
9999	Female	61.944246	113.649103

Comme vous pouvez le voir, le fichier CSV a trois lignes: le sexe, la taille et le poids. Si le Dat aFrame aurait de longues lignes, il serait difficile de connaître toutes les colonnes. Par conséq uent, nous devons utiliser une méthode pour connaître les Colums. Nous ne connaissons pas le nombre de lignes. Utilisons la forme de la forme de Meathod.

```
imprimer (df.shape) # Comme vous pouvez voir 10000 lignes et trois colonnes
```

```
(10000, 3)
```

Laissez-nous obtenir toutes les colonnes à l'aide de colonnes.

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

```
Index (['Gender', 'Height', 'Weight'], DTYPE = 'Object')
```

Maintenant, obtenons une colonne spécifique en utilisant la clé de colonne

```
Heights = df ['height'] # c'est maintenant une série
```

```
Imprimer (Heights)
```

```
    0       73.847017
    1       68.781904
    2       74.110105
    3       71.730978
    4       69.881796
              ...
    9995    66.172652
    9996    67.067155
    9997    63.867992
    9998    69.034243
    9999    61.944246
    Name: Height, Length: 10000, dtype: float64
```

```
poids = df ['poids'] # C'est maintenant une série
```

```
imprimer (poids)
```

```
    0       241.893563
    1       162.310473
    2       212.740856
    3       220.042470
    4       206.349801
               ...
    9995    136.777454
    9996    170.867906
    9997    128.475319
    9998    163.852461
    9999    113.649103
    Name: Weight, Length: 10000, dtype: float64
```

```
print (len (Heights) == len (poids)))
```

```
Vrai
```

La méthode décrit () fournit une valeur statistique descriptive d'un ensemble de données.

imprimer (heights.describe ()) # donner des informations statistiques sur les données de haute ur

```
count    10000.000000
mean        66.367560
std          3.847528
min         54.263133
25%         63.505620
50%         66.318070
75%         69.174262
max         78.998742
Name: Height, dtype: float64
```

Imprimer (poids.Describe ())

```
count    10000.000000
mean       161.440357
std         32.108439
min         64.700127
25%        135.818051
50%        161.212928
75%        187.169525
max        269.989699
Name: Weight, dtype: float64
```

print (df.describe ()) # décrire peut également donner des informations statistiques à parti r d'un dataframe

	Height	Weight
count	10000.000000	10000.000000
mean	66.367560	161.440357
std	3.847528	32.108439
min	54.263133	64.700127
25%	63.505620	135.818051
50%	66.318070	161.212928
75%	69.174262	187.169525
max	78.998742	269.989699

Similaire à décrire (), la méthode info () donne également des informations sur l'ensemble de données.

Modification d'un dataframe

Modification d'un dataframe: * Nous pouvons créer un nouveau DataFrame * Nous pouvons créer une nouvelle colonne et les ajouter à DataFrame, * Nous pouvons supprimer une colon ne existante à partir d'un DataFrame, * Nous pouvons modifier une colonne existante dans u n DataFrame, * Nous pouvons modifier le type de données des valeurs de colonne dans le da taframe Data

Création d'un dataframe

Comme toujours, nous importons d'abord les packages nécessaires. Maintenant, importe d es pandas et Numpy, deux meilleurs amis de tous les temps.

```
Importer des pandas en tant que Pd Import Numpy comme NP Data = [{"Name": "Asabene
h", "Country": "Finland", "City": "Helsinki"}, {"Name": "David", "Country": "UK", "City":
"London", "City": "Nom" "] df = pd.dataframe (data) print (df)
```

	Name	Country	City
0	Asabeneh	Finland	Helsinki
1	David	UK	London
2	John	Sweden	Stockholm

L'ajout d'une colonne à un dataframe, c'est comme l'ajout d'une clé à un dictionnaire.

Utilisons d'abord l'exemple précédent pour créer un dataframe. Après avoir créé le Data Frame, nous commencerons à modifier les colonnes et les valeurs des colonnes.

Ajout d'une nouvelle colonne

Ajoutons une colonne de poids dans le dataframe

```
poids = [74, 78, 69] df ['poids'] =
poids df
```

	Name	Country	City	Weight
0	Asabeneh	Finland	Helsinki	74
1	David	UK	London	78
2	John	Sweden	Stockholm	69

Ajoutons une colonne de hauteur dans le dataframe aussi

```
Heights = [173, 175, 169]
df ['height'] = hauteurs imprimées
(df)
```

	Name	Country	City	Weight	Height
0	Asabeneh	Finland	Helsinki	74	173
1	David	UK	London	78	175
2	John	Sweden	Stockholm	69	169

Comme vous pouvez le voir dans le DataFrame ci-dessus, nous avons ajouté de nouvelles col onnes, du poids et de la hauteur. Ajoutons une colonne supplémentaire appelée BMI (indice d e masse corporelle) en calculant leur IMC en utilisant leur masse et hauteur. L'IMC est masse divisé par la hauteur carrée (en mètres) - poids / hauteur * hauteur.

Comme vous pouvez le voir, la hauteur est en centimètres, donc nous le changeons en mètres. Modifions la ligne de hauteur.

Modification des valeurs de colonne

```
df ['height'] = df ['height'] * 0,01
```

```
df
```

	Name	Country	City	Weight	Height
0	Asabeneh	Finland	Helsinki	74	1.73
1	David	UK	London	78	1.75
2	John	Sweden	Stockholm	69	1.69

```
# L'utilisation de fonctions rend notre code propre, mais vous pouvez calculer l'IMC sans un
def calcul_bmi (): poids = df ['poids'] hauteurs = df ['height'] bmi = [] pour w, h en zip (poids, hauteurs): b = w / (h * h)









BMI = calcul_bmi ()                df ['bmi'] = bmi df
```

	Name	Country	City	Weight	Height	BMI
0	Asabeneh	Finland	Helsinki	74	1.73	24.725183
1	David	UK	London	78	1.75	25.469388
2	John	Sweden	Stockholm	69	1.69	24.158818

Colonnes de dataframe de formation

Les valeurs de la colonne BMI du DataFrame sont flottantes avec de nombreux chiffres signi ficatifs après décimal. Changeons-le en un chiffre significatif après le point.

```
df ['bmi'] = round (df ['bmi'], 1)
Imprimer (DF)
```

	Name	Country	City	Weight	Height	BMI
0	Asabeneh	Finland	Helsinki	74	1.73	24.7
1	David	UK	London	78	1.75	25.5
2	John	Sweden	Stockholm	69	1.69	24.2

Les informations dans le dataframe ne semblent pas encore terminées, ajoutons l'année de n aissance et les colonnes de l'année en cours.

```
naissance_year = ['1769', '1985', '1990'] current_year = pd.series (202
0, index = [0, 1,2])
df ['année de naissance'] = naissance_year df ['anné
e en cours'] = current_year df
```

	Name	Country	City	Weight	Height	BMI	Birth Year	Current Year
0	Asabeneh	Finland	Helsinki	74	1.73	24.7	1769	2020
1	David	UK	London	78	1.75	25.5	1985	2020
2	John	Sweden	Stockholm	69	1.69	24.2	1990	2020

Vérification des types de données de valeurs de colonne

```
print (df.weight.dtype)                d
type ('int64')
```

```
df [«année de naissance»]. DTYPE # Il donne un objet String, nous devons changer cela e
n numéro
```

```
df ['année de naissance'] = df ['naissance'].
```

```
        dtype ('int32')
```

Maintenant identique pour l'année en cours:

```
df ['année en cours'] = df ['année en cours "]. Astype (' int ') df [' année en cours"]
. DTYPE


        dtype ('int32')
```

Maintenant, les valeurs de la colonne de l'année de naissance et de l'année en cours sont des en tiers. Nous pouvons calculer l'âge.

```
ages = df['Current Year'] - df['Birth Year']
ages
```

```
0    251
1     35
2     30
dtype: int32
```

DF ['Ages'] = AGES PRIN
T (DF)

	Name	Country	City	Weight	Height	BMI	Birth Year	Current Year	Ages
0	Asabeneh	Finland	Helsinki	74	1.73	24.7	1769	2019	250
1	David	UK	London	78	1.75	25.5	1985	2019	34
2	John	Sweden	Stockholm	69	1.69	24.2	1990	2019	29

La personne de la première rangée a vécu jusqu'à présent pendant 251 ans. Il est peu probable que quelqu'un vive si longtemps. Soit c'est une faute de frappe, soit les données sont cuites. Pe rmet donc de remplir ces données avec la moyenne des colonnes sans inclure la valeur aberran te.

moyenne = (35 + 30) / 2

```
mean = (35 + 30)/ 2
print('Mean: ',mean)      #it is good to add some description
to the output, so we know what is what
```

Moyenne: 32,5

Indexation booléenne

print (df [df ['viets'] > 120])

	Name	Country	City	Weight	Height	BMI	Birth Year	Current Year	Ages
0	Asabeneh	Finland	Helsinki	74	1.73	24.7	1769	2020	251

```
print (df [df ['viets'] < 120])
```

	Name	Country	City	Weight	Height	BMI	Birth Year	Current Year	Ages
1	David	UK	London	78	1.75	25.5	1985	2020	35
2	John	Sweden	Stockholm	69	1.69	24.2	1990	2020	30

Exercices: Jour 25

1. Lisez le fichier hacker_news.csv à partir du répertoire de données 2. Obtenez les cinq premières lignes 3. Obtenez les cinq dernières lignes 4. Obtenez la colonne de titre en tant q ue Pandas Series 5. Comptez le nombre de lignes et colonne s o Filtrez les titres qui contiennent Python o filtrent les titre s qui contiennent javascrip

Félicitations!