Pandas : Data Manipulation

Premiers Pas :

Création des objets

```
albums = pd.DataFrame({
    "Album" : ["Animals","Animalisms"],
    "Artist" : ["Pink Floyd","The Animals"],
    "Year": [1977,1966]}
)
```

	Album	Artist	Year
0	Animals	Pink Floyd	1977
1	Animalisms	The Animals	1966

```
albums_sales = pd.Series(
    ["326.2 million","240.7 million","141 million"],
    index = ["2010","2015","2018"],
    name="AlbumSalesUS"
)
```

```
2010    326.2 million
2015    240.7 million
2018      141 million
Name: AlbumSalesUS, dtype: object
```

Récupération des données

Head of the file *albums.csv* :

```
,title,artists,year_1,year_2,price,sales,genre
0,The Dark Side of the Moon,Pink Floyd,1973,1973,15,45000000,progressive rock
1,Rumours,Fleetwood Mac,1976,1977,10,40000000,soft rock
```

To get a DataFrame from a file :
`pd.read_csv("../input/albums/albums.csv",index_col=0)`

To put a DataFrame in a file :
`pd.DataFrame.to_csv(path_or_buf='./albums.csv',self=albums)`

	title	artists	...	genre
0	The Dark Side of the Moon	Pink Floyd	...	progressive rock
1	Rumours	Fleetwood Mac	...	soft rock
...	...	...	...	...
6	Appetite For Destruction	Guns N' Roses	...	hard rock
7	The Eminem Show	Eminem	...	hip hop

Selection des données

Command	Result	Example
albums.genre	DF of column genre with indexing	
albums.genre.iloc[0]	1^{st} value of genre column	
albums.iloc[j,i]	column(s) j and row(s) i of DF; accepts int only; does not include j	albums.iloc[[0,1,4,7],[0,2,5,6]]
albums.loc[j,i]	column(s) j and row(s) i of DF; accepts every types; includes j	albums.loc[[0,1,4,7],['title', 'year_1', 'sales', 'genre']]
.(i)loc[j,i]	[:10] is from 0 to 10; ':' is everything	albums.loc[:5]; albums.iloc[:,[0,1,2]]
Conditions		
==,<,>,>=,<=	selecting specific values by comparing them	albums.loc[albums.artists == 'Pink Floyd'] albums[albums.sales>= 40000000]
|,&	combines conditions (or,and)	albums[((albums.artists == 'Pink Floyd') | (albums.artists == 'Led Zeppelin')) & (albums.price <= 15)] Every Pink Floyd or Led Zeppelin album costing less than 15

*DF=DataFrame

Avancé :

Fonctions utiles

Function	Result
.median()	median of int column of DF
.unique()	removes duplications
.value_counts()	counts occurrences
.mean()	mean of int column of DF
.idxmax()	index of ONLY the 1^{st} occurrence of maximum values

Autres Fonctions

map

Searches for 'rock' in albums'genre
albums.genre.map(lambda d: 'rock' in d)

```
0     True
1     True
     ...
6     True
7    False
```

methods definition

```
def stars_count(row):
    if (row.sales >= 45000000) | (row.artists == 'Pink Floyd'):
        return 3
    elif row.sales >=30000000:
        return 2
    else:
        return 1
albums.apply(stars_count,axis="columns")
```

```
0    3
1    2
     ..
6    2
7    1
```

Tri des données

Grouped by artists and shows how many times they occur :
albums.groupby('artists').artists.count()

Sales min and max of each artist :
sales = albums.groupby('artists').sales.agg([min,max])

Sorting those sales from max to min :
sales.sort_values(by=['min','max'],ascending=False)

Typage des données

Function	Result
.dtype	data's type
.astype('*type*')	converts the data into *type*
.isnull()	tells if data is null (NaN)
.fillna('*name*')	fills null values with *name*

Modification des axes

albums.rename(columns={'year_1': 'recorded','year_2': 'release'})