

Importare pandas e numpy

```
import pandas as pd
importa la libreria pandas

import numpy as np
importa la libreria numpy

pd.set_option('max_columns', 300)
limitare il numero di colonne stampate in output:*

pd.set_option('max_rows', 1000)
limitare il numero di righe stampate in output:*
```

Creare una serie

```
my_data = [10,20,30] e labels = ['A', 'B', 'C']
pd.Series(data=my_data, index=labels)
```

Caricare un file in un dataframe

```
df=pd.read_csv("file.csv")
Legge un file csv

parametri aggiuntivi:

header specifica la riga da usare come header (int type), nel caso non sia presente: header =None

sep=';' specifica il separatore usato nel file csv

names= ['col1', 'col2'] esplicita il nome delle colonne del dataframe, utile soprattutto in caso non ci sia header nel file

parse_dates = ['col1', 'col2'] lista di interi nomi di colonne da tentare di parsare come data.

df=pd.read_excel("nome file.xlsx") legge un file excel
```

Salvare un dataframe in csv

```
df.to_csv('path/file_name.csv', sep=';', index=False)
salva il dataframe in un file csv (rimuovendo l'indice e usando come separatore il ;)

df.to_csv('path/file_name.csv', sep=';', index=True)
```

Esplorare un dataframe

```
df.shape
restituisce (numero righe, numero colonne)

df[df.isnull().any(axis=1)]
controlla se ci sono valori nulli

df.dtypes
restituisce il tipo di variabile di ogni colonna

df.head()
restituisce le prime (n) righe di un dataframe (default = 5)

df.tail()
restituisce le ultime (n) righe di un dataframe (default = 5)

df.columns
restituisce l'elenco delle colonne del dataframe

df.describe()
restituisce una descrizione del dataframe con le statistiche di base
```

Gestire i dati mancanti

```
df.dropna(axis=0)
Elimina le righe con valori null presenti in qualsiasi colonna

df.dropna(axis=1)
Elimina le colonne con valori null

df.dropna(axis=0, thresh=5)
Elimina le righe con almeno 5 valori null presenti in qualsiasi colonna

df.fillna(value='VALORE NUOVO')
Riempi le righe con valori null usando il valore value indicato tra parentesi
```

Reshape dei dati

```
df.sort_values(by= ['col1', 'col2'])
ordina il dataframe per colonna o array di colonne

df.sort_values(by= 'col1', ascending=False)
ordina il dataframe per colonna in ordine discendente
```

Reshape dei dati (cont)

```
df.sort_index()
ordina l'indice del dataframe

df.reset_index()
resetta l'indice del dataframe usando l'ordine c colonna

df.reset_index(name='new name')
asigna il nome alla colonna indice durante la

df.drop(columns= ['col1', 'col2'])
elimina le colonne 'col1' e 'col2'

df.rename(columns= {'old_col': 'new_col'})
rinomina le colonne (usando un dizionario)

df.concat([df1, df2])
aggiunge le righe del df2 al df1

df.concat([df1, df2], axis=1)
aggiunge le colonne del df2 al df1
```

ricavare un sottoinsieme di un dataframe

per righe

```
df[df['colonna1']>n]
seleziona tutte le righe dove nella colonna1 i v maggiori di n

df.drop_duplicates()
rimuove le righe duplicate

df.iloc[10:20]
seleziona tutte le righe dalla 10 alla 20

df[df['colonna1'].isin(['value1', 'value2'])]
seleziona tutte le righe dove il valore nella colonna è uguale a value1 o value2

df['colonna1'].unique()
restituisce i valori unici di una colonna


per colonne



```
df['colonna1'].to_csv('file.csv')
seleziona la colonna 1
```


```

ricavare un sottoinsieme di un dataframe (cont)

```
df[['c olo nna 1', 'co lon na2 ', 'co lon na3 ']]
```

Seleziona le colonne 1,2,4

```
df.iloc[10:20, [1,2,5]]
```

seleziona tutte le righe dalla 10 alla 20 e le colonne in posizione 1,2,5

```
df.loc[:, 'x2':'x4']
```

seleziona tutte le colonne dalla posizione x2 alla posizione x4 (compresa)

```
df.loc [df ['a ']> 10, ['a ','c']]
```

seleziona le colonne 'a' e 'c' dove il valore di a è maggiore di 10

Varie

```
df.columns
```

restituisce l'elenco delle colonne

```
del df['co l_1']
```

elimina una colonna

Riassumere i dati

```
len(df)
```

number of rows in a dataframe

```
df['co lum n1' ].n uni que()
```

numero di valori unici in una colonna

```
df['co lum n1' ].sum()
```

somma dei valori di una colonna

```
df['co lum n1' ].min()
```

minimo dei valori di una colonna

```
df['co lum n1' ].max()
```

massimo dei valori di una colonna

```
df['co lum n1' ].m ean()
```

media dei valori di una colonna

```
df['co lum n1' ].var()
```

varianza dei valori di una colonna

```
df['co lum n1' ].std()
```

deviazione standard dei valori di una colonna

Riassumere i dati (cont)

```
df['co lum n1' ].c ount()
```

count dei valori non nulli di una colonna

```
df['co lum n1' ].m edian()
```

mediana dei valori di una colonna

```
df['co lum n1' ].q uan til e([.25, .75])
```

quantile dei valori di una colonna

```
df['co lum n1' ].v alu e_c ount()
```

count di ogni valore di una colonna

Gestione oggetti di tipo datetime

```
df['Da ta' ]=p d.t o_d ate tim e(df[' Data'])
```

converte stringhe in oggetti datetime

```
df['Da ta' ].d t.r oun d('1s')
```

arrotonda la data al secondo

```
df['Du rat ion ']= (df ['D ata 2'] -df ['D ata 1'] ).d t.s econds
```

Calcola la differenza tra due date restituendo un risultato in secondi

```
df['Da ta' ].d t.date
```

estrae solo la data da data ora

```
df['Da ta' ].d t.month
```

estrae il mese

```
df['Da ta' ].d t.week
```

estrae la settimana dell'anno

```
df['Da ta' ].d t.w eekday
```

estrae il giorno della settimana

```
df['Da ta' ].d t.w eek day _name
```

estrae il giorno della settimana (come nome)

```
df['Da ta' ].d t.time
```

estrae l'ora completa

```
df['Da ta' ].d t.hour
```

estrae l'ora

```
df['St agi one ']= pd.cut( (df['D ata '].d t.d ay ofyear + 11) % 365, bins=4, labels=['primavera', 'estate', 'autunno'])
```

crea una colonna con la corrispondente stagione

Crea ed elimina colonne/righe

```
df['ne w_c olumn'] = df['co l_1'] + df['co l_2']
```

crea una nuova colonna somma di due colonne

```
df = df.dro p(' col_1', axis=1)
```

elimina la colonna *col_1*

```
df = df.drop(0, axis=0)
```

elimina la riga con indice 0

