

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("nomefile.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 ;)
```

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')
Riempì 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

df.sort_index()
ordina l'indice del dataframe

df.reset_index()
resetta l'indice del dataframe usando l'ordine
delle righe, l'indice attuale viene salvato in una
nuova colonna

df.reset_index(name='new name')
assegna il nome alla colonna indice durante la
fase di reset

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

df.rename(columns={'old_column':
'new column', 'old_column2': 'new
column2'})
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
valori sono maggiori di n

df.drop_duplicates()
rimuove le righe duplicate

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

ricavare un sottoinsieme di un dataframe (cont)

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

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

per colonne

`df['colonna1']` o `df.colonna1`
seleziona la colonna 1

`df[['colonna1', 'colonna2', 'colonna4']]`
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['col_1']`
elimina una colonna

Riassumere i dati

`len(df)`
number of rows in a dataframe

`df['column1'].nunique()`
numero di valori unici in una colonna

`df['column1'].sum()`
somma dei valori di una colonna

Riassumere i dati (cont)

`df['column1'].min()`
minimo dei valori di una colonna

`df['column1'].max()`
massimo dei valori di una colonna

`df['column1'].mean()`
media dei valori di una colonna

`df['column1'].var()`
varianza dei valori di una colonna

`df['column1'].std()`
deviazione standard dei valori di una colonna

`df['column1'].count()`
count dei valori non nulli di una colonna

`df['column1'].median()`
mediana dei valori di una colonna

`df['column1'].quantile([.25, .75])`
quantile dei valori di una colonna

`df['column1'].value_count()`
count di ogni valore di una colonna

Gestione oggetti di tipo datetime

`df['Data']=pd.to_datetime(df['Data'])`
converte stringhe in oggetti datetime

`df['Data'].dt.round('1s')`
arrotonda la data al secondo

`df['Duration']=(df['Data2']-df['Data1']).dt.seconds`
Calcola la differenza tra due date restituendo un risultato in secondi

`df['Data'].dt.date`
estrae solo la data da data ora

`df['Data'].dt.month`
estrae il mese

Gestione oggetti di tipo datetime (cont)

`df['Data'].dt.week`
estrae la settimana dell'anno

`df['Data'].dt.weekday`
estrae il giorno della settimana

`df['Data'].dt.weekday_name`
estrae il giorno della settimana (come nome)

`df['Data'].dt.time`
estrae l'ora completa

`df['Data'].dt.hour`
estrae l'ora

`df['Stagione']=pd.cut(df['Data'].dt.dayofyear + 11) % 366, [0, 91, 183, 275, 366], labels=['inverno', 'primavera', 'estate', 'autunno'])`
crea una colonna con la corrispondente stagione

Crea ed elimina colonne/righe

`df['new_column'] = df['col_1'] + df['col_2']`
crea una nuova colonna somma di due colonne

`df = df.drop('col_1', axis=1)`
elimina la colonna *col_1*

`df = df.drop(0, axis=0)`
elimina la riga con indice 0

