PANDAS
CHEATSHEET

SERIES E DATAFRAME

```
#Series
s = pd.Series([10, 20, 33, -65], index=['a', 'b', 'c', 'd'])
```

```
#Dataframe
data = {'FirstName': ['Bob', 'Ted', 'Mia'],
        'LastName': ['Smith', 'Hooks', 'Bann'],
        'Age': [12, 78, 30]}
df = pd.DataFrame(data, columns=['FirstName', 'LastName', 'Age'])
```

INFORMAZIONI SU SERIES/DATAFRAME

`df.shape`	Dimensione DataFrame
`df.head()`	Primi elementi DataFrame
`df.tail()`	Ultimi elementi DataFrame
`df.sample()`	Righe casuali
`df.max()`	Valori massimi
`df.min()`	Valori minimi
`df.mean()`	Valore medio
`df.median()`	Valore mediano
`df.sum()`	Somma dei valori
`df.cumsum()`	Somma dei valori cumulativa
`df.describe()`	Statistiche DataFrame
`df.columns`	Descrizione colonne DataFrame
`df.info()`	Informazioni sul DataFrame
`df.count()`	Numero di valori non nulli

INDICIZZAZIONE E ORDINAMENTO

`df.index`	Informazioni su indici
`df.index.values`	Valori indici
`df.set_index('FirstName', inplace=True)`	Impostazione indici
`df.reset_index(inplace=True)`	Reset indici
`df.sort_values(by='Age')`	Ordinamento crescente DataFrame secondo la caratteristica Age
`df.sort_values(by='Age',ascending=False)`	Ordinamento decrescente DataFrame secondo la caratteristica Age
`df.sort_values(by=['LastName','Age'] ,ascending=False)`	Ordinamento multiplo

SELEZIONE DATI

`df.loc['Bob']`	Selezione dei valori in base all'indice e alle label
`df.at['Bob','LastName']`	Selezione di un valore in base all'indice e alle label
`df.iloc[0]`	Selezione dei valori in base alla posizione
`df.iat[0,0]`	Selezione di un valore in base alla posizione

RAGGRUPPAMENTO DATI

`df.groupby('LastName')`	Raggruppamento in funzione della colonna
`df.groupby('LastName').agg(np.mean)`	Raggruppamento dati in funzione della colonna e media corrispondente

FUNZIONI

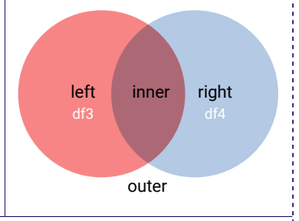
`df['Age'].apply(lambda x: x**2)`	Applica una funzione lungo un asse del DataFrame
`df.memory_usage(deep=True)`	Restituisce l'utilizzo della memoria di ogni colonna in byte
`df['Age'].value_counts()`	Restituisce il numero di occorrenze per colonna
`df.drop_duplicates(inplace=True)`	Eliminazione delle righe duplicate

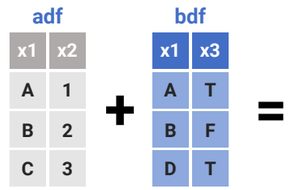
TRATTARE I DATI MANCANTI

`df.isnull()`	Controllo valori mancanti
`df.notnull()`	Controllo valori non nulli
`df.fillna(0)`	Sostituisce i dati nulli con un valore
`df.fillna(method='pad')`	Sostituisce i dati nulli con il valore precedente
`df.fillna(method='bfill')`	Sostituisce i dati nulli con il valore successivo
`df.replace(to_replace=np.nan, value=-99)`	Sostituisce i dati nulli con un valore
`df.interpolate(method='linear', limit_direction='forward')`	Sostituisce i dati con una interpolazione lineare
`df.dropna()`	Restituisce le righe che non presentano valori nulli
`df.dropna(axis=1)`	Restituisce le colonne che non presentano valori nulli

COMBINARE PIÙ DATAFRAME E SERIES

`df = pd.concat([df1,df2])`	Concatenazione di due DataFrame
`df = pd.concat([df1,df2], ignore_index=True)`	Concatenazione di due DataFrame ignorando gli indici
`df = pd.concat([df1,s], axis=1)`	Concatenazione di un DataFrame e una Series
`df1.append(df2)`	Aggiunta di un DataFrame ad un altro
`pd.merge(df1,df2)`	Merge di due DataFrame
`pd.merge(df1,df2,on='utente_id')`	Merge di due DataFrame in funzione di colonne in comune

`pd.merge(df3,df4,how='left')`
`pd.merge(df3,df4,how='right')`
`pd.merge(df3,df4,how='inner')`
`pd.merge(df3,df4,how='outer')`
	



Standard Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NaN

pd.merge(adf, bdf, how= 'left', on='x1')
Join matching rows from bdf to adf.

x1	x2	x3
A	1.0	T
B	2.0	F
D	NaN	T

pd.merge(adf, bdf, how= 'right', on='x1')
Join matching rows from adf to bdf.

x1	x2	x3
A	1	T
B	2	F

pd.merge(adf, bdf, how= 'inner', on='x1')
Join data. Retain only rows in both sets.

x1	x2	x3
A	1	T
B	2	F
C	3	NaN
D	NaN	T

pd.merge(adf, bdf, how= 'outer', on='x1')
Join data. Retain all values, all rows.

Filtering Joins

SERIE TEMPORALI

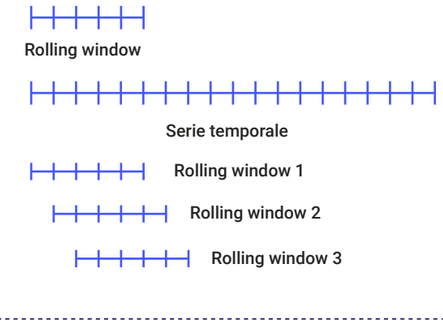
`# Conversione date da stringa in datetime`
`df['Date'] =`
`pd.to_datetime(df['Date'])`
`# Impostazione indice`
`df = df.set_index('Date')`
`# Selezione dati su intervallo temporale`
`df.loc['2017-08-10':'2017-08-20']`
	Selezione date da DataFrame in funzione di un intervallo temporale

`# Definizione range di date`
`index = pd.date_range('1/1/2000', periods=6, freq='T')`
`series = pd.Series(range(6), index=index)`
	`>>> 2000-01-01 00:00:00 0`
	`>>> 2000-01-01 00:01:00 1`
	`>>> 2000-01-01 00:02:00 2`
	`>>> 2000-01-01 00:03:00 3`
	`>>> 2000-01-01 00:04:00 4`
	`>>> 2000-01-01 00:05:00 5`

`# Upsample ogni 3 minuti` `series.resample('3T').sum()`	`>>> 2000-01-01 00:00:00 3` `>>> 2000-01-01 00:03:00 12`

`# Downsample ogni 30 secondi`
`series.resample('30S').sum())`
	`>>> 2000-01-01 00:00:00 0`
	`>>> 2000-01-01 00:00:30 0`
	`>>> 2000-01-01 00:01:00 1`
	`>>> 2000-01-01 00:01:30 0`
	`>>> 2000-01-01 00:02:00 2`
	`>>> 2000-01-01 00:02:30 0`
	`>>> 2000-01-01 00:03:00 3`
	`>>> 2000-01-01 00:03:30 0`
	`>>> 2000-01-01 00:04:00 4`
	`>>> 2000-01-01 00:04:30 0`
	`>>> 2000-01-01 00:05:00 5`

```
# funzione di rolling (media dei cinque valori nella finestra)
df.rolling(5, center=True).mean()
```



DATA MASTERS