Python3 Cheatsheet – Ermano Buikis

CLASS

```
class cibo:
  def __init__(self,carboidrati=0,proteine=0,grassi=0)
     self.proteine=proteine
     self.carboidrati=carboidrati
  def calcolacalorie(self):
     return(self.carboidrati *4 + self.proteine *4 + self.grassi *9)

pasta=cibo(proteine=12,carboidrati=32)
print(pasta.carboidrati)
cibo.calcolacalorie=calcolacalorie
```

FILE

```
file_handler=open('/Users\Ermano\Desktop\Python\seq.txt','a')
for i in list(range(0,10)):
   n=rnd.choice(list(diz.keys()))
   (diz[n])= diz[n]+1
   file_handler.write(str(rnd.choice(list(diz.keys()))))
```

NUMPY

```
a=np.array()
a.ndim
.shape
.stype
.itemsize
.size()
.shape(row,col)
.zeroes(row,col)
.flatten() #return array = 1 Dim

.random.rand(row,col)
.random.randint(max, (row,col))
.random.normal(mu, sigma)(row,col)
.random.seed(seed_value)

.dot(v1,v2) #prodotto scalare
.sum(axis=1) # somma sulle righe
.sum(axis=0) # somma sulle col
```

PANDAS

pd.info()	**info**
df.describe(include='all')	**summarize**
.colums	**columns**
df.values	**values**
.head(5)	**head**
.tail(5)	**tail**

.drop(nome_colonna,axis=0)	**elimina colonna**
.drop(nome_riga,axis=1)	**elimina riga**
.drop("Year", axis=1, inplace=True)	**drop**
.lock(label,index)	**lock**
.copy()	**copy**
len(df)	**len**

Read CSV

```
data = pd.read_csv('mydata.csv')
```

Initialize random dataframe

```
data = pd.DataFrame(np.random.rand(10,5), columns = list('abcde'))
data=pd.DataFrame(data=[f,m], index=['a','b'], columns=[3,4,5,6], dtype=None, copy=False)
```

Save in CSV

```
DF.to_csv("datasets/DF.csv")
```

Save specific columns in CSV

```
df[["COL3", "COL5"]].to_csv("datasets/df.csv", index=False)
```

Assegnazione

```
df[1][2]==3
```

Extract and filter

```
newDf= df[(df['Year']== 2015) & (df["Month"] == "February")]
```

Statistiche

```
std(), var(), mean(), median(), max(), min(), abs()
```

Trim values with thresholds

```
df.clip(lower=-10,upper=10)
```

Dropna

```
df.dropna()
df.dropna(inplace=True)
```

Drop

```
df.drop('nomeColonna')
```

Seleziona - LOC

```
.loc([nome colonna])
df.loc[:,'x2':'x4']  #Select all columns between x2 and x4
df.loc[df['a'] > 10, ['a','c']] #Select rows meeting logical condition,and only the specific columns (a,c)
df.loc[:, 'foo':'sat'] select all rows, and all colums between 'foo' and 'sat'
```

Seleziona - ILOC

```
.iloc([indiceRiga : indiceColonna])
df.iloc[:,[1,2,5]]          # Select columns in positions 1, 2 and 5 (first column is 0).
df.iloc[10:20]              #Select rows by position
df.iloc[1:5, 2:4]
```

Integer slicing

```
df.ix[:4]          #row
df.ix[:,'A']       #col
df['A']            #col
```

Select colums

```
df[['width','length','species']]
```

Filter

```
df.filter(state='Italia')
```

Drop duplicates

```
df.drop_duplicates()
```

Resample

```
df.sample(frac=0.5)
```

Column types

```
df.dtypes
```

Check empty fields

```
df.empty           # return True if obj is empty
```

Check dimenstion dataframe

```
df.ndim
df.size            size  = df (n rows * n col)
df.shape           shape = tuple (r,c)
```

Value counts

```
df['w'].value_counts()      Count values frequencies with each unique value of variable
```

Filtering

```
df_new1= df.copy()[df['nomeColonna1':'nomeColonna2']]

df_new1= df.copy()[df['nomeColonna1']=='Italia']
```

DateTime format column

```
df['Date']=pd.datetime(df['Date'])
```

Set Index

```
df_new2=df_new.set_index('NomeColonna')          prende la colonna come indice
```

Lambda function

```
df['date'].assign(a=lambda df: df.a / 2)
```

Sort Index

```
df.sort_index(inplace=True)
```

Sort Values

```
df.sort_values(by='nomeColonna1',ascending=True,inplace=True )
```

Join

```
df = df.join(df_new1['nomeCol'])
```

Rename

```
df.rename(columns={"county": "County", "st": "State"}, inplace=True)
```

DataFrame to List

```
list_nome_colonna=list(set(df['nomeColonna'].values.tolist()))
```

DataFrame to Dictionary

```
dic = df.to_dict()

state_abbv_dict = state_abbv.to_dict()['Postal Code']
```

Map function

```
df1['State'] = df1['State'].map(df2)
```

Sostituire valori nominali con numeri

```
color_dict = {"J": 1,"I": 2,"H": 3,"G": 4,"F": 5,"E": 6,"D": 7}
df['color'] = df['color'].map(color_dict)
```

Group By

```
df2 = df.groupby("State")
```

Get Group + Set Index

```
df2.get_group("Alabama").set_index("Year").head()
```

Individuare NaN

```
issue_df = df[df['NomeColonna']==0]
```

Unique

```
issue_df['State'].unique()
```

Replace Nan

```
act_min_wage.replace(0, np.NaN).dropna(axis=1).corr().head()
```

axis 1 == columns. 0 is default,0 is for rows

Correlation

```
df [[ 'Col1' , 'Col2' ]].corr()
```

Covariance

```
df [[ 'Col1' , 'Col2' ]].cov()
```

Trasformo i dati di una colonna in colonne diverse secondo le categorie della colonna

```
g_df=pd.DataFrame()
for col in df['nomeColonna'].unique():
  new_df = df.copy()[df['NomeColonna']==col] # crea nuovo df
  new_df = new_df.set_index("date", inplace=True)
  new_df = new_df.sort_index(inplace=True)
  new_df = new_df.join(df['{}_colonna'.format(str(col))])
```

Heat Map Correlation

```
corr=df.corr().head()
import matplotlib.pyplot as plt
plt.matshow(corr)
plt.show()
```

Replace

```
df1 = df1.replace(0, np.NaN).dropna(axis=1)
```

Arange

```
np.arange(start,stop)
```

Linspace

```
np.linspace(start,stop,size)
```

Plot

df.plot.hist()	Histogram
df.plot.scatter(x='w',y='h')	Scatter
plt.scatter(x,y,c='green',s=100)	
df.rolling(n)	Rolling
pd.merge(ydf, zdf)	Merge

MATPLOTLIB

```
import matplotlib.plotlib as plt
```

LOAD CSV

```
with open('example.txt','r') as csvfile:
    plots = csv.reader(csvfile, delimiter=',')
```

LOAD TXT

```
x, y = np.loadtxt('example.txt', delimiter=',', unpack=True)
plt.plot(x,y, label='Loaded from file!')
```

HIST

```
bins = [0,10,20] # sono da dove partono le barre
plt.hist(population_ages, bins, histtype='bar', rwidth=0.8)
```

SCATTER

```
plt.scatter(x,y, label='skitscat', color='k', s=25, marker="o")
```

STACK PLOT

```
days = [1,2,3,4,5] #X
sleeping = [7,8,6,11,7] #Y0
eating =   [2,3,4,3,2] #Y1
working =  [7,8,7,2,2] #Y2
playing =  [8,5,7,8,13] #Y3
plt.plot([],[],color='m', label='Sleeping', linewidth=5) # legend features
plt.plot([],[],color='c', label='Eating', linewidth=5)
plt.plot([],[],color='r', label='Working', linewidth=5)
plt.plot([],[],color='k', label='Playing', linewidth=5)
```

```
plt.stackplot(days, sleeping,eating,working,playing, colors=['m','c','r','k'])
plt.legend()
```

PIE CHART

```
slices = [7,2,2,13]
activities = ['sleeping','eating','working','playing']
cols = ['c','m','r','b']
plt.pie(slices,
        labels=activities,
        colors=cols,
        startangle=90,
        shadow= True,
        explode=(0,0.1,0,0), # Explode: If we wanted to pull out the first slice a bit, we would do 0.1,0,0,0.
        autopct='%1.1f%%')
fig = plt.figure(figsize=plt.figaspect(2.0))
fig.add_axes()
```

legend

```
plt.legend()
```

savefig

```
plt.savefig(path,nomefile)
```

parametri immagine

```
plt.rcparams['figure.figuresize']=[12,8]
```

legge un immagine

```
img=mpimg.imread(path)
```

ottimizza dimensione

```
plt.tight.layout()
```

styles

```
from matplotlib import style
print(plt.style.available)
style.use('dark_background')
plt.style.aviable
plt.style.use("nome stile")
```

Caratteristiche immagine

```
img.shape
img.imshow(img,cmap=)
img.xtricks([])                 Tricks
img.xlim()                      Limits
img.grid()                      Grid
```

Image annotation

```
img.Annotate("string", xy=(num,num), xytest=("string"))

ax1.annotate('Bad News!',(date[9],highp[9]),
             xytext=(0.8, 0.9), textcoords='axes fraction',
```

arrowprops = dict(facecolor='grey',color='grey'))

Text insertion

img.text((x,y),"text", size)

Fill immagine

ax1.fill_between(date, 0, closep)

ax1.fill_between(date, closep, closep[0],where=(closep > closep[0]), facecolor='g', alpha=0.5)

ax1.fill_between(date, closep, closep[0],where=(closep < closep[0]), facecolor='r', alpha=0.5)

Griglia immagine

ax1.grid(True)#, color='g', linestyle='-', linewidth=5)

SEABORN

import seaborn as sns

Instogramma sovrapposto

```
data = np.random.multivariate_normal([0, 0], [[5, 2], [2, 2]], size=2000)
data = pd.DataFrame(data, columns=['x', 'y'])
for col in 'xy':
    plt.hist(data[col], normed=True, alpha=0.5)
```

Gaussiana sovrapposta

```
for col in 'xy':
    sns.kdeplot(data[col], shade=True)
grafico=sns.lmplot(data, x='nome_col_x',y=)          linear model

sns.distplot(data['colname'],bins=numbins)           instogramma
sns.boxplot(data,x,y)

sns.distplot(data['x'])                              applico la gaussiana all'instogramma
sns.distplot(data['y']);
```

LOAD datasets

tips = sns.load_dataset('tips')

Set Axis style

```
with sns.axes_style(style='ticks'):
    g = sns.factorplot("x", "y", "sex", data=tips, kind="box")
    g.set_axis_labels("x", "y");
```

Violin Plot

```
men = (data.gender == 'M')
women = (data.gender == 'W')
with sns.axes_style(style=None):
    sns.violinplot("age_dec", "split_frac", hue="gender", data=data,
                   split=True, inner="quartile",
                   palette=["lightblue", "lightpink"]);
```