PYTHON TIPS AND TRICKS

01- How to find python version

In [ ]:
```
import pandas as pd
pd.__version__
```

Out[ ]:
```
'1.3.4'
```

In [ ]:
```
pd.show_versions()
```

```
INSTALLED VERSIONS
------------------
commit           : 945c9ed766a61c7d2c0a7cbb251b6edebf9cb7d5
python           : 3.9.7.final.0
python-bits      : 64
OS               : Windows
OS-release       : 10
Version          : 10.0.19044
machine          : AMD64
processor        : Intel64 Family 6 Model 58 Stepping 9, GenuineIntel
byteorder        : little
LC_ALL           : None
LANG             : None
LOCALE           : English_Pakistan.1252

pandas           : 1.3.4
numpy            : 1.20.3
pytz             : 2021.3
dateutil         : 2.8.2
pip              : 21.2.4
setuptools       : 58.0.4
Cython           : 0.29.24
pytest           : 6.2.4
hypothesis       : None
sphinx           : 4.2.0
blosc            : None
feather          : None
xlsxwriter       : 3.0.1
lxml.etree       : 4.6.3
html5lib         : 1.1
pymysql          : None
psycopg2         : None
jinja2           : 2.11.3
IPython          : 7.29.0
pandas_datareader: 0.10.0
bs4              : 4.10.0
bottleneck       : 1.3.2
fsspec           : 2021.10.1
fastparquet      : None
gcsfs            : None
matplotlib       : 3.4.3
numexpr          : 2.7.3
odfpy            : None
openpyxl         : 3.0.9
pandas_gbq       : None
pyarrow          : 7.0.0
pyxlsb           : None
```

```
s3fs            : None
scipy           : 1.7.1
sqlalchemy      : 1.4.22
tables          : 3.6.1
tabulate        : 0.8.9
xarray          : 2022.3.0
xlrd            : 2.0.1
xlwt            : 1.3.0
numba           : 0.54.1
```

02- Make a dataframe

In [ ]:
```
df= pd.DataFrame({'A column': [1, 2, 3], 'B column': [4, 5, 6]})
df
```

Out[ ]:

	A column	B column
0	1	4
1	2	5
2	3	6

In [ ]:
```
# Create a dataframe with numpy array
import numpy as np
arr= np.array([[1, 2, 3], [4, 5, 6]])
arr
```

Out[ ]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [ ]:
```
pd.DataFrame(arr)
```

Out[ ]:

	0	1	2
0	1	2	3
1	4	5	6

In [ ]:
```
pd.DataFrame(np.random.rand(5, 3))
```

Out[ ]:

	0	1	2
0	0.961410	0.692550	0.252524
1	0.443909	0.219072	0.381439
2	0.747727	0.913689	0.506950
3	0.775276	0.021673	0.334458
4	0.947448	0.028839	0.419858

In [ ]:
```
pd.DataFrame(np.random.rand(5, 3),columns=['A', 'B', 'C'])
```

Out[ ]:

	A	B	C

	A	B	C
0	0.508250	0.933340	0.071358
1	0.403564	0.657557	0.406338
2	0.555639	0.713242	0.088102
3	0.858696	0.248986	0.979215
4	0.963242	0.153747	0.346719

In [ ]:
```
bigdf= pd.DataFrame(np.random.rand(30, 3),columns=['A', 'B', 'C'])
bigdf.head()
```

Out[ ]:

	A	B	C
0	0.068368	0.207223	0.911486
1	0.830283	0.759223	0.319438
2	0.108546	0.945356	0.277705
3	0.651413	0.144023	0.893426
4	0.825834	0.293766	0.565454

03- How to rename columns

In [ ]:
```
df = bigdf.rename(columns={'A': 'a', 'B': 'b', 'C': 'c'})
df.head()
```

Out[ ]:

	a	b	c
0	0.068368	0.207223	0.911486
1	0.830283	0.759223	0.319438
2	0.108546	0.945356	0.277705
3	0.651413	0.144023	0.893426
4	0.825834	0.293766	0.565454

In [ ]:
```
df.columns= ['aa', 'bb', 'cc']
df.head()
```

Out[ ]:

	aa	bb	cc
0	0.068368	0.207223	0.911486
1	0.830283	0.759223	0.319438
2	0.108546	0.945356	0.277705
3	0.651413	0.144023	0.893426
4	0.825834	0.293766	0.565454

In [ ]:
```
df.columns=df.columns.str.replace('a', 'A')
```

```
df.head()
```

Out[ ]:

	AA	bb	cc
0	0.068368	0.207223	0.911486
1	0.830283	0.759223	0.319438
2	0.108546	0.945356	0.277705
3	0.651413	0.144023	0.893426
4	0.825834	0.293766	0.565454

In [ ]:

```
# add prefix in column names
df = df.add_prefix('col_')
df.head()
```

Out[ ]:

	col_AA	col_bb	col_cc
0	0.068368	0.207223	0.911486
1	0.830283	0.759223	0.319438
2	0.108546	0.945356	0.277705
3	0.651413	0.144023	0.893426
4	0.825834	0.293766	0.565454

In [ ]:

```
# add suffix in column names
df = df.add_suffix('col_')
df.head()
```

Out[ ]:

	col_AAcol_	col_bbcol_	col_cccol_
0	0.068368	0.207223	0.911486
1	0.830283	0.759223	0.319438
2	0.108546	0.945356	0.277705
3	0.651413	0.144023	0.893426
4	0.825834	0.293766	0.565454

04- Using template data

In [ ]:

```
import seaborn as sns
kashti = sns.load_dataset('titanic')
kashti.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	decl
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	decl
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN

In [ ]:
```
#summary of data
df.describe()
```

Out[ ]:

	col_AAcol_	col_bbcol_	col_cccol_
count	30.000000	30.000000	30.000000
mean	0.511110	0.439279	0.503229
std	0.310530	0.249940	0.290560
min	0.012941	0.061508	0.017971
25%	0.184353	0.197967	0.284311
50%	0.623794	0.468020	0.505368
75%	0.742045	0.607483	0.668631
max	0.940488	0.946832	0.989461

In [ ]:
```
#colums and rows
df.shape
```

Out[ ]:
```
(30, 3)
```

In [ ]:
```
#column names
df.columns
```

Out[ ]:
```
Index(['col_AAcol_', 'col_bbcol_', 'col_cccol_'], dtype='object')
```

In [ ]:
```
# save dataframe to csv
df.to_csv('titanic.csv')
```

In [ ]:
```
# save dataframe to excel
df.to_excel('titanic.xlsx')
```

05- Using your own data

In [ ]:
```
df = pd.read_csv('titanic.csv')
df.head()
```

Out[ ]:

	Unnamed: 0	col_AAcol_	col_bbcol_	col_cccol_
0	0	0.068368	0.207223	0.911486
1	1	0.830283	0.759223	0.319438
2	2	0.108546	0.945356	0.277705
3	3	0.651413	0.144023	0.893426

	Unnamed: 0	col_AAcol_	col_bbcol_	col_cccol_
4	4	0.825834	0.293766	0.565454

In [ ]:
```
df= pd.read_excel('titanic.xlsx')
df.head()
```

Out[ ]:

	Unnamed: 0	col_AAcol_	col_bbcol_	col_cccol_
0	0	0.068368	0.207223	0.911486
1	1	0.830283	0.759223	0.319438
2	2	0.108546	0.945356	0.277705
3	3	0.651413	0.144023	0.893426
4	4	0.825834	0.293766	0.565454

06- Reversing a row order

In [ ]:
```
df= sns.load_dataset('titanic')
df.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	decl
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN

In [ ]:
```
df.loc[::-1].head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	d
890	0	3	male	32.0	0	0	7.75	Q	Third	man	True	N
889	1	1	male	26.0	0	0	30.00	C	First	man	True	
888	0	3	female	NaN	1	2	23.45	S	Third	woman	False	N
887	1	1	female	19.0	0	0	30.00	S	First	woman	False	
886	0	2	male	27.0	0	0	13.00	S	Second	man	True	N

08- Select a column by dtype

In [ ]:
```
df.dtypes
```

Out[ ]:
```
survived          int64
pclass            int64
```

```
sex              object
age             float64
sibsp             int64
parch             int64
fare            float64
embarked         object
class          category
who              object
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
dtype: object
```

In [ ]:
```
#only select numeric columns
df.select_dtypes(include=['number']).head()
```

Out[ ]:

	survived	pclass	age	sibsp	parch	fare
0	0	3	22.0	1	0	7.2500
1	1	1	38.0	1	0	71.2833
2	1	3	26.0	0	0	7.9250
3	1	1	35.0	1	0	53.1000
4	0	3	35.0	0	0	8.0500

In [ ]:
```
#only select numeric object ctegory columns
df.select_dtypes(include=['number','category','object']).head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	deck	embark_to
0	0	3	male	22.0	1	0	7.2500	S	Third	man	NaN	Southamp
1	1	1	female	38.0	1	0	71.2833	C	First	woman	C	Cherbo
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	NaN	Southamp
3	1	1	female	35.0	1	0	53.1000	S	First	woman	C	Southamp
4	0	3	male	35.0	0	0	8.0500	S	Third	man	NaN	Southamp

09- Covert strings to number

In [ ]:
```
df= pd.DataFrame({'col_A':['1','2','3','4','5'],'col_B':[6,7,8,9,10],'col_C':[11,12,
df.head()
```

Out[ ]:

	col_A	col_B	col_C
0	1	6	11
1	2	7	12
2	3	8	13
3	4	9	14

	col_A	col_B	col_C
4	5	10	15

In [ ]:
```
df.dtypes
```

Out[ ]:
```
col_A    object
col_B     int64
col_C     int64
dtype: object
```

In [ ]:
```
df.astype({'col_A':'int64','col_B':'float'}).dtypes
```

Out[ ]:
```
col_A      int64
col_B    float64
col_C      int64
dtype: object
```

10- Reduce dataframe size

In [ ]:
```
df = sns.load_dataset('titanic')
df.shape
```

Out[ ]:
```
(891, 15)
```

In [ ]:
```
df.sample(frac=0.1).shape
```

Out[ ]:
```
(89, 15)
```

In [ ]:
```
df.sample(frac=0.5).shape
```

Out[ ]:
```
(446, 15)
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     891 non-null    int64   
 1   pclass       891 non-null    int64   
 2   sex          891 non-null    object  
 3   age          714 non-null    float64 
 4   sibsp        891 non-null    int64   
 5   parch        891 non-null    int64   
 6   fare         891 non-null    float64 
 7   embarked     889 non-null    object  
 8   class        891 non-null    category
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool    
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object  
 13  alive        891 non-null    object  
 14  alone        891 non-null    bool    
```

```
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```