IOTA ACADEMY
Become Job Ready...

Operations

There are lots of operations with pandas that will be really useful to you, but don't fall into any distinct category. Let's show them here in this lecture:

In [252…
```python
import pandas as pd
import numpy as np
```

1) Viewing data

In [253…
```python
titanic = pd.read_csv("train.csv")
```

head()

Return the first `n` rows.

- **Syntax:** DataFrame.head(n: 'int' = 5)

In [254…
```python
titanic.head()
```

Out[254]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

tail()

Return the last `n` rows.

- **Syntax:** dataFrame.tail(n: 'int' = 5)

In [255…
```python
titanic.tail()
```

Out[255]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

sample()

Return a random sample of items from an axis of object.

- **Syntax:** DataFrame.sample(n=None, frac=None, replace=False, weights=None, random_state=None, axis=None, ignore_index=False)

In [256…
```python
titanic.sample()
```

Out[256]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
834	835	0	3	Allum, Mr. Owen George	male	18.0	0	0	2223	8.3	NaN	S

In [257…
```python
# We can choose a random sample size
titanic.sample(3)
```

Out[257]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
443	444	1	2	Reynaldo, Ms. Encarnacion	female	28.00	0	0	230434	13.0000	NaN	S
803	804	1	3	Thomas, Master. Assad Alexander	male	0.42	0	1	2625	8.5167	NaN	C
324	325	0	3	Sage, Mr. George John Jr	male	NaN	8	2	CA. 2343	69.5500	NaN	S

2) DataFrame Attributes

a) shape

Return the dimensionality of the DataFrame in the form of a tuple.

In [258…
```python
titanic.shape  # 891 rows and 12 columns
```

Out[258]:
```
(891, 12)
```

In [259…
```python
titanic.shape[0] # rows
```

Out[259]:
```
891
```

b) size

Return the number of elements in the DataFrame.

In [260…
```python
titanic.size
```

Out[260]:
```
10692
```

c) dtypes

Return the dtypes of each column in the DataFrame

In [261…
```python
titanic.dtypes
```

Out[261]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

d) columns

This attribute is used to fetch the label values for columns present in a particular data frame.

In [262…
```python
titanic.columns
```

Out[262]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

e) index

the index attribute is used to display the row labels of a data frame object.

In [263…
```python
titanic.index
```

Out[263]:
```
RangeIndex(start=0, stop=891, step=1)
```

f) empty

check whether the data frame is empty or not. This attribute returns true if the data frame is empty and false if the DataFrame is not empty.

In [264…
```python
titanic.empty
```

Out[264]:
```
False
```

g) values

This attribute is used to represent the values/data of dataframe in NumPy array form.

In [265…
```python
titanic.values  # The return value is a 2-dimensional array with one array for each row.
```

Out[265]:
```
array([[1, 0, 3, ..., 7.25, nan, 'S'],
       [2, 1, 1, ..., 71.2833, 'C85', 'C'],
       [3, 1, 3, ..., 7.925, nan, 'S'],
       ...,
       [889, 0, 3, ..., 23.45, nan, 'S'],
       [890, 1, 1, ..., 30.0, 'C148', 'C'],
       [891, 0, 3, ..., 7.75, nan, 'Q']], dtype=object)
```

3) DataFrame methods

`info()`

Print a concise summary of a DataFrame.

This method prints information about a DataFrame including **the index dtype and columns, non-null values and memory usage.**

In [266…
```python
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

`describe()`

Generate descriptive statistics.

Descriptive statistics include those that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding `NaN` values.

In [267…
```python
titanic.describe()
```

Out[267]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

`unique()`

Return unique values

In [268…
```python
titanic["Embarked"].unique()
```

Out[268]:
```
array(['S', 'C', 'Q', nan], dtype=object)
```

In [269…
```python
# Example
titanic["Sex"].unique()
```

Out[269]:
```
array(['male', 'female'], dtype=object)
```

`nunique()`

Count number of distinct elements in specified axis

In [270…
```python
titanic.nunique()  # on whole dataframe
```

Out[270]:
```
PassengerId    891
Survived         2
Pclass           3
Name           891
Sex              2
Age             88
SibSp            7
Parch            7
Ticket         681
Fare           248
Cabin          147
Embarked         3
dtype: int64
```

`nlargest()`

Return the first `n` rows with the largest values in `columns`, in descending order

In [271…
```python
titanic["Fare"].nlargest()
```

Out[271]:
```
258    512.3292
679    512.3292
737    512.3292
27     263.0000
88     263.0000
Name: Fare, dtype: float64
```

In [272…
```python
# Example
titanic["Age"].nlargest(6) # How many largest values we want can be specified.
```

Out[272]:
```
630    80.0
851    74.0
96     71.0
493    71.0
116    70.5
672    70.0
Name: Age, dtype: float64
```

`nsmallest()`

Return the first `n` rows with the smallest values in `columns`, in ascending order.

In [273…
```python
titanic["Age"].nsmallest()
```

Out[273]:
```
803    0.42
755    0.67
469    0.75
644    0.75
78     0.83
Name: Age, dtype: float64
```

`value_counts()`

Return a Series containing counts of unique values.

In [274…
```python
titanic["Sex"].value_counts()
```

Out[274]:
```
male      577
female    314
Name: Sex, dtype: int64
```

In [275…
```python
# Example
titanic["Pclass"].value_counts()
```

Out[275]:
```
3    491
1    216
2    184
Name: Pclass, dtype: int64
```

In [276…
```python
# Example
titanic[["Sex","Pclass"]].value_counts()
```

Out[276]:
```
Sex     Pclass
male    3         347
female  3         144
male    1         122
        2         108
female  1          94
        2          76
dtype: int64
```

In [277…
```python
titanic["Age"].value_counts(bins=4)
# Bins can be useful for going from a continuous variable to a categorical variable; instead of counting unique
```

Out[277]:
```
(20.315, 40.21]    385
(0.339, 20.315]    179
(40.21, 60.105]    128
(60.105, 80.0]      22
Name: Age, dtype: int64
```

`sort_values()`

The sort_values() method sorts the DataFrame by the specified label.

In [278…
```python
titanic.sort_values(by="Sex")  # inplace=False by default
```

Out[278]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
383	384	1	1	Holverson, Mrs. Alexander Oskar (Mary Aline To...	female	35.0	1	0	113789	52.0000	NaN	S
218	219	1	1	Bazzani, Miss. Albina	female	32.0	0	0	11813	76.2917	D15	C
609	610	1	1	Shutes, Miss. Elizabeth W	female	40.0	0	0	PC 17582	153.4625	C125	S
216	217	1	3	Honkanen, Miss. Eliina	female	27.0	0	0	STON/O2. 3101283	7.9250	NaN	S
215	216	1	1	Newell, Miss. Madeleine	female	31.0	1	0	35273	113.2750	D36	C
...	...	...	...	...	...	...	...	...	...	...	...	...
371	372	0	3	Wiklund, Mr. Jakob Alfred	male	18.0	1	0	3101267	6.4958	NaN	S
372	373	0	3	Beavan, Mr. William Thomas	male	19.0	0	0	323951	8.0500	NaN	S
373	374	0	1	Ringhini, Mr. Sante	male	22.0	0	0	PC 17760	135.6333	NaN	C
360	361	0	3	Skoog, Mr. Wilhelm	male	40.0	1	4	347088	27.9000	NaN	S
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [279…
```python
titanic.sort_values(by="Fare",ascending=False) # Now, It will sort in descending order by fare.
```

Out[279]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
258	259	1	1	Ward, Miss. Anna	female	35.0	0	0	PC 17755	512.3292	NaN	C
737	738	1	1	Lesurer, Mr. Gustave J	male	35.0	0	0	PC 17755	512.3292	B101	C
679	680	1	1	Cardeza, Mr. Thomas Drake Martinez	male	36.0	0	1	PC 17755	512.3292	B51 B53 B55	C
88	89	1	1	Fortune, Miss. Mabel Helen	female	23.0	3	2	19950	263.0000	C23 C25 C27	S
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000	C23 C25 C27	S
...	...	...	...	...	...	...	...	...	...	...	...	...
633	634	0	1	Parr, Mr. William Henry Marsh	male	NaN	0	0	112052	0.0000	NaN	S
413	414	0	2	Cunningham, Mr. Alfred Fleming	male	NaN	0	0	239853	0.0000	NaN	S
822	823	0	1	Reuchlin, Jonkheer. John George	male	38.0	0	0	19972	0.0000	NaN	S
732	733	0	2	Knight, Mr. Robert J	male	NaN	0	0	239855	0.0000	NaN	S
674	675	0	2	Watson, Mr. Ennis Hastings	male	NaN	0	0	239856	0.0000	NaN	S

891 rows × 12 columns

Sort by multiple columns

In [280…
```python
titanic.sort_values(by=["Sex","Fare"],ascending=False)
```

Out[280]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
679	680	1	1	Cardeza, Mr. Thomas Drake Martinez	male	36.0	0	1	PC 17755	512.3292	B51 B53 B55	C
737	738	1	1	Lesurer, Mr. Gustave J	male	35.0	0	0	PC 17755	512.3292	B101	C
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000	C23 C25 C27	S
438	439	0	1	Fortune, Mr. Mark	male	64.0	1	4	19950	263.0000	C23 C25 C27	S
118	119	0	1	Baxter, Mr. Quigg Edmond	male	24.0	0	1	PC 17558	247.5208	B58 B60	C
...	...	...	...	...	...	...	...	...	...	...	...	...
367	368	1	3	Moussa, Mrs. (Mantoura Boulos)	female	NaN	0	0	2626	7.2292	NaN	C
780	781	1	3	Ayoub, Miss. Banoura	female	13.0	0	0	2687	7.2292	NaN	C
19	20	1	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.2250	NaN	C
875	876	1	3	Najib, Miss. Adele Kiamie "Jane"	female	15.0	0	0	2667	7.2250	NaN	C
654	655	0	3	Hegarty, Miss. Hanora "Nora"	female	18.0	0	0	365226	6.7500	NaN	Q

891 rows × 12 columns

`duplicated()`

Return boolean Series denoting duplicate rows.

In [281…
```python
# Create a DataFrame
dupt_data = pd.DataFrame({
  "name": ["Sally", "Mary", "John", "Mary"],
  "age": [50, 40, 30, 40]
})
```

In [282…
```python
dupt_data.duplicated()
```

Out[282]:
```
0    False
1    False
2    False
3     True
dtype: bool
```

In [283…
```python
# select rows
dupt_data[dupt_data.duplicated()]  # There are one duplicate row in this table.
```

Out[283]:

	name	age
3	Mary	40

`drop_duplicates()`

Return DataFrame with duplicate rows removed

In [284…
```python
dupt_data
```

Out[284]:

	name	age
0	Sally	50
1	Mary	40
2	John	30
3	Mary	40

In [285…
```python
dupt_data.drop_duplicates()
```

Out[285]:

	name	age
0	Sally	50
1	Mary	40
2	John	30

`apply()`

The apply() method allows you to apply a function along one of the axis of the DataFrame

In [286…
```python
apply_data = pd.DataFrame({"Name":['Vinod', 'Mayur', 'Bhola', 'Chinu'],
                           "Height_in_cm":[180, 175, 168, 190]}
                          )

apply_data
```

Out[286]:

	Name	Height_in_cm
0	Vinod	180
1	Mayur	175
2	Bhola	168
3	Chinu	190

In [287…
```python
# We shall convert the students' heights in cm to feet with two decimal places.

# Create a function

def cm_to_feet(h):
    return np.round(h/30.48, 2)
```

In [288…
```python
apply_data["Height_in_cm"].apply(cm_to_feet)
```

Out[288]:
```
0    5.91
1    5.74
2    5.51
3    6.23
Name: Height_in_cm, dtype: float64
```

In [289…
```python
# Create new column
apply_data["Height_in_feet"] = app_data["Height_in_cm"].apply(cm_to_feet)
apply_data
```

Out[289]:

	Name	Height_in_cm	Height_in_feet
0	Vinod	180	5.91
1	Mayur	175	5.74
2	Bhola	168	5.51
3	Chinu	190	6.23

In [290…
```python
# we can do same with lambda function
apply_data["Height_in_feet_by_lambda"] = apply_data["Height_in_cm"].apply(lambda x: np.round(x/30.48, 2))
apply_data
```

Out[290]:

	Name	Height_in_cm	Height_in_feet	Height_in_feet_by_lambda
0	Vinod	180	5.91	5.91
1	Mayur	175	5.74	5.74
2	Bhola	168	5.51	5.51
3	Chinu	190	6.23	6.23

`astype()`

The apply() method allows you to apply a function along one of the axis of the DataFrame

In [291…
```python
# Create a DataFrame
tech = pd.DataFrame({
    "Items":["Apple","Nokia","Samsung"],
    'Price' :["20000","25000","26000"],
    'Discount':["1000","2300","1500"]})

tech
```

Out[291]:

	Items	Price	Discount
0	Apple	20000	1000
1	Nokia	25000	2300
2	Samsung	26000	1500

In [292…
```python
# check datatype
tech.dtypes
```

Out[292]:
```
Items       object
Price       object
Discount    object
dtype: object
```

In [293…
```python
# change the Price and Discount columns' datatype from object to int

tech["Price"] = tech["Price"].astype("int")
tech["Discount"] = tech["Discount"].astype("int")
```

In [294…
```python
tech.dtypes  # data type has been changed
```

Out[294]:
```
Items       object
Price        int32
Discount     int32
dtype: object
```

`rank()`

Compute numerical data ranks (1 through n) along axis.

By default, equal values are assigned a rank that is the average of the ranks of those values.

In [295…
```python
# Create a DataFrame
df_rank = pd.DataFrame(data={'Days':["Mon","Tue","Wed","Thurs","Fri","Satu","Sun"],
                        'Temperature': [36,34,40,31,29,28,25],
                        })
df_rank["rank_1"] = df_rank["Temperature"].rank()  # it gives first rank to lowest
df_rank["rank_2"] = df_rank["Temperature"].rank(ascending=False)  # it gives first rank to highest
df_rank
```

Out[295]:

	Days	Temperature	rank_1	rank_2
0	Mon	36	6.0	2.0
1	Tue	34	5.0	3.0
2	Wed	40	7.0	1.0
3	Thurs	31	4.0	4.0
4	Fri	29	3.0	5.0
5	Satu	28	2.0	6.0
6	Sun	25	1.0	7.0

4) Pivot tables

`pivot_table()`

Create a spreadsheet-style pivot table as a DataFrame.

- **Syntax:** pd.pivot_table(data, values=None, index=None, columns=None, aggfunc='mean', fill_value=None, margins=False, dropna=True, margins_name='All', observed=False, sort=True)

In [296…
```python
# read the titanic dataset
titanic.head()
```

Out[296]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [297…
```python
pd.pivot_table(titanic,values='Survived', index='Sex', columns='Pclass') # default aaggfunc: 'AggFuncType' = 'm
```

Out[297]:

Pclass	1	2	3
Sex			
female	0.968085	0.921053	0.500000
male	0.368852	0.157407	0.135447

In [298…
```python
# It produces a summary of the total fare paid by men and women in each class.
pd.pivot_table(titanic,values='Fare', index='Sex', columns='Pclass',aggfunc="sum")
```

Out[298]:

Pclass	1	2	3
Sex			
female	9975.8250	1669.7292	2321.1086
male	8201.5875	2132.1125	4393.5865

In [299…
```python
# count the number of male and female
pd.pivot_table(titanic,values='PassengerId', index='Sex',aggfunc="count")
```

Out[299]:

	PassengerId
Sex	
female	314
male	577

5. work with text data

There are two ways to store text data in pandas:

`object` - dtype NumPy array.

`StringDtype` - extension type.

In [300…
```python
# Create a DataFrame
text_df = pd.DataFrame({
    'name': ['alice smith','jones','jones','daisy white'],
    'age': [25,20,30,35]
})

print(text_df)

print(text_df["name"].dtype)
```

```
          name  age
0  alice smith   25
1        jones   20
2        jones   30
3  daisy white   35
object
```

In [301…
```python
# change column dtype object to string
text_df["name"] = text_df["name"].astype("string")
print(text_df["name"].dtype)
```

```
string
```

String handling

`Series.str` can be used to access the values of the series as strings and apply several methods to it

Vectorized string functions for Series and Index.

In [302…
```python
# create a DataFrame
text_df_2 = pd.DataFrame({'name': ['John Paul Smith', 'Brian David Jones', 'Harry William Roberts'],
                  'website': ['https://www.johnsmith.com', 'https://www.brianjones.com', 'https://www.harryrob
                  'telephone': ['(01234) 5678910', '(05432) 9876543', '(09876) 5432109'],
                   'username': ['john_smith', 'brian_jones', 'harry_roberts']})

text_df_2
```

Out[302]:

	name	website	telephone	username
0	John Paul Smith	https://www.johnsmith.com	(01234) 5678910	john_smith
1	Brian David Jones	https://www.brianjones.com	(05432) 9876543	brian_jones
2	Harry William Roberts	https://www.harryroberts.com	(09876) 5432109	harry_roberts

In [303…
```python
# Extarct First name
# Before using any string method directly on a column, use the .str  method.
# Elements in the split lists can be accessed using get or .str[] notation:

print(text_df_2["name"].str.split().str[0])
print(text_df_2["name"].str.split().str.get(0))
```

```
0     John
1    Brian
2    Harry
Name: name, dtype: object
0     John
1    Brian
2    Harry
Name: name, dtype: object
```

In [304…
```python
# example
text_df_2["name"].str.upper()
```

Out[304]:
```
0          JOHN PAUL SMITH
1        BRIAN DAVID JONES
2    HARRY WILLIAM ROBERTS
Name: name, dtype: object
```

For more details follow this link

https://pandas.pydata.org/docs/user_guide/text.html

Great Job!