Open Source packages for DS: Pandas

(Python data analysis library)

Nazgul Rakhimzhanova

Vladlen Chsheglov

Ersain Chinibayev

IITU 2020

OUTLINE

- **Previously**
- **Data variable** types
- **Manipulating** dataframes with pandas
- **Readings**

PREVIUOSLY

- Discussed about **DS purpose**
- Did overview of the course **NumPy package**
- Practiced NumPy functions and objectes

What do you remember about each activity?

COURSE SCHEDULE

week	Mid Term (weeks 01-07)	End Term (weeks 08-14)	week
01	Intro: Data Science Area and open source tools for Data Science	Statistics: Distribution – Lognormal, Exponential	08
02	NumPy package for data science	Sampling and Estimation	09
03	Pandas package for data science	Visualization II	10
04	Visualization with matplotlib	Correlation and Covariance	11
05	Statistics: Distribution – Normal	Hypothesis testing	12
06	Exploratory Data Analysis (EDA)	Linear Regression	13
07	**Summary for 6 weeks QA session**	**Summary for 6 weeks QA session**	**14**
15	**Course summary**		

PREVIUOSLY

Have you thought about additional DS methods applications?

Data variables types

	B	C	D	E	F	G	H
	Date	**Day**	**Temperature**	**Rainfall**	**Flyers**	**Price**	**Sales**
	01.01.2017	Sunday	27	2,00	15	0,3	10
	02.01.2017	Monday	28,9	1,33	15	0,3	13
	03.01.2017	Tuesday	34,5	1,33	27	0,3	15
	04.01.2017	Wednesday	44,1	1,05	28	0,3	17
	05.01.2017	Thursday	42,4	1,00	33	0,3	18
	06.01.2017	Friday	25,3	1,54	23	0,3	11
	07.01.2017	Saturday	32,9	1,54	19	0,3	13
	08.01.2017	Sunday	37,5	1,18	28	0,3	15
	09.01.2017	Monday	38,1	1,18	20	0,3	17

Data variables types

Date – datetime

Day – string

Temperature – float

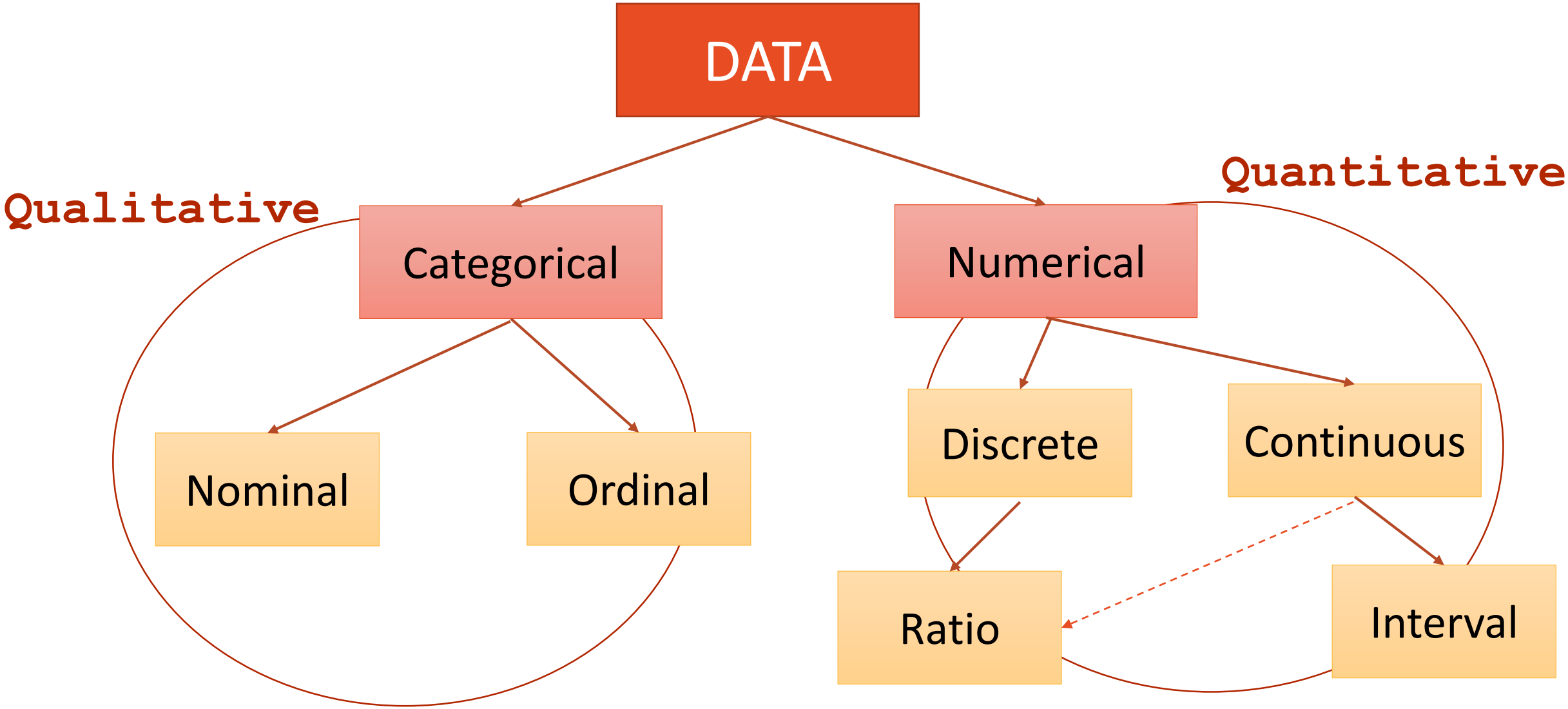
Rainfall – float

Flayers – integer

Price – float

Sales – integer

Data variables types



Data variables types

Why its important to know?

-to choose the right visualization method;

-to perform Exploratory Data Analysis (EDA);

-to create more accurate models

Categorical data

-Categorical data represents characteristics

-Can also take on numerical values

-Numerical value has no mathematical meaning

Example,

Gender: Female, Male;

Gender: 1, 2;

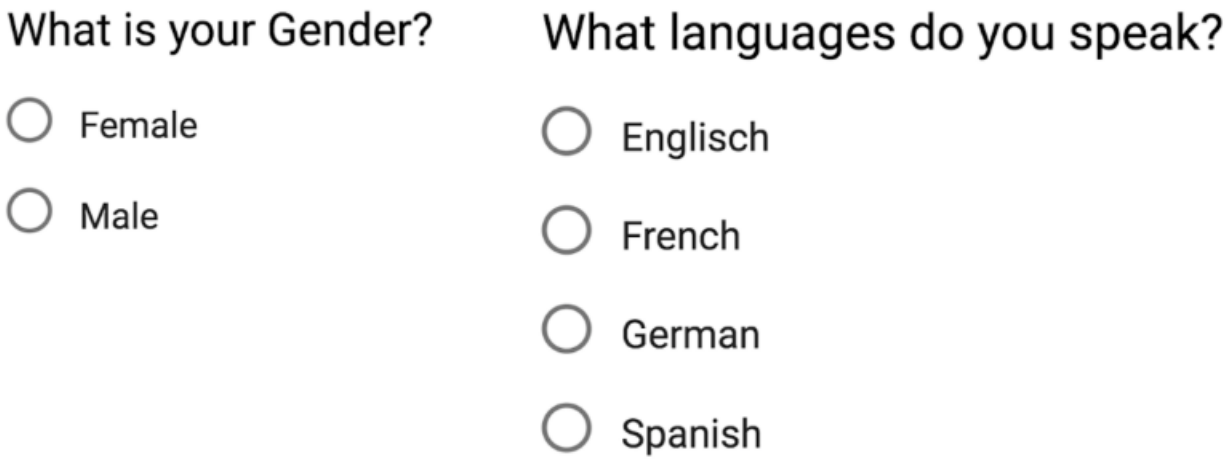
Weekdays: Monday, Tuesday, Wednesday, …;

Weekdays: 1, 2, 3, 4, 5, 6, 7.

Categorical data

Nominal:

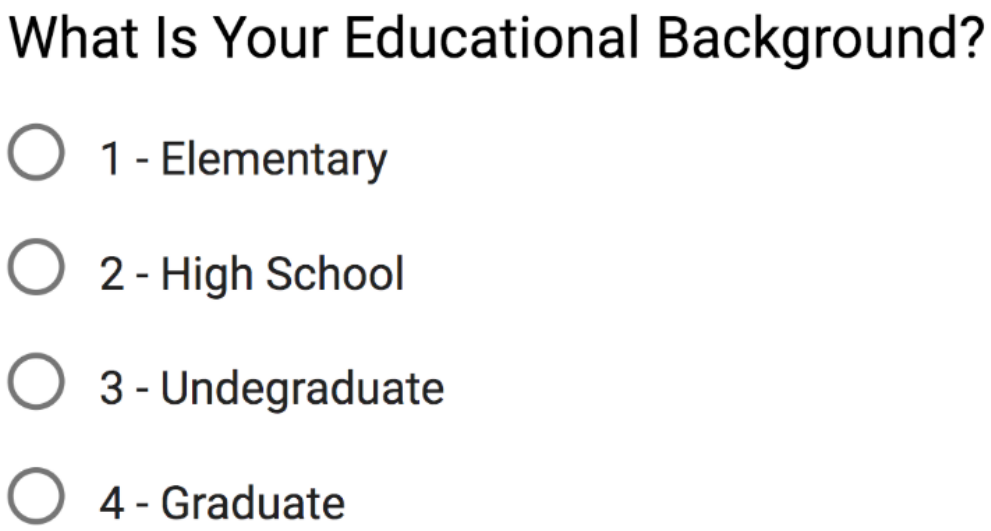
- Nominal values represent discrete units and are used to label variables
- Nominal data that has no order.



Categorical data

Ordinal:

- Ordinal values represent discrete and ordered units
- Ordinal data that has the order.



Numerical data

Discrete (Ratio):

- Its values are **distinct** and **separate:** data can only take on certain values
- Can’t be measured but it can be **counted.**

Example,

Number of students in class, number of sales, number of children in family, weight of the luggage

Numerical data

Continuous (Interval):

- An interval scale is one where there is order and the difference between two values is meaningful
- **Can** be **measured** and can be **counted.**

Example,

Temperature, IELTS score, credit score

Let's practice

Ratio

B	C	D	E	F	G	H
Date	**Day**	**Temperature**	**Rainfall**	**Flyers**	**Price**	**Sales**
01.01.2017	Sunday	27	2,00	15	0,3	10
02.01.2017	Monday	28,9	1,33	15	0,3	13
03.01.2017	Tuesday	34,5	1,33	27	0,3	15

Numerical, discrete

Nominal

Interval

Ratio

Discrete

Discrete

Variables types

- **Independent**: cannot be affected by researchers
- **Confounding**: hidden independent
- **Dependent (target)**:can be affected by researchers experiments

Variables types

B	C	D	E	F	G	H
Date	**Day**	**Temperature**	**Rainfall**	**Flyers**	**Price**	**Sales**
01.01.2017	Sunday	27	2,00	15	0,3	10
02.01.2017	Monday	28,9	1,33	15	0,3	13
03.01.2017	Tuesday	34,5	1,33	27	0,3	15

Independent variables

Dependent (target)

Pandas

- **Pandas:** Python Data Analysis Library
- "**An open source,** BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language" (https://pandas.pydata.org/)
- **Sponsored by** NumFOCUS, a non-proft organization in the US (like NumPy, Matplotlib, Jupyter, and Julia)
- **Used in StatsModel, sklearn-pandas, Plotly, IPython, Jupyter, Spyder** (http://pandas-docs.github.io/pandas-docs-travis/ecosystem.html)

Pandas

- **Built** on top of NumPy
- **Part of the SciPy** ecosystem (Scientifc Computing Tools for Python)
- **Version history**
  - Project initiated in 2008
  - Oldest version in the doc: 0.4.1 (September 2011)
  - Current version: 1.0.0 (Jan 29, 2020)
  - Previous version: 0.25.3 (Oct 31, 2019)

Pandas

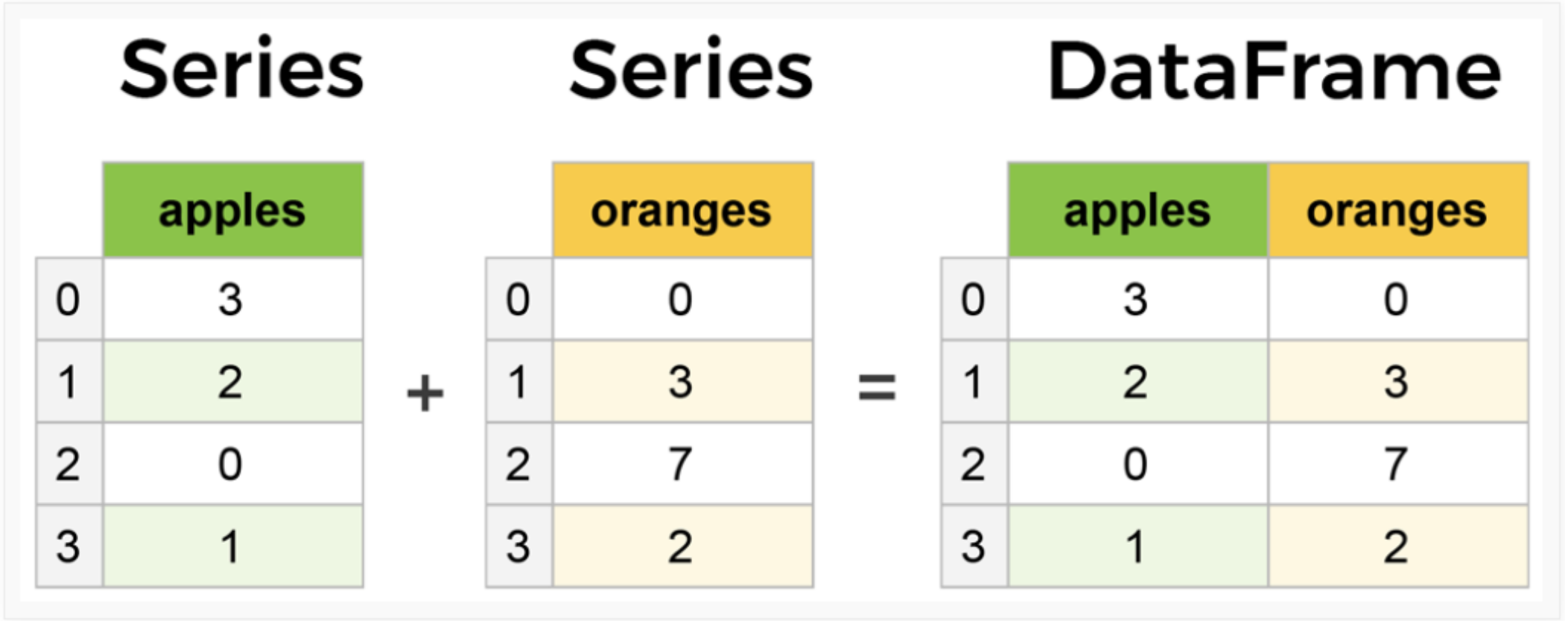
- **Data structure:**
  - Series: 1 dimensional
  - DataFrames: 2 dimensional

Pandas

- **Series**: "**One-dimensional ndarray** with axis labels (including time series)." (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas. Series.html)

- **DataFrame**: "**Two-dimensional** size-mutable, potentially heterogeneous tabular **data structure** with labeled axes (rows and columns). Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects. The primary pandas data structure." (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas. DataFrame.html)

Pandas


Series
Series
DataFrame
apples
oranges
apples
oranges
0 3
1 2
2 0
3 1
+
0 0
1 3
2 7
3 2
=
0 3 0
1 2 3
2 0 7
3 1 2

Creation of Series

```
In [5]: temp_series = pd.Series([10,20,30,40,50],
                                index = ['ALA', 'AMS', 'TXL', 'KBP', 'BCN'],
                                name = 'Airports')
        temp_series

Out[5]: ALA    10
        AMS    20
        TXL    30
        KBP    40
        BCN    50
        Name: Airports, dtype: int64
```

```
In [6]: temp_series.dtype

Out[6]: dtype('int64')
```

```
In [7]: temp_series.name

Out[7]: 'Airports'
```

```
In [8]: temp_series.index

Out[8]: Index(['ALA', 'AMS', 'TXL', 'KBP', 'BCN'], dtype='object')
```

Accessing of Series

```
In [10]: temp_series['ALA']

Out[10]: 10

In [11]: temp_series['ALA':'KBP']

Out[11]: ALA    10
         AMS    20
         TXL    30
         KBP    40
         Name: Airports, dtype: int64

In [12]: temp_series['TXL'] = 35

In [13]: temp_series

Out[13]: ALA    10
         AMS    20
         TXL    35
         KBP    40
         BCN    50
         Name: Airports, dtype: int64
```

Addition of Series

In [15]:
```
temp_series02 = pd.Series([3,15,7], index=['ALA','AMS','BCN'])
temp_series02
```

Out[15]:
```
ALA     3
AMS    15
BCN     7
dtype: int64
```

In [16]:
```
temp_series.add(temp_series02)
```

Out[16]:
```
ALA    13.0
AMS    35.0
BCN    57.0
KBP     NaN
TXL     NaN
dtype: float64
```

In [17]:
```
temp_series.add(temp_series02, fill_value=0)
```

Out[17]:
```
ALA    13.0
AMS    35.0
BCN    57.0
KBP    40.0
TXL    35.0
dtype: float64
```

Dataframes

- "Two-dimensional size-mutable, potentially Axis 1 heterogeneous tabular data structure with labeled axes (rows and columns)."

- **Mutability:** Columns can have different dtypes and can be added and removed, but they have a fixed size.

- **Semantic:** Similar to a table in a relational database.

Dataframe creation

In [20]:
```
data = pd.DataFrame({'Code':['ALA', 'BCN', 'TXL', 'CDG', 'FUM', 'AMS'],
                     'Passengers':[10,20,30,40,55,60],
                     'Importance':[1,5,4,5,5,5]})
data
```

Out[20]:

	Code	Passengers	Importance
0	ALA	10	1
1	BCN	20	5
2	TXL	30	4
3	CDG	40	5
4	FUM	55	5
5	AMS	60	5

In [21]:
```
data.dtypes
```

Out[21]:
```
Code          object
Passengers     int64
Importance     int64
dtype: object
```

In [22]:
```
data.shape
```

Out[22]:
```
(6, 3)
```

In [23]:
```
data.columns
```

Out[23]:
```
Index(['Code', 'Passengers', 'Importance'], dtype='object')
```

Dataframe creation

```
In [39]: dates = pd.date_range('20200101', periods = 5)

In [40]: df = pd.DataFrame(np.random.randn(5, 4), index=dates, columns=list('ABCD'))

In [41]: df
```

Out[41]:

	A	B	C	D
2020-01-01	0.297807	-1.370035	0.049935	0.244069
2020-01-02	1.317901	-0.515574	-1.133906	-0.781119
2020-01-03	-0.245720	0.585406	1.797836	0.350020
2020-01-04	0.910939	0.596556	-1.234914	0.049009
2020-01-05	-1.870426	0.160980	-0.454190	0.182484

Dataframe viewing

In [44]:
```
data.head(2)
```

Out[44]:

	Code	Passengers	Importance
0	ALA	10	1
1	BCN	20	5

In [45]:
```
data.tail(2)
```

Out[45]:

	Code	Passengers	Importance
4	FUM	55	5
5	AMS	60	5

In [46]:
```
data.Code
```

Out[46]:
```
0    ALA
1    BCN
2    TXL
3    CDG
4    FUM
5    AMS
Name: Code, dtype: object
```

Dataframe viewing

In [25]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Code        6 non-null      object
 1   Passengers  6 non-null      int64
 2   Importance  6 non-null      int64
dtypes: int64(2), object(1)
memory usage: 184.0+ bytes
```

In [26]:
```
data.describe()
```

Out[26]:

	Passengers	Importance
count	6.000000	6.000000
mean	35.833333	4.166667
std	19.600170	1.602082
min	10.000000	1.000000
25%	22.500000	4.250000
50%	35.000000	5.000000
75%	51.250000	5.000000
max	60.000000	5.000000

Dataframe functions

```
In [27]: data.sum()
```

```
Out[27]: Code          ALABCNTXLCDGFUMAMS
         Passengers                   215
         Importance                    25
         dtype: object
```

```
In [30]: data.sum(axis=1)
```

```
Out[30]: 0    11
         1    25
         2    34
         3    45
         4    60
         5    65
         dtype: int64
```

Dataframe functions

```
In [27]: data.sum()
```

```
Out[27]: Code          ALABCNTXLCDGFUMAMS
         Passengers                   215
         Importance                    25
         dtype: object
```

```
In [30]: data.sum(axis=1)
```

```
Out[30]: 0    11
         1    25
         2    34
         3    45
         4    60
         5    65
         dtype: int64
```

Selecting data

For getting a cross section using a label

In [51]:
```
data.loc[0:3]
```

Out[51]:

	Code	Passengers	Importance
0	ALA	10	1
1	BCN	20	5
2	TXL	30	4
3	CDG	40	5

In [56]:
```
data.loc[:, ['Code', 'Importance']]
```

Out[56]:

	Code	Importance
0	ALA	1
1	BCN	5
2	TXL	4
3	CDG	5
4	FUM	5
5	AMS	5

Selecting data

Select via the position of the passed integers:

```
In [57]: data.iloc[0,2]
Out[57]: 1

In [58]: data.iloc[:, 1:2]
```

Out[58]:

	Passengers
0	10
1	20
2	30
3	40
4	55
5	60

Boolean selecting

```
In [59]: data[data.Passengers > 20]
```

Out[59]:

	Code	Passengers	Importance
2	TXL	30	4
3	CDG	40	5
4	FUM	55	5
5	AMS	60	5

```
In [63]: data[data.Code.isin(['ALA', 'TXL'])]
```

Out[63]:

	Code	Passengers	Importance
0	ALA	10	1
2	TXL	30	4

Missing values

In [73]:
```
data
```

Out[73]:

	Code	Passengers	Importance
0	ALA	10.0	1
1	BCN	20.0	5
2	TXL	30.0	4
3	CDG	40.0	5
4	FUM	55.0	5
5	AMS	60.0	5
6	SXP	NaN	3

In [77]:
```
data.isna()
```

Out[77]:

	Code	Passengers	Importance
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False
5	False	False	False
6	False	True	False

Missing values

```
In [78]: data.fillna(value = 1)
```

Out[78]:

	Code	Passengers	Importance
0	ALA	10.0	1
1	BCN	20.0	5
2	TXL	30.0	4
3	CDG	40.0	5
4	FUM	55.0	5
5	AMS	60.0	5
6	SXP	1.0	3

```
In [81]: data.dropna(how='any')
```

Out[81]:

	Code	Passengers	Importance
0	ALA	10.0	1
1	BCN	20.0	5
2	TXL	30.0	4
3	CDG	40.0	5
4	FUM	55.0	5
5	AMS	60.0	5

Apply

In [84]:
```
data[['Passengers']].apply(np.cumsum)
```

Out[84]:

	Passengers
0	10.0
1	30.0
2	60.0
3	100.0
4	155.0
5	215.0
6	NaN

In [85]:
```
data
```

Out[85]:

	Code	Passengers	Importance
0	ALA	10.0	1
1	BCN	20.0	5
2	TXL	30.0	4
3	CDG	40.0	5
4	FUM	55.0	5
5	AMS	60.0	5
6	SXP	NaN	3

In [86]:
```
data.Importance.apply(lambda x: x*x)
```

Out[86]:
```
0     1
1    25
2    16
3    25
4    25
5    25
6     9
Name: Importance, dtype: int64
```

Grouping

Out[130]:

	Code	Passengers	Importance
0	ALA	10	1
1	BCN	20	5
2	TXL	30	4
3	CDG	40	5
4	FUM	55	5
5	AMS	60	5
0	ALA	5	1
1	TXL	20	4
2	BCN	8	5

In [129]:
```
data.groupby('Code').sum()
```

Out[129]:

Code	Passengers	Importance
ALA	15	2
AMS	60	5
BCN	28	10
CDG	40	5
FUM	55	5
TXL	50	8

Readings

- https://pandas.pydata.org/docs/getting_started/install.html
- **Data Science from Scratch,** Book by Joel Grus

Additional resources

- Khan academy