DATA CLEANING

Meta Brains

Introduction to Data Cleaning

- Data extracted from real world contains incorrect, incomplete, irrelevant or missing values which need to be cleaned.

- Cleaning can be done by modifying, replacing or deleting such values.

- Data cleaning is a fundamental part of data science lifecycle.

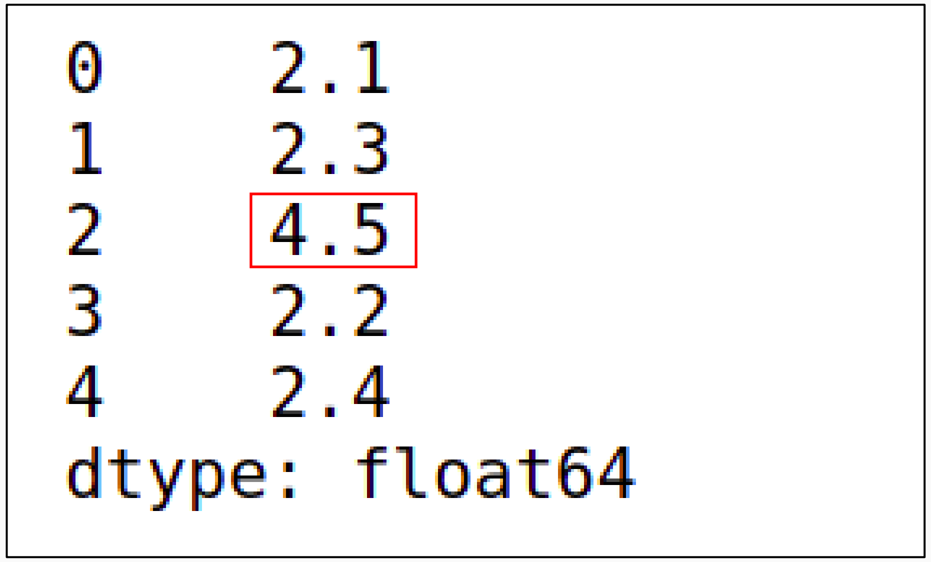
https://www.educative.io/blog/what-is-data-cleaning

Quality of Data

- Today’s world is all about data-driven decision making, hence the quality of our data will ultimately impact the quality of the decision that we make based on that data.
- Data is generally considered high quality if it is “**fit for [its] intended uses in operations, decision making and planning.**”
- Through different data cleaning techniques, we can improve the quality of our data extracted from the real world.

Examples of Anomalies (1/2)

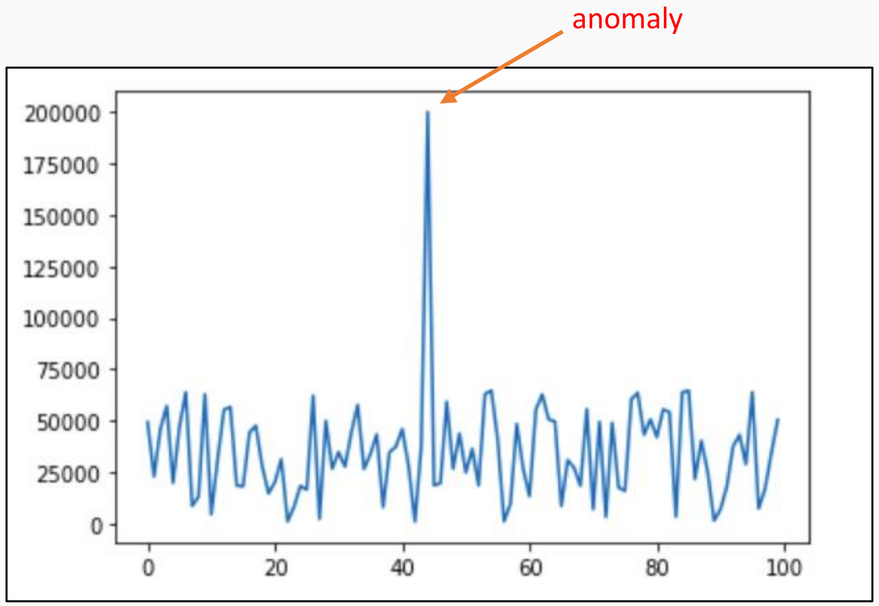
- An anomaly in data, also known as an outlier, is the observation which differs significantly from the standard pattern.
- Anomalies in data could arise for a number of reasons, such as human error.
- Consider the given series containing width of a table measured by 5 different students.
  - The third value is significantly different than the rest – an anomaly!

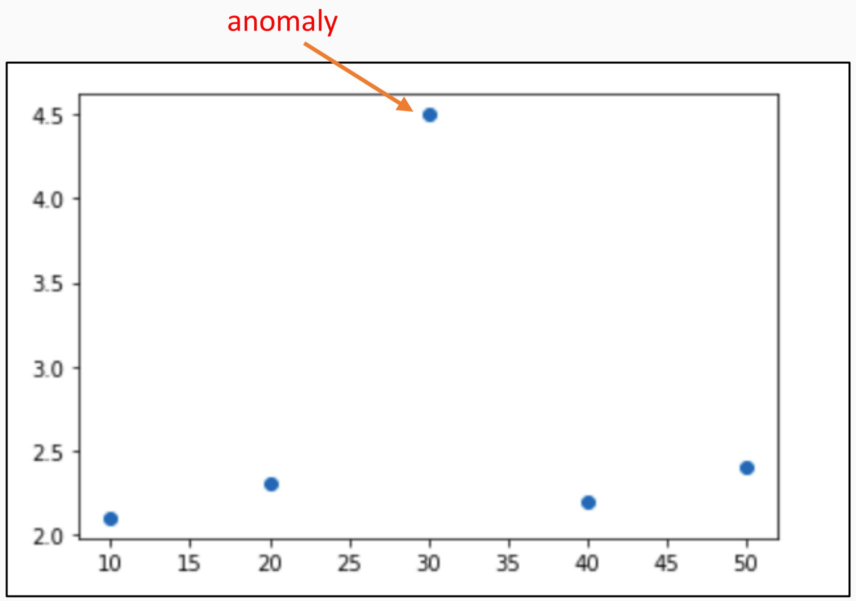


```
0    2.1
1    2.3
2    4.5
3    2.2
4    2.4
dtype: float64
```

Examples of Anomalies (2/2)

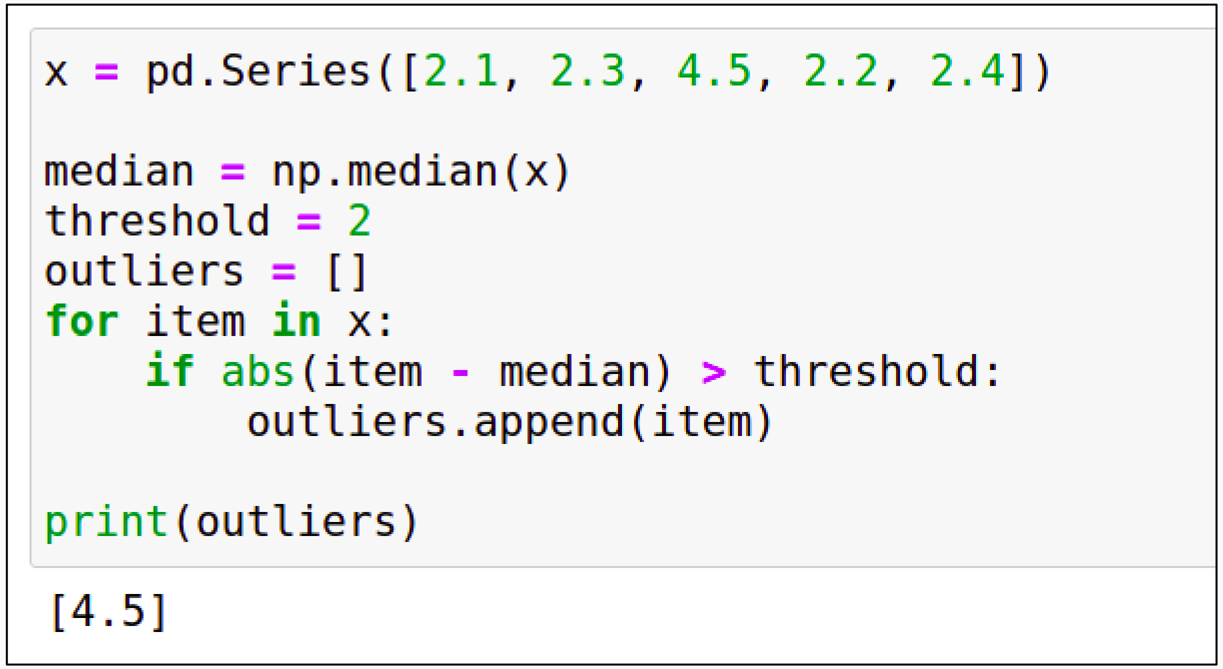
- Data extracted from real world often contains such anomalies;
  - Temperature data over a period of 6 months might contain some irregular values due to tremendous shifts in weather conditions.
- It is important to detect such anomalies in the data and deal with them appropriately.





Median-based Anomaly Detection

- One way to detect anomalies is using the median value.
- We set a reasonable threshold and if for a certain value, |value – median| > threshold, then the value is considered an anomaly.



```
x = pd.Series([2.1, 2.3, 4.5, 2.2, 2.4])

median = np.median(x)
threshold = 2
outliers = []
for item in x:
    if abs(item - median) > threshold:
        outliers.append(item)

print(outliers)
```

```
[4.5]
```

Quiz Time

A certain dataset has a median value 3, and the threshold for anomaly detection is 2. Third value of the dataset is 7. According to median-based anomaly detection, is the third value an anomaly?

- Yes
- No

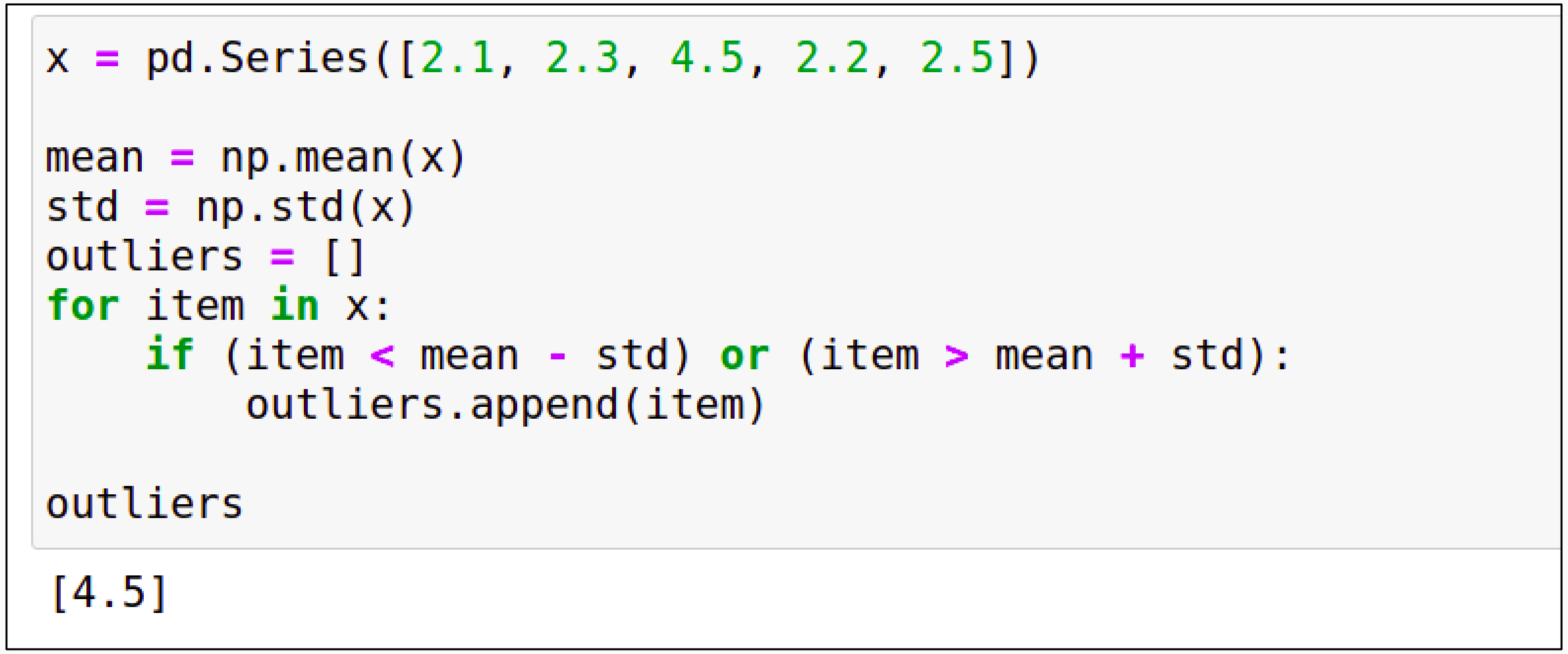
Quiz Time

A certain dataset has a median value 3, and the threshold for anomaly detection is 2. Third value of the dataset is 7. According to median-based anomaly detection, is the third value an anomaly?

- Yes
- No

Mean-based Anomaly Detection

- Another way to detect anomalies is using mean and standard deviation of the data.
- We define the range as (mean – standard deviation) <= value <= (mean + standard deviation)
  - i.e., any value less than (mean – standard deviation) or greater than (mean + standard deviation) is considered an anomaly.



```
x = pd.Series([2.1, 2.3, 4.5, 2.2, 2.5])

mean = np.mean(x)
std = np.std(x)
outliers = []
for item in x:
    if (item < mean - std) or (item > mean + std):
        outliers.append(item)

outliers
```

```
[4.5]
```

Quiz Time

1. A certain dataset has a mean value 20 and a standard deviation of 2.5. A certain value x in the dataset is 16. According to mean-based anomaly detection, is x an anomaly?

- Yes
- No

Quiz Time

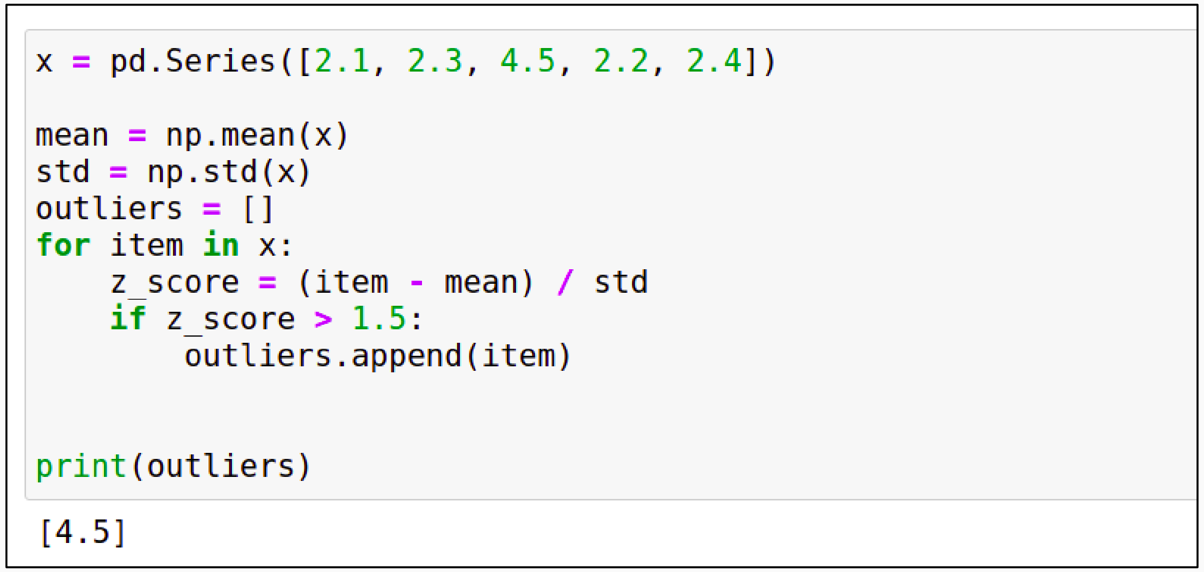
1. A certain dataset has a mean value 20 and a standard deviation of 2.5. A certain value x in the dataset is 16. According to mean-based anomaly detection, is x an anomaly?

- Yes
- No

Meta Brains

Z-score-based Anomaly Detection

- Another technique for anomaly detection is the Z-score.
- Z-score is a statistical measure showing how many standard deviations a certain value is from the mean.
- It is defined as;
    - Z = (value – mean) / standard deviation
- If the Z-score of a value is greater than a reasonable threshold, it is considered an anomaly.



```
x = pd.Series([2.1, 2.3, 4.5, 2.2, 2.4])

mean = np.mean(x)
std = np.std(x)
outliers = []
for item in x:
    z_score = (item - mean) / std
    if z_score > 1.5:
        outliers.append(item)


print(outliers)
```

```
[4.5]
```

Quiz Time

1. A certain dataset has a mean value 12 and a standard deviation of 2. A certain value x in the dataset is 8. Compute the Z-score of the value x?

- 4
- 20
- 2
- 10

Meta Brains

Quiz Time

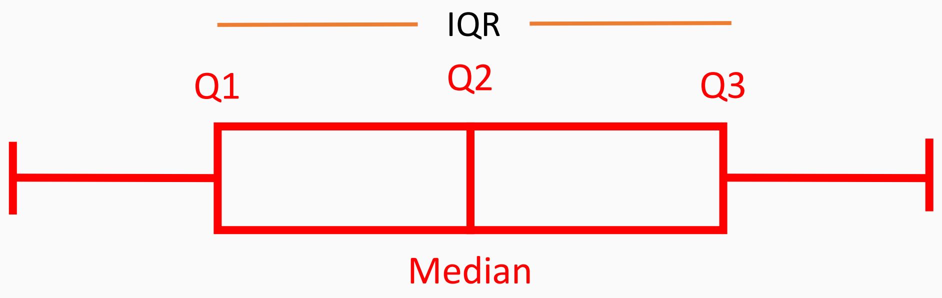
1. A certain dataset has a mean value 12 and a standard deviation of 2. A certain value x in the dataset is 8. Compute the Z-score of the value x?

- 4
- 20
- 2
- 10

Interquartile Range for Anomaly Detection (1/3)

IQR

- We can also use interquartile range (IQR) for detecting anomalies in the data.
- A quartile divides the dataset (sorted from smallest to largest) into 3 points and 4 intervals.
- The interquartile range (IQR) is the difference between the 3rd quartile and 1st quartile (IQR = Q3 -Q1).



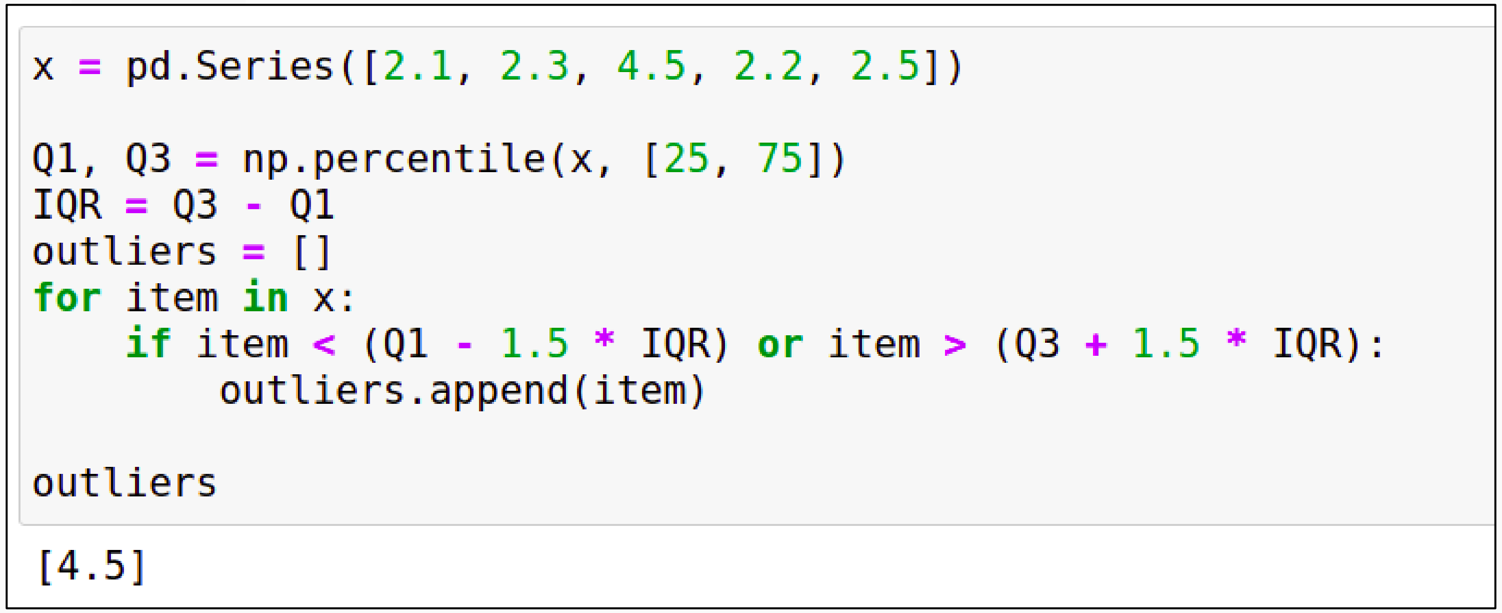
Interquartile Range for Anomaly Detection (2/3)

- Consider the series of widths from the previous slide as a list as shown below;
  - Widths = [2.3, 2.2, 4.5, 2.1, 2.5]
- We first sort this list from smallest to largest.
  - Widths = [2.1, 2.2, 2.3, 2.5, 4.5]
- The first quartile is at 25%, which is 2.2.
- The second quartile is 50%, which is 2.3 (median).
- The third quartile is 75%, which is 2.5.
- Use the numpy.percentile() function to get the quartiles of a dataset.
  - Automatically sorts the data from smallest to largest.

Interquartile Range for Anomaly Detection (3/3)

Anomaly

- Any value less than (Q1 – 1.5x IQR) or greater than (Q3 + 1.5x IQR) is considered an anomaly.



```python
x = pd.Series([2.1, 2.3, 4.5, 2.2, 2.5])

Q1, Q3 = np.percentile(x, [25, 75])
IQR = Q3 - Q1
outliers = []
for item in x:
    if item < (Q1 - 1.5 * IQR) or item > (Q3 + 1.5 * IQR):
        outliers.append(item)

outliers
```

```
[4.5]
```

Quiz Time

1. Consider the given dataset x=[1,2,3,4,5]. What is the second quartile (Q2) of the dataset?
    - 2
    - 3
    - 4
2. Compute the interquartile range for the dataset x=[1,2,3,4,5].
    - 1
    - 2
    - 3

Meta Brains

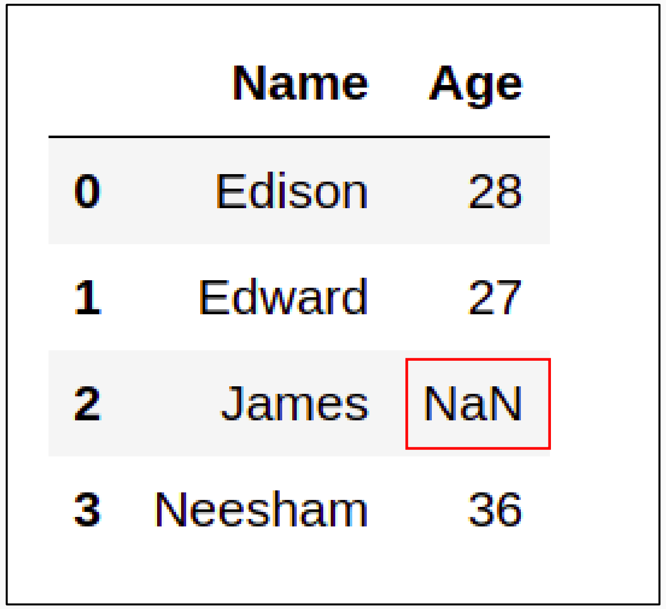
Quiz Time

1. Consider the given dataset x=[1,2,3,4,5]. What is the second quartile (Q2) of the dataset?
   - 2
   - 3
   - 4
2. Compute the interquartile range for the dataset x=[1,2,3,4,5].
   - 1
   - 2
   - 3

Meta Brains

Dealing with Missing values (1/6)

- Apart from outliers/anomalies, data also often contains missing values.
- A missing value means loss of information.
- In pandas, an NaN indicates a missing value.
- Consider the following dataframe called data with one missing value in the 'Age' column.

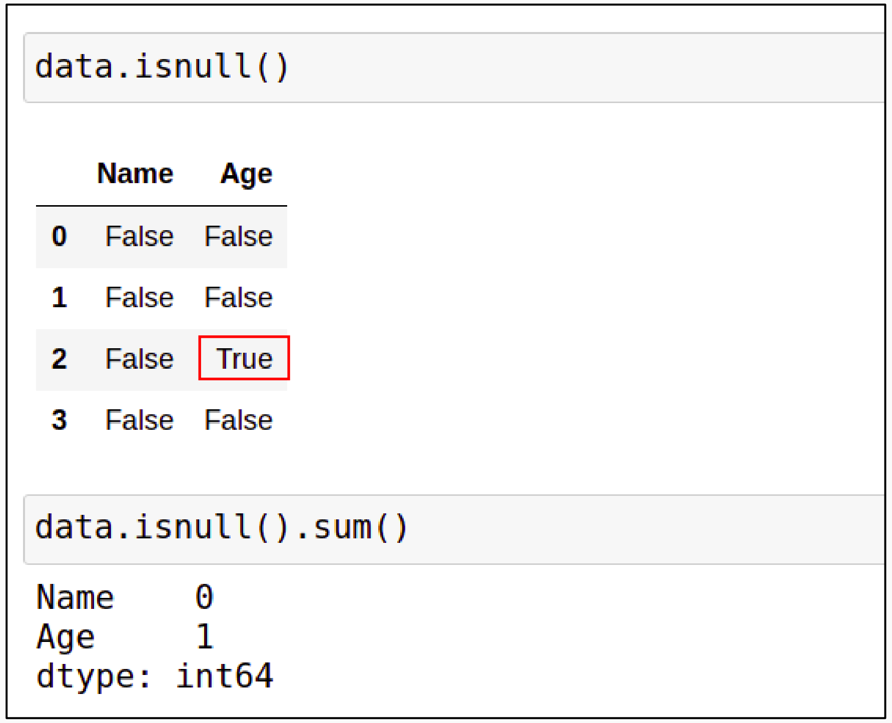


	Name	Age
0	Edison	28
1	Edward	27
2	James	NaN
3	Neesham	36

Dealing with Missing values (2/6)

Finding Missing Values in Pandas

- The .isnull() function tells us if a cell is empty or not.
- Use the .sum() function with the .isnull() function to find total number of missing values in the data.



```
data.isnull()
```

	Name	Age
0	False	False
1	False	False
2	False	True
3	False	False

```
data.isnull().sum()
```

```
Name      0
Age       1
dtype: int64
```

Dealing with Missing values (3/6)

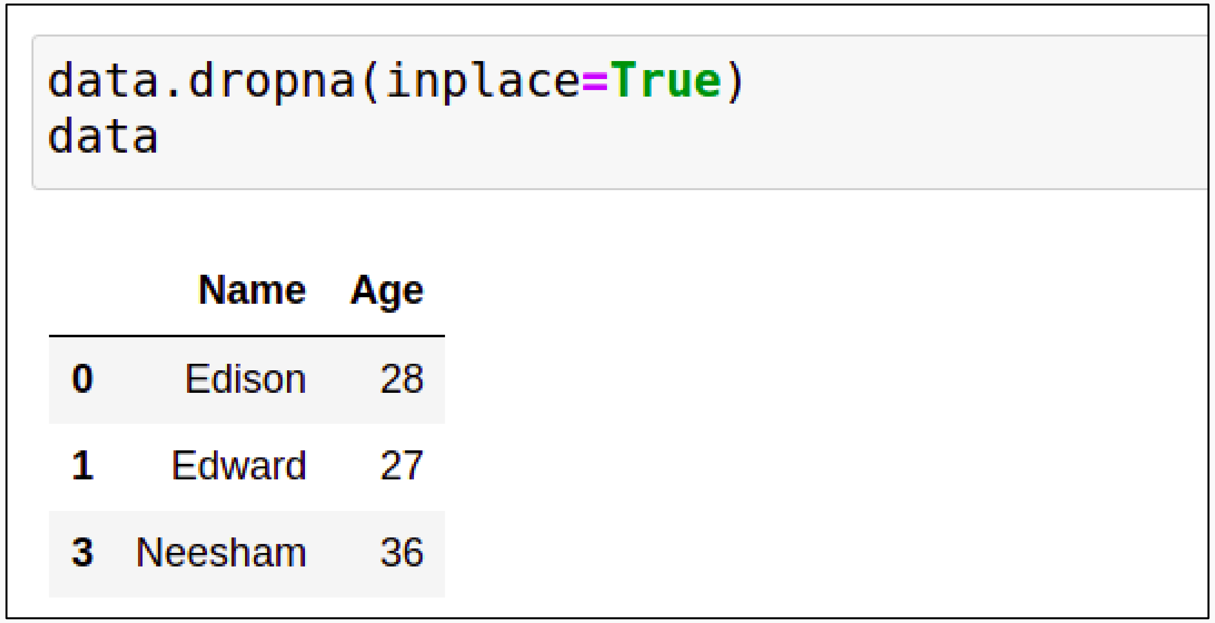
Dealing with Missing Values in Pandas

- There are a number of ways to deal with these missing values.
- Which method to use depends upon the kind of data and the task that the data is supposed to accomplish.
- Different Methods Used are;
    - Deleting rows with missing values.
    - Replacing missing values with mean/median/mode.

Dealing with Missing values (4/6)

Deleting Rows with Missing Values

- One way to deal with the missing values is to delete the rows containing missing values.
- Use the .dropna() with inplace set to True to remove missing values from the dataset.



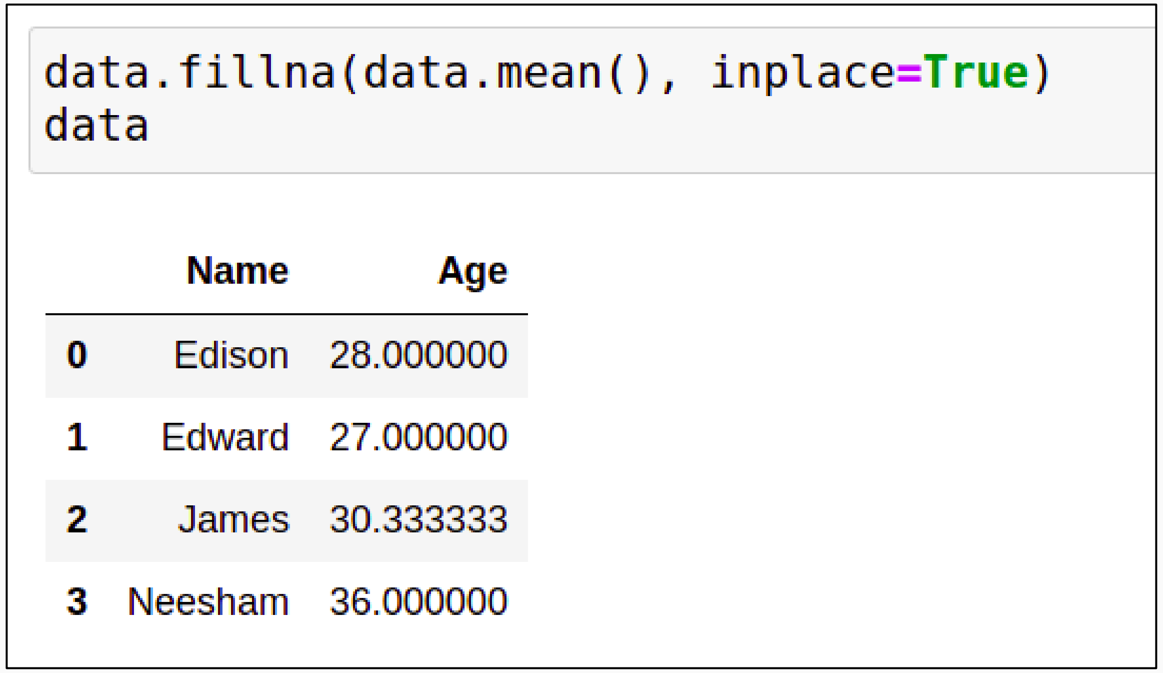
```
data.dropna(inplace=True)
data
```

	Name	Age
0	Edison	28
1	Edward	27
3	Neesham	36

Dealing with Missing values (5/6)

Replacing Missing Values with Mean/Median/Mode

- We can also replace the missing values in each column with one of the statistical measures (mean/median/mode) of that column.
- Use the .fillna() method to fill the missing values with mean, median, or mode.



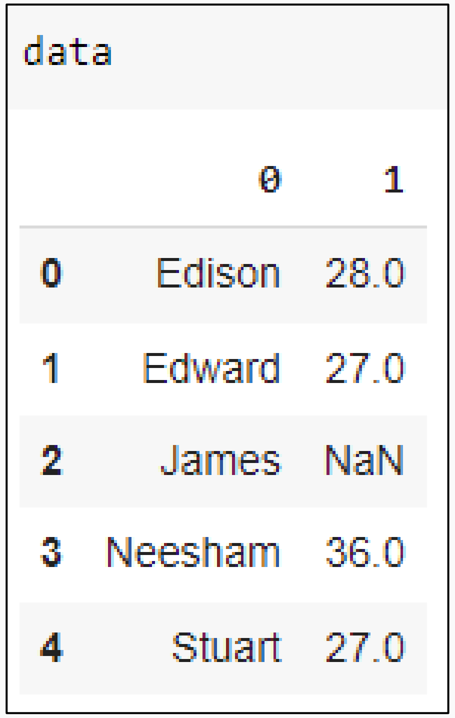
```
data.fillna(data.mean(), inplace=True)
data
```

	Name	Age
0	Edison	28.000000
1	Edward	27.000000
2	James	30.333333
3	Neesham	36.000000

Dealing with Missing values (6/6)

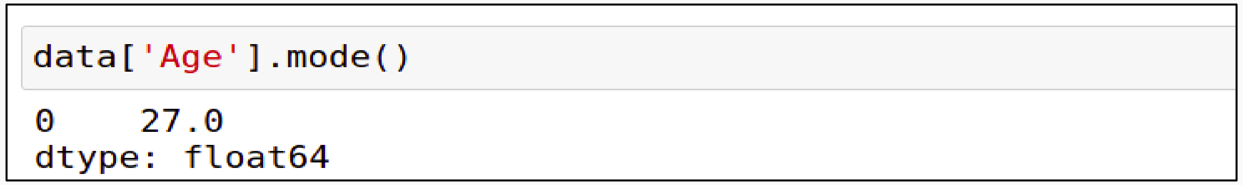
Replacing Missing Values with Mean/Median/Mode

- In this example, we replace the missing value in the ‘Age’ column with the mode of the ‘Age’ column.



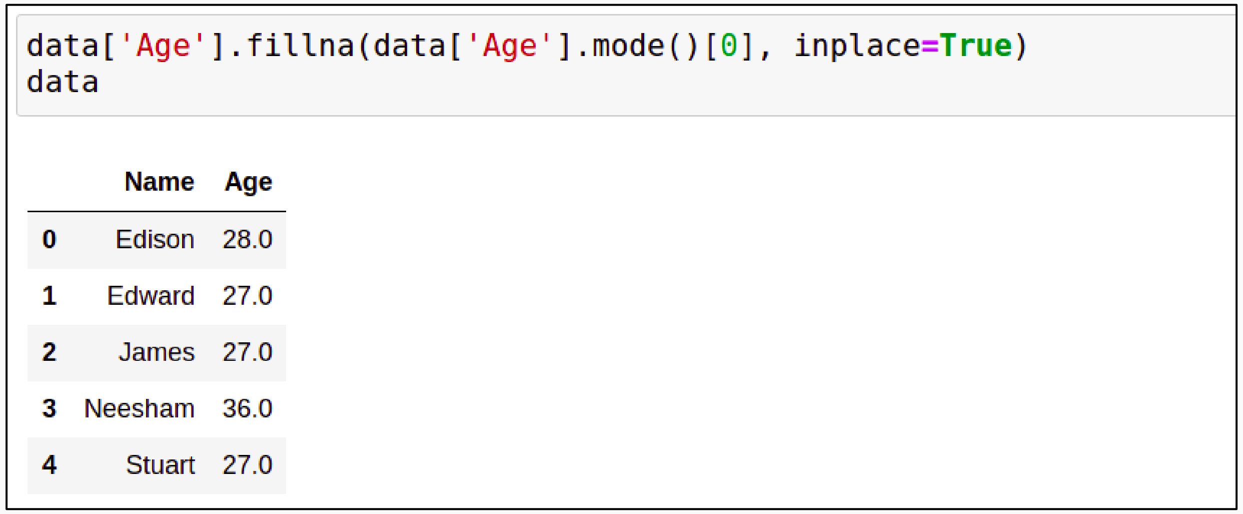
```
data
```

	0	1
0	Edison	28.0
1	Edward	27.0
2	James	NaN
3	Neesham	36.0
4	Stuart	27.0



```
data['Age'].mode()
```

```
0    27.0
dtype: float64
```

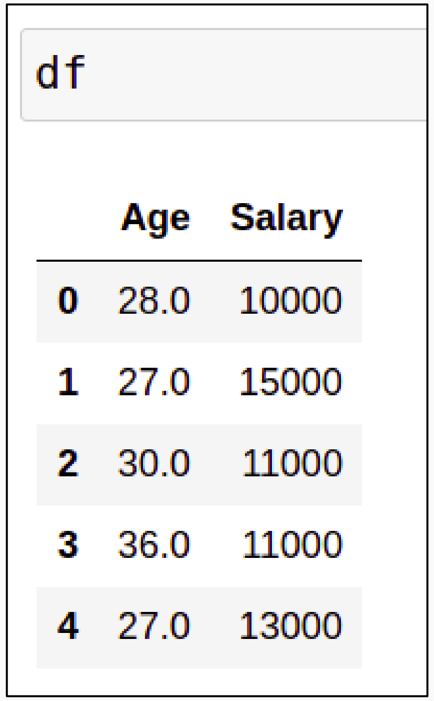


```
data['Age'].fillna(data['Age'].mode()[0], inplace=True)
data
```

	Name	Age
0	Edison	28.0
1	Edward	27.0
2	James	27.0
3	Neesham	36.0
4	Stuart	27.0

Feature Scaling (1/5)

- Sometimes, we might wish to normalize the range of each feature (column) of the given dataset.
- For example, consider the given dataframe where the range of each feature (column) is different.
- We would like to scale all the features to the same range, e.g. 0-1, 1-100, etc.



```
df
```

	Age	Salary
0	28.0	10000
1	27.0	15000
2	30.0	11000
3	36.0	11000
4	27.0	13000

Feature Scaling (2/5)

Normalization

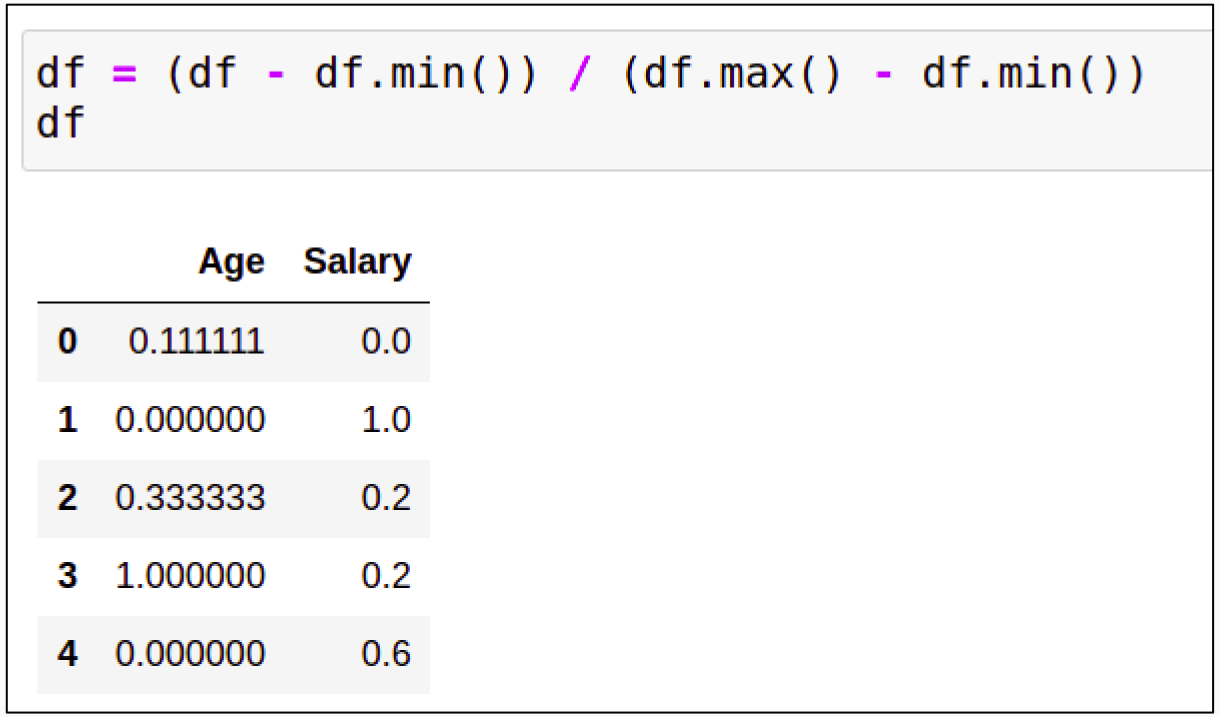
- One simple method of feature scaling is normalization, also called min-max scaling.
- For every value in a feature (column), we subtract the minimum value of the particular feature (column) from it and divide it by the difference of maximum and minimum value of that feature (column).
  - Normalized value = (original value – minimum) / (maximum – minimum)
- This method scales the features in the range [0, 1]

Meta Brains

Feature Scaling (3/5)

Normalization

- For min-max scaling, use the following line of code;
  - normalized_df=(df-df.min())/(df.max()-df.min())
- Pandas will automatically use the feature (column) min-max values for each feature.



```
df = (df - df.min()) / (df.max() - df.min())
df
```

	Age	Salary
0	0.111111	0.0
1	0.000000	1.0
2	0.333333	0.2
3	1.000000	0.2
4	0.000000	0.6

Feature Scaling (4/5)

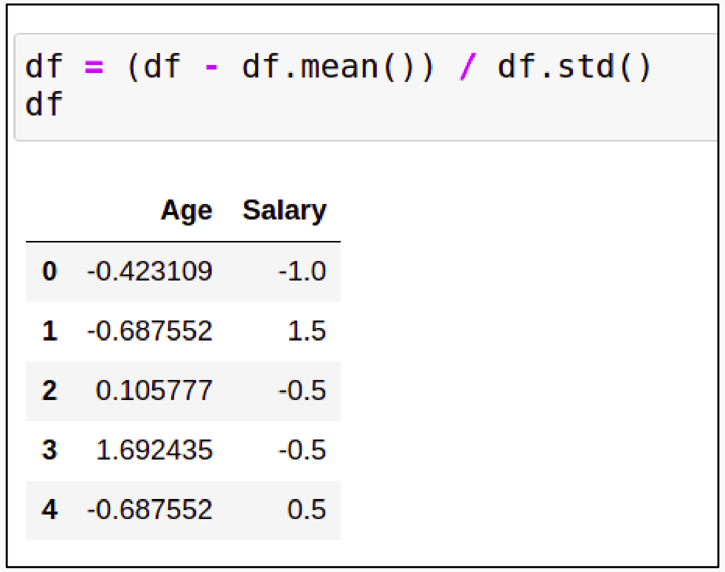
Standardization

- In standardization, for each value of a feature (column), we subtract the mean of that feature (column) from the value and divide the result by the standard deviation of that feature (column).
  - Standardized value = (original value – mean) / standard deviation
- As a result, the standard deviation of the feature (column) becomes 1.

Feature Scaling (5/5)

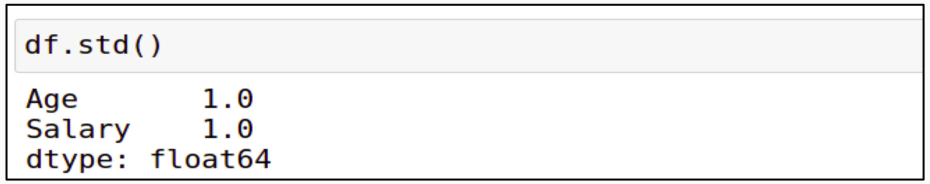
Standardization

- For standardization, use the following line of code;
  - standardized_df=(df-df.mean())/df.std()
- Once again, Pandas will automatically use the feature (column) mean and std values for each feature.



```
df = (df - df.mean()) / df.std()
df
```

	Age	Salary
0	-0.423109	-1.0
1	-0.687552	1.5
2	0.105777	-0.5
3	1.692435	-0.5
4	-0.687552	0.5



```
df.std()
```

```
Age       1.0
Salary    1.0
dtype: float64
```

Quiz Time

1. Min-max scaling is also known as;
   - Normalization
   - Standardization

Meta Brains

Quiz Time

1. Min-max scaling is also known as;
    - Normalization
    - Standardization

Meta Brains

Regular Expressions (1/8)

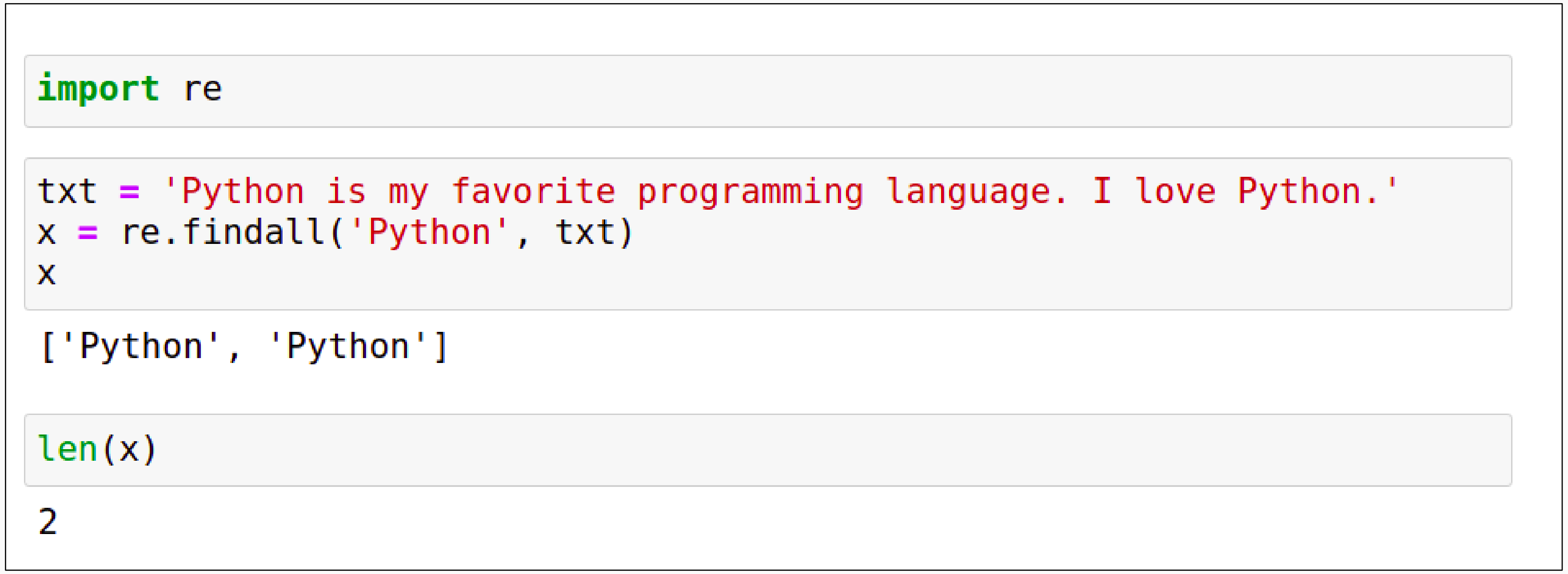
- Regular Expression or RegEx is an expression containing a sequence of characters for matching patterns in strings.
- Almost all the major programming languages have implementation for RegEx.
- The 're' module of Python is used for pattern matching using RegEx in Python.
- Following functions are available in the 're' module;
  - findall()
  - search()
  - sub()

Meta Brains

Regular Expressions (2/8)

re.findall()

- re.findall() is used to match all the occurrences of a pattern in a string.
- A list with all the matches is returned.
- In the given figure, we find how many times does the word 'Python' appear in the given string.



```
import re
```

```
txt = 'Python is my favorite programming language. I love Python.'
x = re.findall('Python', txt)
x
```

```
['Python', 'Python']
```

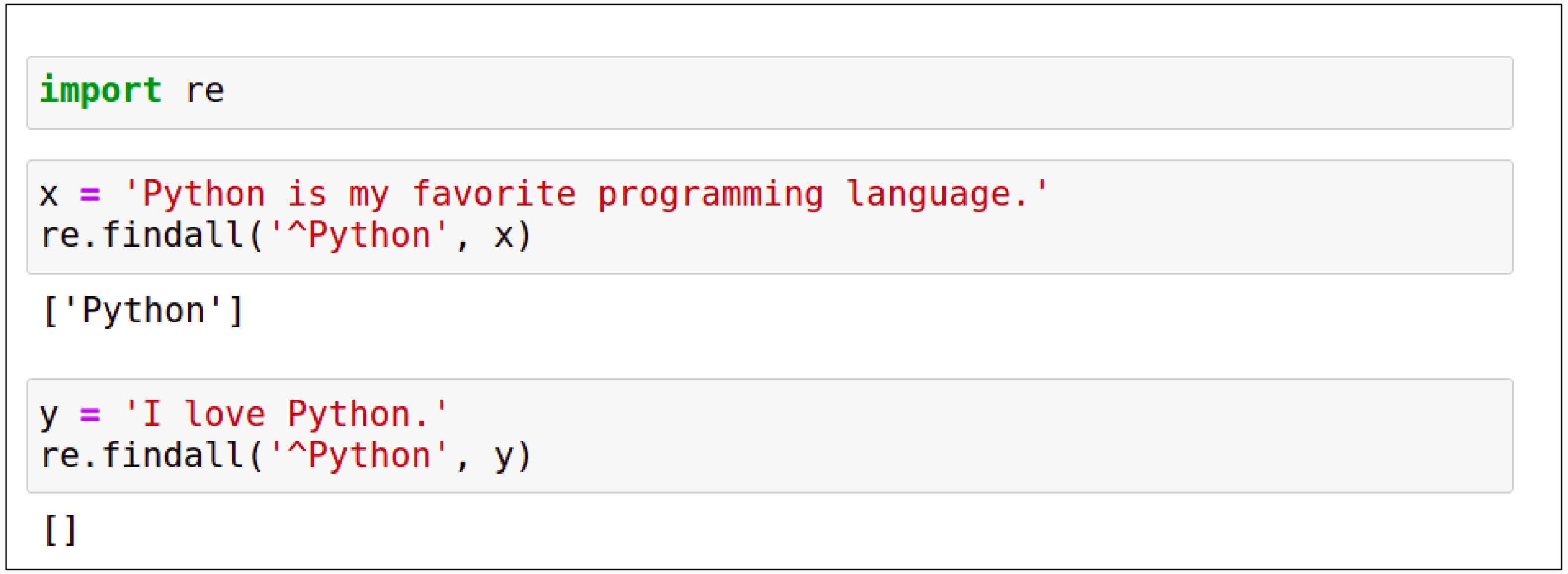
```
len(x)
```

```
2
```

Regular Expressions (3/8)

re.findall()

- In the given figure, we check if the string x starts with the word 'Python' or not.
- The ^ character returns a match only if the string starts with the pattern given after ^ symbol.
- The string y contains the word Python but does not begin with it, hence we get an empty list.



```
import re
```

```
x = 'Python is my favorite programming language.'
re.findall('^Python', x)
```

```
['Python']
```

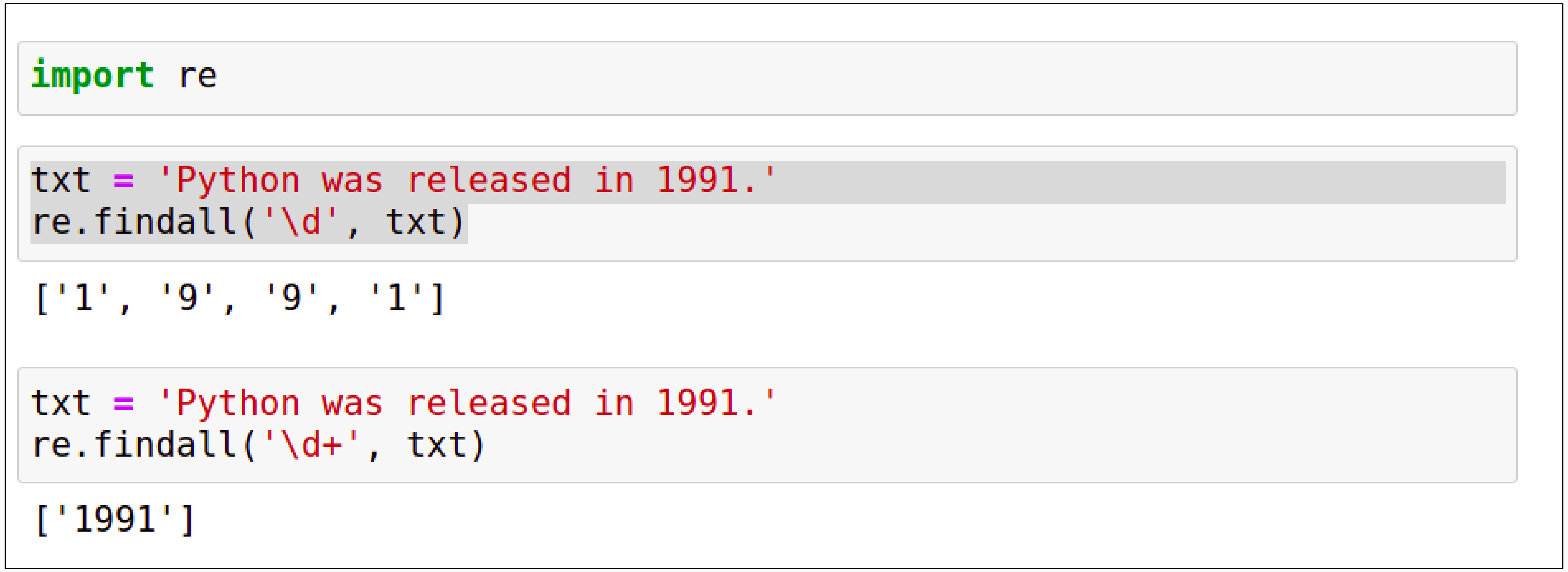
```
y = 'I love Python.'
re.findall('^Python', y)
```

```
[]
```

Regular Expressions (4/8)

re.findall()

- To match numbers in a string, we use the \d sequence.
- A + sign at the end of \d makes sure that number such as 50 is treated as 50 and not as 5 and 0.
- You can find a list of sequences at https://www.w3schools.com/python/python_regex.asp



```
import re
```

```
txt = 'Python was released in 1991.'
re.findall('\d', txt)
```

```
['1', '9', '9', '1']
```

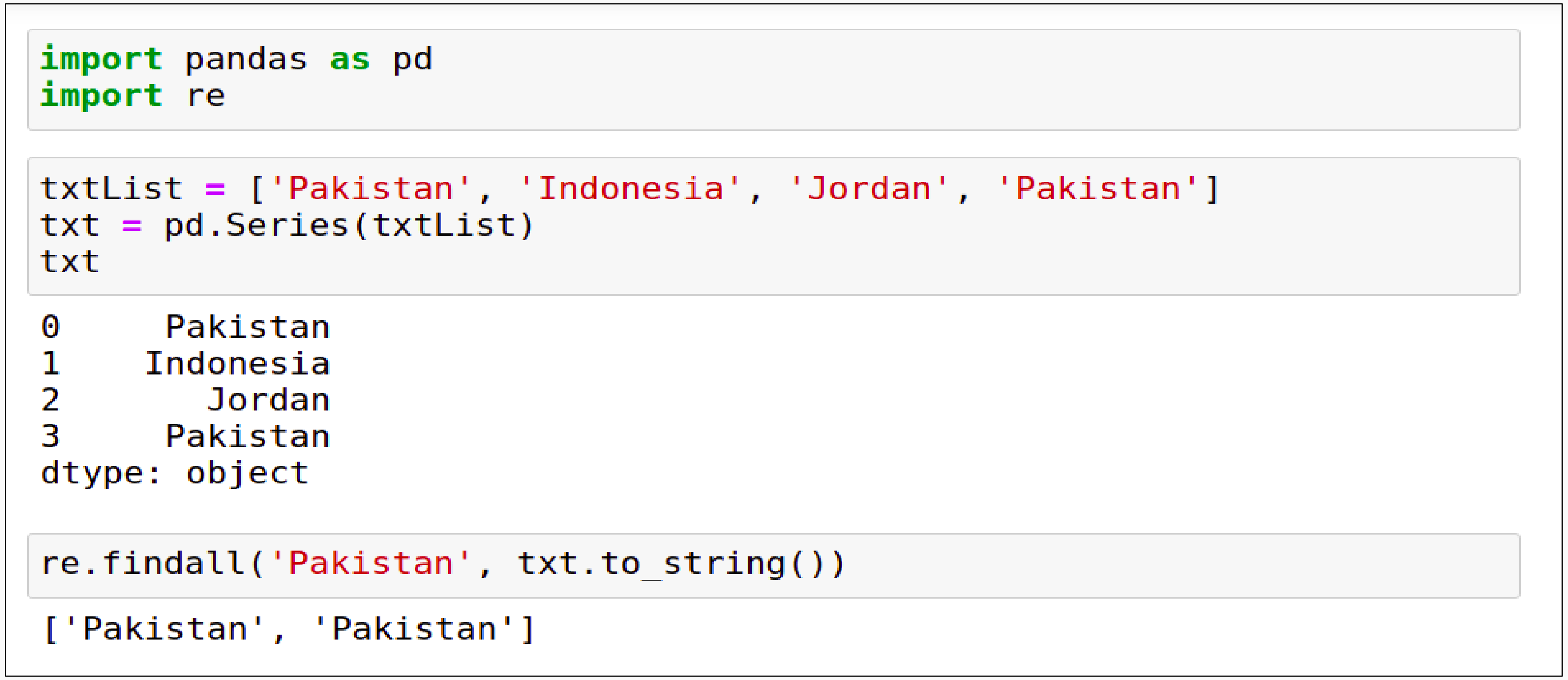
```
txt = 'Python was released in 1991.'
re.findall('\d+', txt)
```

```
['1991']
```

Regular Expressions (5/8)

re.findall()

- To find matches in a Series, we first convert the Series into a string using the .to_string() method.



```
import pandas as pd
import re
```

```
txtList = ['Pakistan', 'Indonesia', 'Jordan', 'Pakistan']
txt = pd.Series(txtList)
txt
```

```
0     Pakistan
1    Indonesia
2       Jordan
3     Pakistan
dtype: object
```

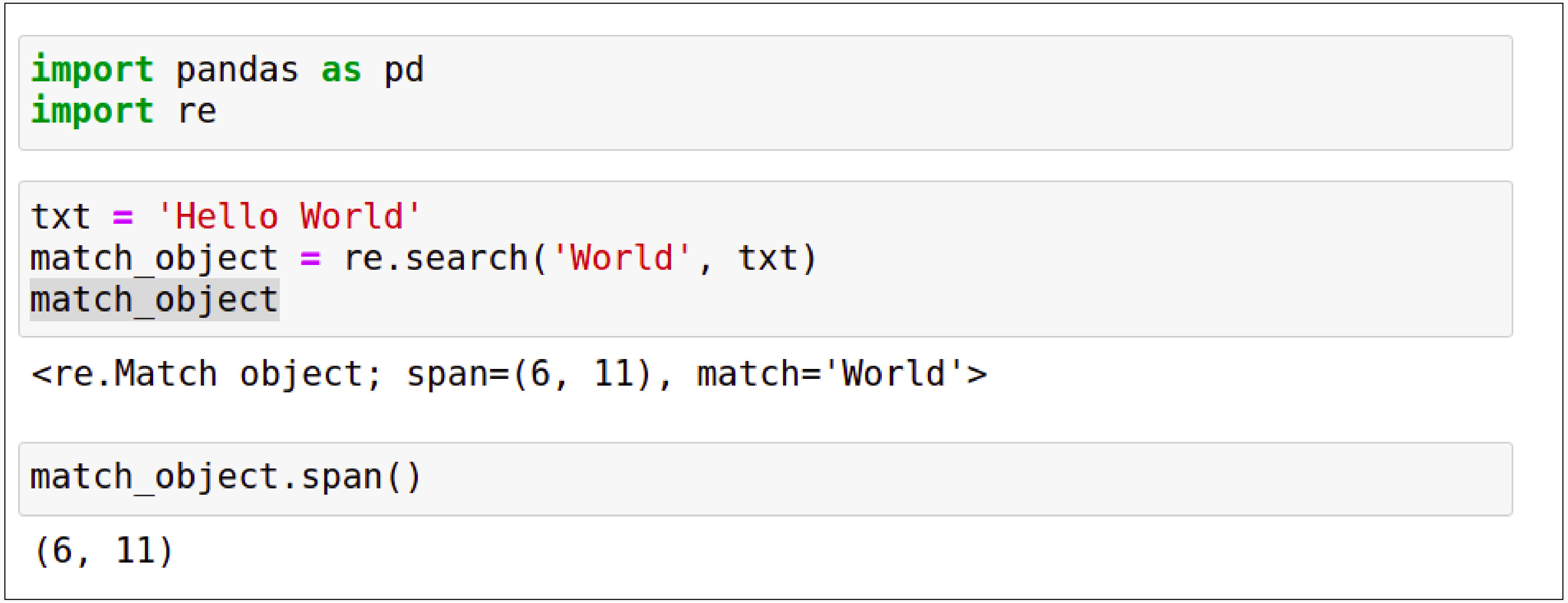
```
re.findall('Pakistan', txt.to_string())
```

```
['Pakistan', 'Pakistan']
```

Regular Expressions (6/8)

re.search()

- re.search() returns a Match Object in case of a pattern match in the string.
- We can get the position of the match using the .span() method of the Match Object.



```
import pandas as pd
import re
```

```
txt = 'Hello World'
match_object = re.search('World', txt)
match_object
```

```
<re.Match object; span=(6, 11), match='World'>
```

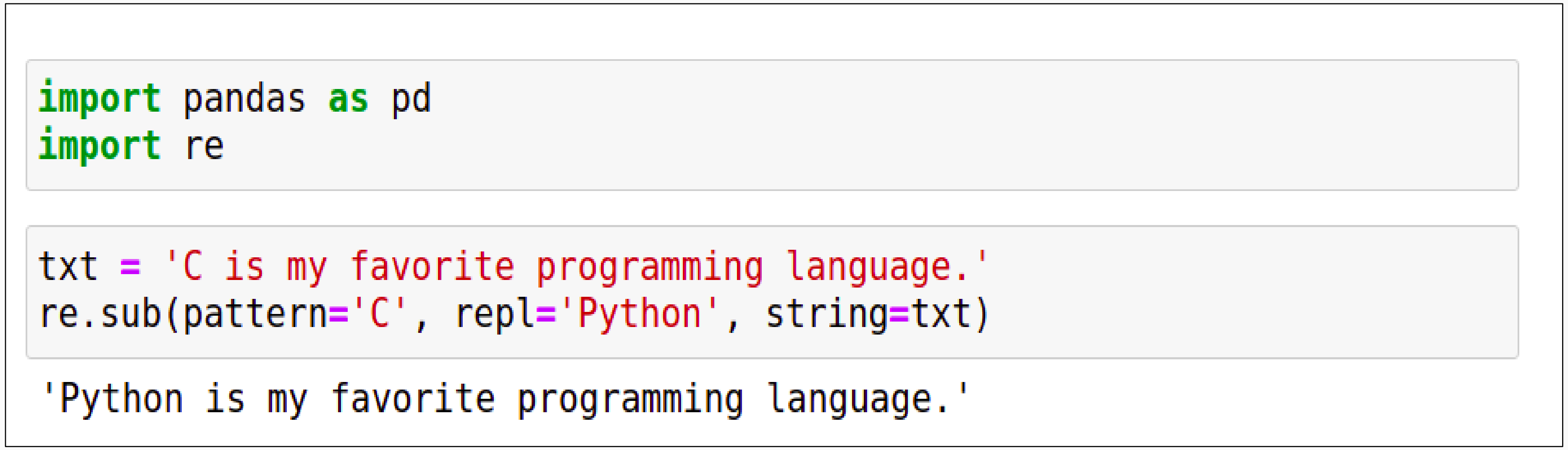
```
match_object.span()
```

```
(6, 11)
```

Regular Expressions (7/8)

re.sub()

- To replace text in a string with a different text, use the re.sub() method.



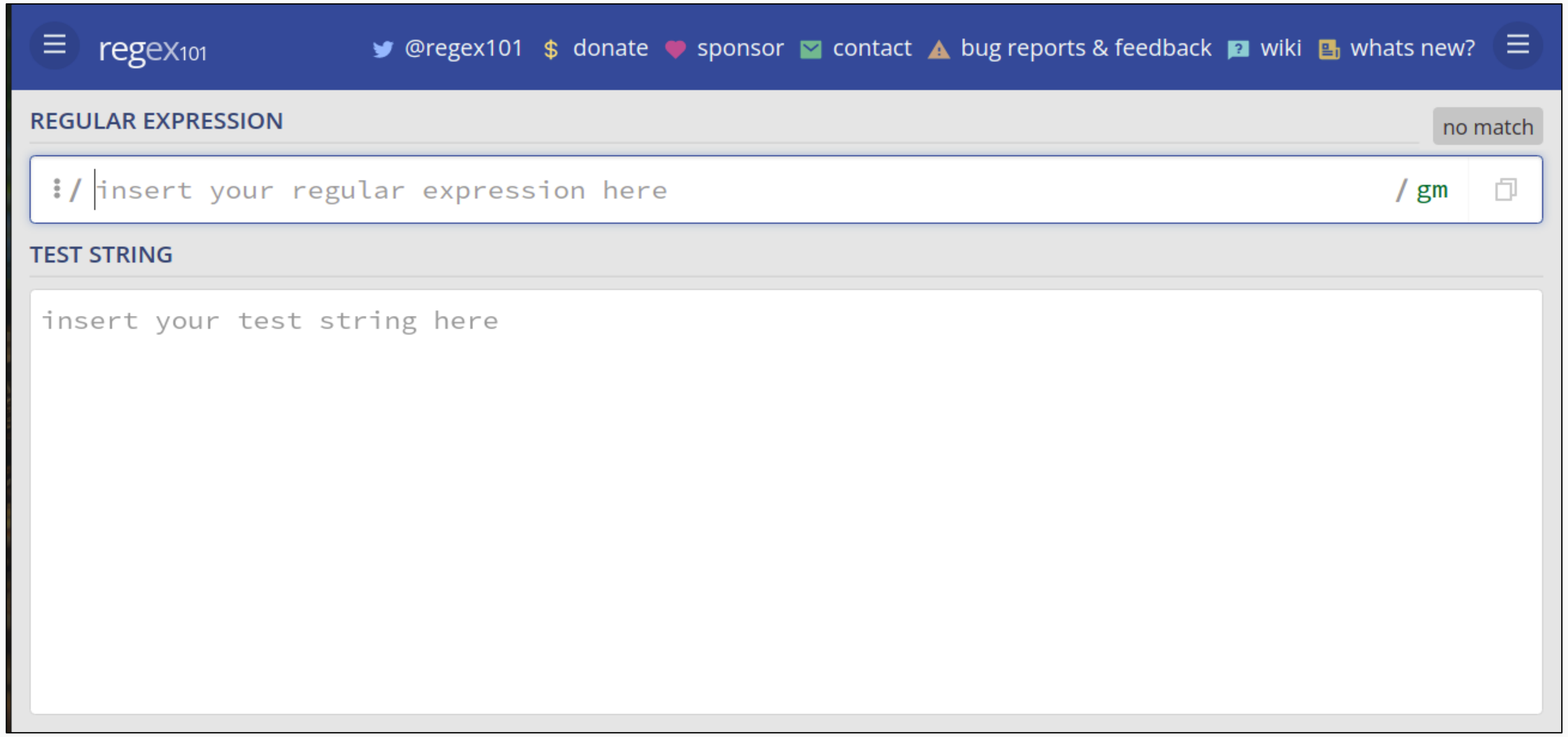
```
import pandas as pd
import re
```

```
txt = 'C is my favorite programming language.'
re.sub(pattern='C', repl='Python', string=txt)
```

```
'Python is my favorite programming language.'
```

Regular Expressions (8/8)

- https://regex101.com/ is a very good website to create and test regular expressions.
- We have also added some resources to learn more about regular expression in Python in the 'Resources slide'.



REGULAR EXPRESSION

no match

/ insert your regular expression here / gm

TEST STRING

insert your test string here

Resources

- https://www.w3schools.com/python/pandas/pandas_cleaning.asp
- https://www.geeksforgeeks.org/interquartile-range-to-detect-outliers-in-data/
- https://www.kdnuggets.com/2021/04/data-science-101-normalization-standardization-regularization.html
- https://www.w3schools.com/python/python_regex.asp

Meta Brains