Artificial Intelligence

BS (CS) _SP_2024

Lab_02Manual

Learning Objectives:

1. Lambda Functions and Regular Expressions
2. Introduction to Numpy
3. Introduction to Pandas
4. Matplotlib

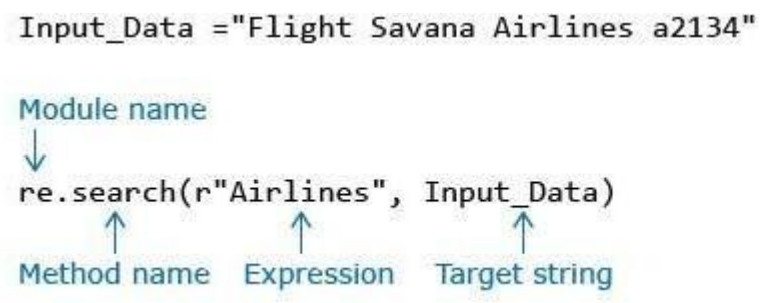
Lab Manual

Regular expression:

Many times a lot of data would be stored in files and we may have to pick and change only relevant portions from a file. Even though there are string functions that allow us to manipulate strings, when dealing with more complicated requirements, we would need more powerful tools.

Regular expressions (regex) in Python provide a powerful way to search, manipulate, and validate strings based on specific patterns. The re module in Python allows you to work with regular expressions.

Python has a module named 're' for regular expressions.

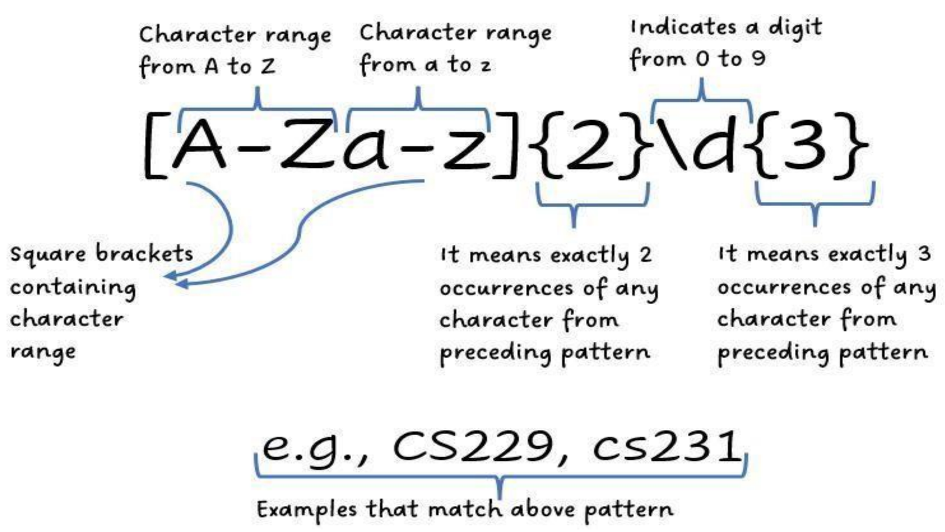


```
Input_Data ="Flight Savana Airlines a2134"

re.search(r"Airlines", Input_Data)
```

Character	Description	Example
[]	A set of characters	"[a-m]"
\	Signals a special sequence (can also be used to escape special characters)	"\d"
.	Any character (except newline character)	"he..o"
^	Starts with	"^hello"
$	Ends with	"world$"
*	Zero or more occurrences	"aix*"
+	One or more occurrences	"aix+"
{}	Exactly the specified number of occurrences	"al{2}"
|	Either or	"falls|stays"
()	Capture and group	

Example:


Character range from A to Z
Character range from a to z
Indicates a digit from 0 to 9
[A-Za-z]{2}\d{3}
Square brackets containing character range
It means exactly 2 occurrences of any character from preceding pattern
It means exactly 3 occurrences of any character from preceding pattern
e.g., CS229, cs231
Examples that match above pattern

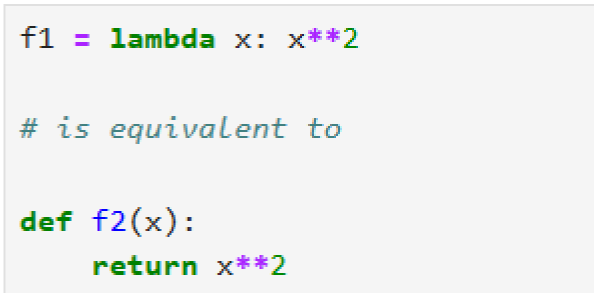
Lambda Functions:

Lambdas are functions without names, in other words they are anonymous functions. They take inputs and return outputs but do not have a name. They are shortcuts to create simple temporary functions.

Syntax:

lambda arguments : expression

Example:



```python
f1 = lambda x: x**2

# is equivalent to

def f2(x):
    return x**2
```

Lambda Functions(regular expression basis):

In the context of regular expressions (regex), lambda functions can be used in languages that support them to define custom behavior for matching patterns. For example, in Python, you can use lambda functions with the re module to perform regex operations.

Example:

```
import re
# List of words words = ["apple", "banana", "cherry", "date"]
# Filter words that start with 'a' filtered_words = filter(lambda x: re.match("^a", x), words)
# Print the filtered words print(list(filtered_words))
# Output: ['apple']
```

Introduction to NumPy:

Numpy:

Numpy is a python library that deals with numerical data and arrays. Python lists same purpose and arrays but numpy arrays are far faster than the traditional python lists. The array object in nunpy is called ndarray which is created by np.array()

1. **Install numpy:**

```
pip install numpy
```

2. **Import numpy:**

```
import numpy as np
```

3. **Create numpy array:**

```
arr1d = np.array([1, 2, 3, 4, 5])      #creating a 1D array

arr2d = np.array([[1, 2, 3], [4, 5, 6]]) #creating a 2D array

arr3d = np.array([[[1, 2, 3],          #creating a multidimensional array
                  [4, 5, 6]],
                  [[7, 8, 9],
                  [10, 11, 12]]])
```

Different Numpy Operations:

`np.shape(array)`	Find out shape of the array
`np.ndim(array)`	Dimension of the array
`np.size(array)`	Size of the array
`np.reshape(array)`	Reshaping the array
`np.mean(array)`	To find the mean of array
`np.sum(array)`	Sum of the values of array
`np.max(array)` `np.min(array)`	Maximum and minimum value in array with specified axis
`np.sort(array)`	Sorting array in ascending order

Accessing and Slicing:

Access through their index:

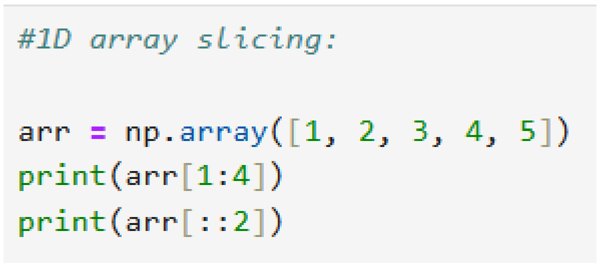
```
print(arr1d[1]) #1D array
print(arr2d[0, 1]) #2D array
print(arr3d[0, 1, 2]) #3D array
```

Slicing:

Slicing general pattern goes like: [start : end].

We can also define the step, like this: [start : end : step].

- If we don't pass start it considered 0
- If we don't pass end its considered length of array in that dimension.
- If we don't pass step it considered 1



```
#1D array slicing:

arr = np.array([1, 2, 3, 4, 5])
print(arr[1:4])
print(arr[::2])
```

Pandas

Pandas is a powerful Python library for data manipulation and analysis. It provides data structures like Series and DataFrame, which are designed for handling structured data efficiently.

Installing pandas:

If you haven't installed pandas yet, you can do so using pip:

pip install pandas

Importing pandas:

In your Python script or notebook, import pandas as follows:
import pandas as pd

Creating Pandas Dataframe:

data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'], 'Age': [25, 30, 35, 40]}

df = pd.DataFrame(data)

- Saving my dataframe into a csv file: df.to_csv("Abc.csv")
- Reading csv and storing it into dataframe: df1=pd.read_csv("Abc.csv")
- Assign values to a specific column: values=[18,19,20,18,17,17,18]
  df["age"]=values
- Use to select specific rows by their index labels: df.loc[[4]]
- Display details of specific column: df["column_name"]
- Consice summary of dataframe: df.info()
- Display first 5 elements of the dataframe: df.head(5)
- Statistical details of the dataframe: df.describe()

MatplotLib

Matplotlib is a popular Python library for creating static, animated, and interactive visualizations. It provides a wide range of plotting functions and customization options for creating high-quality plots.

Installing Matplotlib:

If you haven't installed matplotlib yet, you can do so using pip: pip install matplotlib

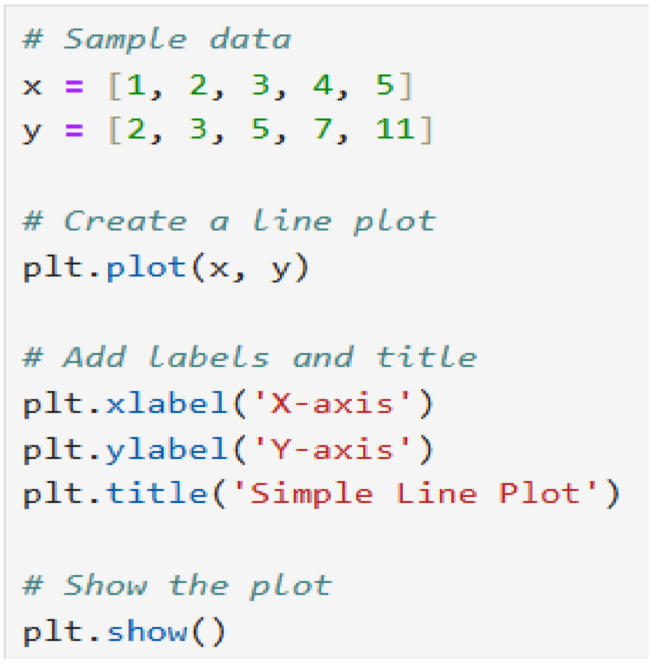
Importing Matplotlib:

In your Python script or notebook, import matplotlib's pyplot module, which provides a MATLABlike interface for plotting:

import matplotlib.pyplot as plt

Simple Line Plot:

Use the plot function to create a simple line plot:



```python
# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]

# Create a line plot
plt.plot(x, y)

# Add labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Simple Line Plot')

# Show the plot
plt.show()
```

Customizing Plots:

You can customize various aspects of the plot, such as colors, line styles, markers, and more:

```
# Customize the plot with color, line style, and marker
plt.plot(x, y, color='red', linestyle='--', marker='o', label='Data')
```

Multiple Plots:

You can create multiple plots in the same figure using subplot:

```
# Create two subplots
plt.subplot(1, 2, 1) # (rows, columns, plot_number)
plt.plot(x, y)
plt.title('Plot 1')


plt.subplot(1, 2, 2)
plt.plot(y, x)
plt.title('Plot 2')
# Adjust layout
plt.tight_layout()
```

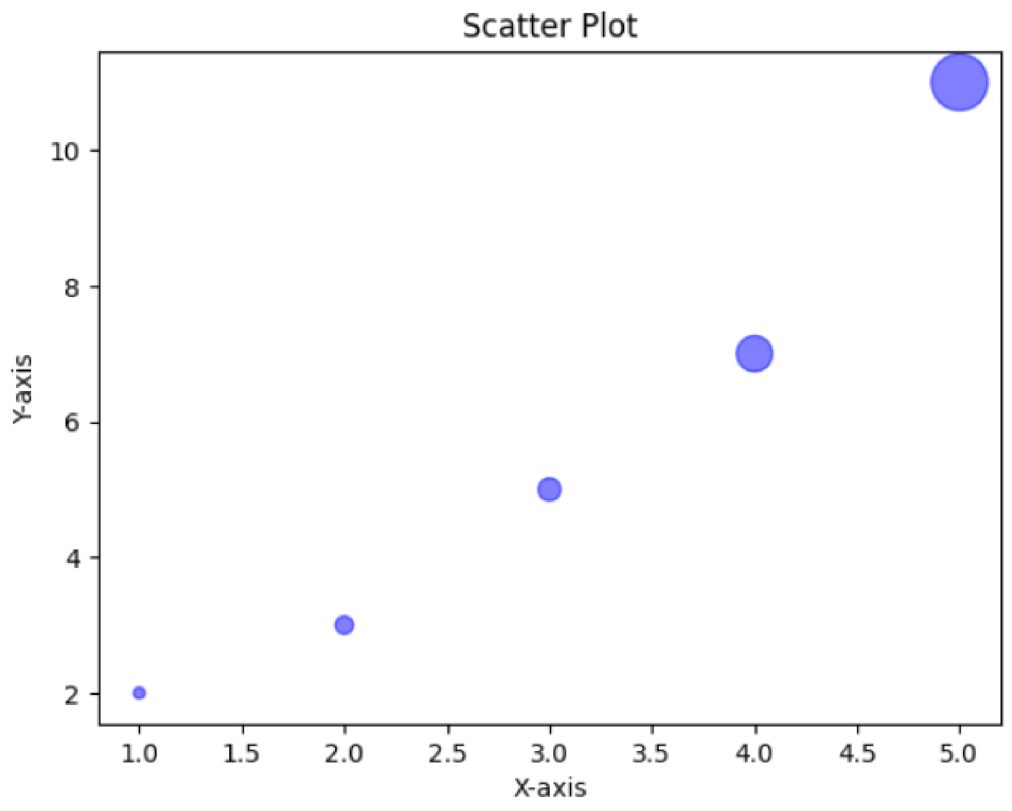
Scatter Plot

```
# Sample data for scatter plot
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]
sizes = [20, 50, 80, 200, 500]


# Create a scatter plot
plt.scatter(x, y, s=sizes, c='blue', alpha=0.5, label='Data')


# Add labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot')
```

```
Text(0.5, 1.0, 'Scatter Plot')
```



Other Plots

- **Bar Plot:** plt.bar(x, height)
- **Histogram:** plt.hist(data, bins)
- **Box Plot:** plt.boxplot(data)
- **Scatter Plot:** plt.scatter(x, y)
- **Pie Chart:** plt.pie(data, labels)
- **Heatmap:** sns.heatmap(data) # using seaborn library