DATA SCIENCE: CAREER OF THE FUTURE

# INTRODUCTION TO DATA SCIENCE

SANJAY RAJVANSHI

# SCHEDULE


| Session | Date | Time | Topic |
|---|---|---|---|
| 1 | Sep 25 | 7:00 pm – 8:00 pm | Introduction to data science and associated tools. |
| 2 | Oct 2 | 7:00 pm – 8:00 pm | Introduction to Python. Learn how to use Python for data analysis. Python is simple, yet powerful language that is often used in data science. |
| 3 | Oct 9 | 7:00 pm – 8:00 pm | Data wrangling with Python. Learn how to gather data and make it useful for analysis. |
| 4 | Oct 16 | 7:00 pm – 8:00 pm | Data visualization and analysis with Python. Learn how to create useful visualizations to aid in the analysis of the data. |
| 5 | Oct 23 | 7:00 pm – 8:00 pm | Brief introduction to artificial intelligence and machine learning. Get a peek into how to make data based predictions. |

Note: All classes are on Wednesdays.

# SESSION 1 – RECAP


- Data Science Background
- Environment Setup
- References
- Examples
- Exercises


# SESSION 2: INTRODUCTION TO PYTHON

# INTRODUCTION TO PYTHON


- Data Types
  - Numbers
  - Strings
  - Lists
- Variables, Assignments, Operations
- Control Flow Statements
  - **if** statement
  - **for** statement
  - **while** statement
  - Remember to indent
- Modules
  - import
- Comments
- Packages
  - pandas
  - numpy
  - pyplot
  - seaborn
  - mplot3d

# DATA TYPES


- Numbers
  - int
    - 1, 5, 43, 100…..
  - Float
    - 1.0, 2.0, 4.5, 5.7, 50.9 …..
  - Other ones, FYI only for now
    - Decimal, Fraction, complex numbers
- Strings
  - Enclose in "like this " or 'like this'
  - "I live in N Potomac"
  - 'I live in N Potomac'

- Be careful of special characters like "\" as they have a special meaning
- Use **+** to concatenate two strings
- Index with 1st character as index 0
- Negative index starts from the right
- Slicing allows extracting substrings
- Can not be modified
- **len (***string***)** returns the string length

# DATA TYPES


- Lists
  - [1, 5, 43, 100]
  - ['p', 'y', 't', 'h', 'o', 'n']
  - First position is referenced by 0, not 1
  - Indexing and slicing works like strings
  - Lists can be modified using indexing, slicing, **append ()**,
  - **len (***list***)** returns the list length
  - FYI only for now, lists can be nested

# OPERATORS


- Arithmetic Operators
  - **+** : addition of numbers, concatenation of strings
  - **-** : subtraction
  - ***** : multiplication
  - **/** : division
  - **//** : Floor division (returns only integer result)
  - **%** : modulus (calculates remainder in division)
  - ****** : Exponentiation (to the power of)
- Comparison Operators
  - **==** : equal to
  - **<** : less than
  - **>** : greater than
  - **<=** : less than or equal to
  - **>=** : greater than or equal to
  - **!=** : not equal to
- Assignment Operators
  - **=** : assignment
  - **+=** : addition and assignment

# OPERATORS, COMMENTS


- Assignment Operators (contd.)
  - **-=** : subtraction and assignment
  - ***=** : multiplication and assignment
  - **/=** : division and assignment
  - **%=**, **//=**, ****=** are some of the others
- Logical Operators
  - **and** : returns True if both sides are true
  - **or** : returns True if one of the sides is true
  - **not** : negates the results
- Other Operators
  - Identity
  - Membership
  - Bitwise
- Comments
  - Adding information to clarify code
  - Comments are not executed
  - Multiple ways to add comment

```
# This is one way to add a comment
x = 5 # This is second way
      # This is the third way
```

# VARIABLES, ASSIGNMENTS, OPERATIONS


- x = 43
- y = 50.9
- z = x + y
- z += 5
- aString = "I live in N Potomac"
- bString = aString + ", MD "
- aString [0:5]
- **len** (aString)
- aString [-5:-1]
- aString [-7:]

- aList = [1, 5, 43, 100]
- anotherList = ['p', 'y', 't', 'h', 'o', 'n']
- **len** (anotherList)
- anotherList [0:3]
- anotherList [-4:]
- anotherList.append ('is great')

# CONTROL FLOW


- **if** statement
  - Used for decision making
  - Example:

```
age = 43
if age < 18:
        canVote = 'No'
else:
        canVote = 'Yes'
print (canVote)
```

  - Multiple conditions? Use **elif**

- **for** statement
  - Used for going through a list or performing operation(s) in a loop for a defined number of steps
  - Example:

```
ageList = [5, 17, 18, 27, 43, 55]
for i in ageList:
        if i < 18:
                canVote = 'No'
        else:
                canVote = 'Yes'
        print (i, canVote)
```

# CONTROL FLOW


- **while** statement
  - Used for performing operation(s) in a until a condition remains true
  - Example:

```
#Sums expenses for Jan-Mar
expenseList = [509.50, 1019.43, 1527.22]
i = 0
totalExpense = 0
while i < 3:
        totalExpense = totalExpense +
                       expenseList [i]
        i = i + 1
print (totalExpense)
```

- Few other things:
  - Indentation is important
  - **range** () function
  - Example:

```
ageList = [5, 17, 18, 27, 43, 55]
print (ageList [1])
ageList  [1] += 2
print (ageList [1])
ageList [1] = ageList [1] ** 2
print (ageList [1])
ageList.append (60)
print ( ageList)
```

# PACKAGES/LIBRARIES


- Packages
  - Useful to build and add new capabilities to Python language
  - Also referred to as Libraries
  - Key relevant examples:
    - pandas
      - Example – IO tools (like read_csv)
    - numpy
      - Example – N-dimensional array
    - matplotlib, pyplot
      - Example – Bar graph
    - seaborn
      - Example – Scatter plot
    - mplot3d
      - Example – 3D graphs

# INTRODUCTION TO PYTHON


| Package | Description | Website |
|---|---|---|
| -- | Official website for Python | https://www.python.org/ |
| -- | Another good Python reference | https://www.w3schools.com/python/default.asp |
| pandas | Open source library providing data structure and data analysis tools | https://pandas.pydata.org/ |
| numpy | Fundamental package for scientific computing with Python | https://numpy.org/ |

# EXERCISE-1: IF AND FOR STATEMENTS


- Create a Python file with name "S2-Ex1"
- Type and execute the following code:

```
ageList = [5, 11, 17, 18, 27, 43, 55, 65, 67]
for i in ageList:
        if i < 18:
            canVote = 'No'
        else:
            canVote = 'Yes'
        print (i, canVote)
```

- Now add more classification to this exercise
  - Count and print the number of children (0-12 years), teenagers (13-17 years), adults (18-59 years), and senior adults (60 years and above).

# EXERCISE-2: IF AND FOR STATEMENTS


- Create a Python file with name "S2-Ex2"
- Now try another example where you are given score/marks 13 students got in class and you have to
  - assign a letter grade (A for 90-100, B for 70-89, C for 50-69, D for 30-49, E for 10-29, F for < 10)
  - compute the average of all the scores/marks
- Assume the following list of scores/marks is given to you:
  - marks = [7, 11, 29, 30, 50, 57, 69, 75, 88. 89, 90, 92, 97]

# EXERCISE-3: **WHILE** STATEMENT


- Create a Python file with name "S2-Ex3"
- Type and execute the following code:

```
#Sums expenses for Jan-Mar
expenseList = [509.50, 1019.43, 1527.22]
i = 0
totalExpense = 0
while i < 3:
    totalExpense = totalExpense + expenseList [i]
    i = i + 1
print (totalExpense)
```

- Another exercise: Assume you are a soccer player who played 7 games in a championship. Add all the goals you have scored. Use **range** () and **len** () functions.

# EXERCISE-4: BINARY SEARCH


- Create a Python file with name "S2-Ex4" to implement binary search algorithm.
- Binary search is used to search for an element in an array sorted in an increasing order.
- Type and execute the following code:

```
idList = (2, 5, 10, 17, 20, 29, 33, 49, 51)

searchForItem = 20 # Try different items

i = 0

j = len (idList) - 1

found = False

while i <= j:

    k = (i + j)//2
```

# EXERCISE-4: BINARY SEARCH


```
        if idList [k] < searchForItem:
                i = k + 1
        elif idList [k] > searchForItem:
                j = k - 1
        else:
                found = True
                break
if (found == True):
        print ('Item found at position: ', k)
else:
        print ('Item not found')
```

# EXERCISE-5: CREATE A SIMPLE BAR GRAPH


- Create a Python file with name "S2-Ex5"
- This is a sample from pyplot library
- Type and execute the following code:

```
y = (17, 20, 15, 17, 15)
x = ('Class 1', 'Class 2', 'Class 3', 'Class 4', 'Class 5')
plt.xlabel ('Class')
plt.ylabel ('Attendance')
plt.title ('Class Attendance')
plt.bar (x, y)
```

- Share your observations. Any thoughts on why you got an error?

# EXERCISE-5: CREATE A SIMPLE BAR GRAPH


- We also need to add the libraries to be able to use the plot capabilities to fix the error.
- Create a new cell before the cell in which you entered the code on previous slide.
- Now insert the following code in that new cell:

```
import matplotlib

import matplotlib.pyplot as plt

import numpy as np
```

- Now execute the code in both the cells in sequence
- You should be able to see the bar graph now (you may have to click on "Run" more than once to see the graph – please contact the instructor if the graph doesn't show after repeated clicks on "Run").

# SESSION 2 – RECAP


- Data Types
- Control Flow Statements
- Packages/Libraries
- Introduced Plots
- Exercises (reach out to the library or the instructor for solution to the exercises if needed)

# SESSION 2 – HOME WORK


- Do more Python programming  practice relating to this session.
  - Refer to the following websites:
    - https://docs.python.org/3/tutorial
    - https://www.w3schools.com/python/
  - Experiment with different data types.
  - Experiment with all the control flow statements.
  - Experiment with additional items you find in the above tutorials.
  - Write a program to implement Bubble sort.
  - Now try the binary search we implemented earlier on the data you sorted using Bubble sort.

# SESSION 3 – AGENDA


- Data wrangling with Python. Learn how to gather data and make it useful for analysis.
- Learn how to use Python for data analysis. We will start to learn how to make the data suitable for the problem, clean/convert/transform it – sometimes referred to as data wrangling or data munging.
- Specifically we will focus on DataFrames, large amount of data, and how to analyze that.

# SESSION 3 – PRE-WORK


- Familiarize with pandas library (https://pandas/pydata.org)
- It provides two primary data structures:
  - Series (1-dimensional)
  - DataFrame (2-dimensional)
- Review and try examples/code from the following:
  - Intro to data structures (https://pandas.pydata.org/pandas-docs/stable/getting_started/dsintro.html )
  - 10 minutes to pandas (https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html)
  - Try Cookbook on pandas website (https://pandas.pydata.org/pandas-docs/stable/user_guide/cookbook.html#cookbook)

# SESSION 3 – PRE-WORK


- Make a copy of S1-Ex2 to see some of the capabilities of DataFrames. Name the new file S3-Ex1.
- Also, make a new copy of the data file and name it "S3-Ex1-US-Presidents.csv" – change the first heading from "No." to "Number" and save the data file.
- Now modify the S3-Ex1 to read from the new data file "S3-Ex1-US-Presidents.csv".
- Execute all of the code in S3-Ex1.
- Now add the following lines, each line in a new cell. Execute each cell.

```
listOfPresDf.columns

listOfPresDf.count ()

listOfPresDf ['Name']

listOfPresDf ['State'] = ''
```

# SESSION 3 – PRE-WORK


listOfPresDf

listOfPresDf.dtypes

listOfPresDf = listOfPresDf.append ({'Number': 11, 'Name': 'James K. Polk', 'Term': '1845-1850', 'State': ''}, ignore_index = True)

listOfPresDf.loc [listOfPresDf.Number == 11, 'Term'] = '1845-1849'

listOfPresDf.head (20)

listOfPresDf.drop (columns = ['State'])

listOfPresDf.head (20)

# SESSION 3 – PRE-WORK


- Explore large data sets and pick one per your interest:
  - Montgomery County, MD data sets – https://data.montgomerycountymd.gov/
    - Download *Montgomery College Enrollment Data* from https://data.montgomerycountymd.gov/Education/Montgomery-College-Enrollment-Data/wmr2-6hn6
  - US Govt. open data sets – https://www.data.gov/
  - Non Govt. website with lots of data sets – https://www.kaggle.com/
  - **Pay attention to the licensing terms before downloading**
  - You may contact the library or the instructor for any help in identifying data set(s) you might be looking for or for any other questions related to the data set(s).

# REFERENCES


*Note: you are not required to sign-up for an account on any of the sites to read these articles.*

1. *Official website for Python and tutorials* – https://www.python.org/
2. *Another good Python reference and tutorials* – https://www.w3schools.com/python/default.asp
3. *pandas (Open source library providing data structure and data analysis tools)* – https://pandas.pydata.org/
4. *numpy (Fundamental package for scientific computing with Python)* – https://numpy.org/