Artificial Intelligence Lab

CSL 411

Lab Journal 2

TAUHEED BUTT
01-134191-068
BSCS-6B

Department of Computer Science
BAHRIA UNIVERSITY, ISLAMABAD

Lab # 2: Lists, Tuples, Set, Dictionary, Numpy & Pandas

Objectives:

To learn about different data structures in python and how to use them.

Tools Used:

IDLE (Python 3.4 GUI Python)

Submission Date:

Evaluation: **Signatures of Lab Engineer:**

Task # 1:

Create list of Fibonacci numbers after calculating Fibonacci series up to the number n which you will pass to a function as an argument. The number n must be input by the user.

Fibonacci numbers are calculated using the following formula: The first two numbers of the series are always equal to 1, and each consecutive number returned is the sum of the last two numbers.

Hint: Can you use only two variables in the generator function?

The code below will simultaneously switch the values of a and b.

```
a = 1
b = 2
a, b = b, a
```

The first number in the series should be 1. (The output will start like 1,1,2,3,5,8,…)

Procedure/Program:

```python
def fibonacci(n):
    if(n<0): raise ValueError(f"{n} terms not possible for Fibonacci Series")
    elif(n==0): return []
    elif(n==1): return [1]
    elif(n==2): return [1,1]
    first = 1
    second = 1
    series = [1,1]
    for i in range(n):
        sum = first + second
        series.append(sum)
        first,second = second,sum
    return series

n = int(input("Enter length of Fibonacci Series: "))
print(f"Series: {fibonacci(n)}")
```

Result/Output:

```
PS C:\Users\Tauheed\OneDrive\Documents\UNI Codes> python -u "c:\Users\Tauheed\On
eDrive\Documents\UNI Codes\AI\Lab3\tempCodeRunnerFile.py"
Enter length of Fibonacci Series: 10
Series: [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144]
PS C:\Users\Tauheed\OneDrive\Documents\UNI Codes> python -u "c:\Users\Tauheed\On
eDrive\Documents\UNI Codes\AI\Lab3\tempCodeRunnerFile.py"
Enter length of Fibonacci Series: 1
Series: [1]
PS C:\Users\Tauheed\OneDrive\Documents\UNI Codes> python -u "c:\Users\Tauheed\On
eDrive\Documents\UNI Codes\AI\Lab3\tempCodeRunnerFile.py"
Enter length of Fibonacci Series: -2
Traceback (most recent call last):
  File "c:\Users\Tauheed\OneDrive\Documents\UNI Codes\AI\Lab3\tempCodeRunnerFile
.py", line 16, in <module>
    print(f"Series: {fibonacci(n)}")
  File "c:\Users\Tauheed\OneDrive\Documents\UNI Codes\AI\Lab3\tempCodeRunnerFile
.py", line 2, in fibonacci
    if(n<0): raise ValueError(f"{n} terms not possible for Fibonacci Series")
ValueError: -2 terms not possible for Fibonacci Series
PS C:\Users\Tauheed\OneDrive\Documents\UNI Codes>
```

Task # 2:

Write a program that lets the user enter in some English text, then converts the text to Pig-Latin. To review, Pig-Latin takes the first letter of a word, puts it at the end, and appends “ay”. The only exception is if the first letter is a vowel, in which case we keep it as it is and append “hay” to the end. For example: “hello” -> “ellohay”, and “image” -> “imagehay”
It will be useful to define a list or tuple at the top called VOWELS. This way, you can check if a letter *x* is a vowel with the expression x in VOWELS.
It’s tricky for us to deal with punctuation and numbers with what we know so far, so instead, ask the user to enter only words and spaces. You can convert their input from a string to a list of strings by calling split on the string:
“My name is John Smith”.split(“ ”) -> [“My”, “name”, “is”, “John”, “Smith”]

Procedure/Program:

```
def pig_latin(string):
    string = string.split(" ")
    vowels = ['a','e','i','o','u']
    for i in range(len(string)):
        if string[i][0].lower() in vowels:
            string[i] += "hay"
        else:
            string[i] += f"{string[i][0]}ay"
            string[i].replace(string[i][0], "")
    return string


string = input("Enter Sentance: ")
updated = pig_latin(string)
print(updated)
```

Result/Output:

```
PS C:\Users\Tauheed\OneDrive\Documents\UNI Codes> python -u "c:\Users\Tauheed\On
eDrive\Documents\UNI Codes\AI\Lab3\task_2a_2.py"
Enter Sentance: Hello there Image umbrella
['HelloHay', 'theretay', 'Imagehay', 'umbrellahay']
PS C:\Users\Tauheed\OneDrive\Documents\UNI Codes>
```

Task # 3:

Write a Pandas/Numpy program to find the index of the first occurrence of the smallest and largest value of a given series

Procedure/Program:

```
import numpy as np
import pandas as pd
import random

# create random list in range 1 - 5
s = pd.Series([random.randrange(1, 5, 1) for i in range(10)])
print(f'Series\n{s}')
print(f'Smallest: {s.idxmin()}')
print(f'Largest: {s.idxmax()}')
```

Result/Output:

```
Series
0    3
1    3
2    2
3    3
4    2
5    2
6    3
7    3
8    2
9    3
dtype: int64
Smallest: 2
Largest: 0
```

Task # 4:

Write a Pandas program to compute the Euclidean distance between two given series.

Euclidean distance
From Wikipedia, In mathematics, the Euclidean distance or Euclidean metric is the "ordinary" straight-line distance between two points in Euclidean space. With this distance, Euclidean space becomes a metric space.

Series-1: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Series-2: [11, 8, 7, 5, 6, 5, 3, 4, 7, 1]

Procedure/Program:

```
import numpy as np
import pandas as pd
from math import *
import random

def euclidean_dist(p, q):
    return sqrt((sum([pow((q[i]-p[i]), 2) for i in range(len(q))])))


q = pd.Series([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
p = pd.Series([11, 8, 7, 5, 6, 5, 3, 4, 7, 1])

print(f"Euclidean distance: {euclidean_dist(p, q)}")
```

Result/Output:

```
PS C:\Users\Tauheed\OneDrive\Documents\UNI Codes> python -u '
Euclidean distance: 16.492422502470642
PS C:\Users\Tauheed\OneDrive\Documents\UNI Codes>
```

Task # 5:

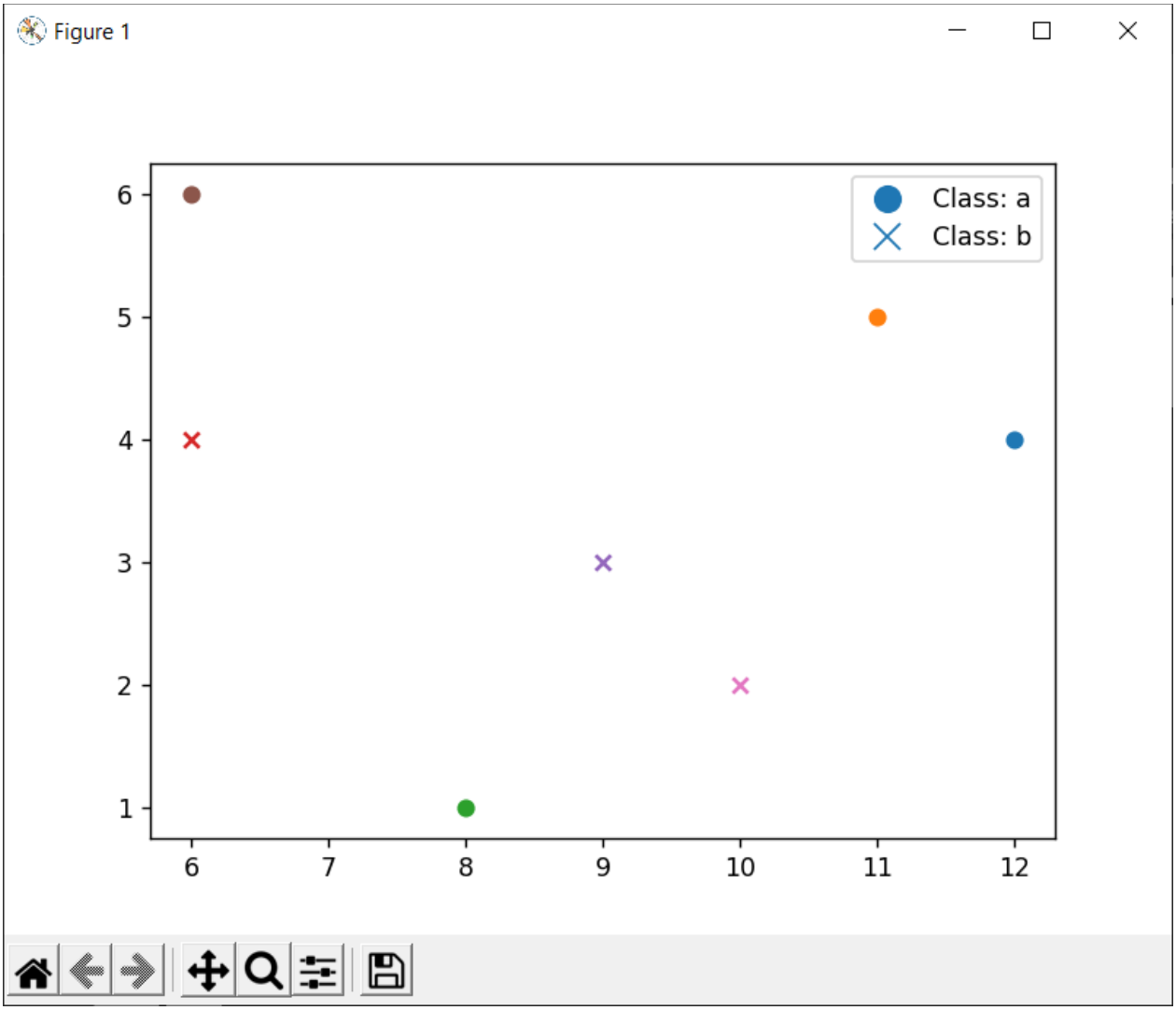
Visualize the following data in python. Please provide the reason for the choice of graph.

Feature 1	Feature 2	Class
12	4	a
11	5	a
8	1	a
6	4	b
9	3	b
6	6	a
10	2	b

Procedure/Program:

```
import matplotlib.pyplot as plt
import matplotlib.lines as mlines

# Data
feature1 = [12, 11, 8, 6, 9, 6, 10]
feature2 = [4, 5, 1, 4, 3, 6, 2]
classes = ['a', 'a', 'a', 'b', 'b', 'a', 'b']

# Add Legend
a = mlines.Line2D([], [], marker='o', linestyle='None',
                          markersize=10, label='Class: a')
b = mlines.Line2D([], [], marker='x', linestyle='None',
                          markersize=10, label='Class: b')
plt.legend(handles=[a,b])

# Scatter the Graph
for i in range(len(classes)):
    marker='o'
    if classes[i]=='b':
        marker='x'
    plt.scatter(feature1[i], feature2[i], marker=marker)

plt.show()
```

Result/Output:



Analysis:

The reason a scatter graph was chosen for the given data is because of the variance it holds. Each point has a class associated with it. Therefore, each class was given a marker which can be seen in the legend. This way, we can properly visualize our given data on the graph.