***Artificial Intelligence Lab***

***CSL 411***

***Lab Journal 2***

****

**AHMAD HASSAN**

**01-134191-002**

**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:**

def fib(n):

    if n == 0:

        return 0

    elif n == 1:

        return 1

    else:

        return (fib(n-1)+fib(n-2))

n = int(input("\nEnter value of N for Fibonacci : "))

seq = list()

for i in range(1, n+1):

    seq.append(fib(i))

print(f"\nFor N = {n}  :  Sequence = {seq}\n")

**Result/Output:**

Table

Description automatically generated with medium confidence

**Analysis/Conclusion:**

In above task we learn how to make Fibonacci series in python

**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:**

sentence = input("\nEnter any sentence : ")

newList = sentence.split(" ")

finalList = []

vowel = ['a', 'e', 'i', 'o', 'u']

for word in newList:

    if word[0].lower() in vowel:

        word = word+'hay'

    else:

        temp = ""

        length = len(word)

        for i in range(1, length):

            letter = word[i-length]

            temp += letter

        temp += word[0]

        word = temp+"ay"

    finalList.append(word)

print(f"\nOutput : {finalList}\n")

**Result/Output:**

**Graphical user interface, text

Description automatically generated with medium confidence**

**Analysis/Conclusion:** Python has easiest logic

**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 pandas as pn

import numpy as np

randomList=pn.DataFrame(np.random.randint(100,size=10))

print(f"List : {randomList}\n Index of : \nSmallest: {randomList.idxmin()}\nLargest: {randomList.idxmax()}")

**Result/Output:**

**Chart, text

Description automatically generated**

**Analysis/Conclusion:**

**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 pandas as pn

from math import sqrt

def euclidian(list1,list2):

    distance=0

    for i in range(0,len(list1)):

        distance += (list1[i]-list2[i])\*\*2

    return sqrt(distance)

Series\_1 = pn.Series([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

Series\_2 = pn.Series([11, 8, 7, 5, 6, 5, 3, 4, 7, 1])

print(f"\nSeries 1:\n{Series\_1.to\_string(index=False)} \n\nSeries 2:\n \n{Series\_2.to\_string(index=False)}")

print(f"\nEuclidian Distance = {euclidian(Series\_1,Series\_2)}\n")

**Result/Output:**

**Table

Description automatically generated with medium confidence**

**Analysis/Conclusion:**

**Euclidian distance is easy to find by the help of python**

**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 numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

dataset = pd.read\_csv('dataset.csv')

plt.scatter(dataset['Feature 1'], dataset['Class'], alpha=0.5)

plt.scatter(dataset['Feature 2'], dataset['Class'], alpha=0.5)

plt.title('Feature VS Class')

plt.xlabel('Feature 1, Feature 2')

plt.ylabel('Class')

plt.show()

**Result/Output:**

**Chart, scatter chart

Description automatically generated**

**Analysis/Conclusion:**

To establish the relationship between Class and Features scatter Plot is most effective. It is a relationship chart that are suited to show how one variable relates to one or numerous different variables. You could use this to show how something positively effects, has no effect, or negatively effects another variable. For relationship charts you can also use Bubble or Line.