Course: CS5590 Python Programming
Lab Assignment - #1

Report

Team id: #16

Name: Revanth Chakilam | ClassID: 02

Name: Swati Singh | ClassID: 33

Github: https://github.com/revanthchakilam/CS5590_Python/wiki/Lab-Assignment-1

Youtube: https://youtu.be/aOqBqopCMT0

Introduction

This Wiki/Report contains the basic Python programming concepts which includes Object Oriented approaches (Polymorphism, Inheritance, Data Encapsulation), class design, implementing data structures (list, array, tuples, set), random number generation, logical algorithms/problems

Objectives

- The motive is to understand python programming and its features to solve real time problems.

- Task #1: For any real time web applications to login, the user password need to be validated against database rules. So, we designed a sample logic which will validate the provided password against the password creation criteria

- Task #2: Should develop a Python function which will accept a sentence of words from the user and determine the Middle word, Longest word in the sentence and also reverses all the words in the sentence

- Task #3: A python program to display the triplet combinations in the list which gives the sum of zero.

- Task #4: To find the common and uncommon students between two courses lists

- Task #5: Class design using object-oriented approach and its features

- Task #6: Generate a random vector of size 15 and find maximum and minimum values in it

Task 1: Validation of passwords

Approach/Method:

We need to take input from the user and validate the input with the password criteria. For the validation we used regular expressions for pattern matching and loops and conditional statements, user defined functions etc and displaying the success and failure messages

```
password = input("Enter password");

# declare variables
hasNumber = False
hasUpperCase = False
hasLowerCase = False
hasSpecialCharacters = False
hasCorrectLength = True if len(password)> 5 and len(password)< 17 \
    else False
```

```
if hasCorrectLength:
    for temp in password:
        if temp.isupper():
            hasUpperCase = True
        if temp.islower():
            hasLowerCase = True
        if temp.isnumeric():
            hasNumber = True
        if temp in "[$@!*]":
            hasSpecialCharacters = True
else:
    print("Password must have a minimum length of 6 and maximum length of 16")

# validation with the predefined conditions for validating password
if hasSpecialCharacters and hasNumber and hasUpperCase and hasLowerCase:
    print("Password created successfully")
else:
    print("Please check the rules for password criteria")
```

Output:

```
Enter passwordAbcd1234
Please check the rules for password criteria
Password must need a special character [$@!*]
```

Task 2: To allow input from the user and find Middle word, Longest word and also Reverse the words in the sentence

Approach/Method:

Firstly, we allow input sentence from the user and initialize few variables like index, maximum length, words length and declare an empty array for the middle words and declare a list for reversing the sentence. For finding the middle word I divided the words length by 2 and based on the reminder setting the index and retrieving the middle words. For finding the longest word iterating all the words with its lengths and while comparison between the word lengths, I add the longest word to the empty string declared earlier. For reversing the sentence, iterate all the words and reverse each word and append the reversed word to the list

```
# To get middle indexes of middle words
if wordsLen % 2 == 0:
    middleLengths = [wordsLen // 2 - 1, wordsLen // 2]
else:
    middleLengths = [wordsLen // 2]
```

```
  # get middle words
  if len(middleLengths) > middleLengthIndex and index == middleLengths[middleLengthIndex]:
      middleWords.append(word)
    # print(middleWords)
      middleLengthIndex += 1
# print(middleLengthIndex)
  word = word[::-1]
  reverseSentence.append(word)
  index += 1

joining words
verseSentence = ' '.join(reverseSentence)
ddleWords = ' '.join(middleWords)
```

Output:

```
C:\Users\revan\AppData\Local\Programs\Python\Python36-32\python.exe
Enter input sentence:My name is Jacqueline Fernandez Dsouza
Middle words: is Jacqueline
Longest word: Jacqueline
Sentence with reverse words: yM eman si enileuqcaJ zednanreF azuosD
```

Task 3: To find the triplets which sums to zero

Approach/Method:

From Itertools import the combinations package. Declare a list with the input and combination value as 3 such that triplets of 3 numbers are iterated. Define an empty array and initialize a variable index to 0. Now using a range function from 0 to length of the list, iterate through each value and check if sum becomes equal to zero. When the sum is equal to zero append the value from the list based on the index position.

```
inputData = list(combinations([1,3,6,2,-1,2,8,-2,9],3))

for i in range(length): # 0 to 9
  if (sum(inputData[index])==0):  #check if the sum of the cmbination is 0
    tripletsFound.append(inputData[index]) # if a combination triplet is found append
  index+=1 # Increment the index so that the next element can be checked
```

Output:

```
C:\Users\revan\AppData\Local\Pr
Triplets are:  [(3, -1, -2)]
```

Task 4: To find the common and uncommon elements between the two classes/lists

Approach/Method:

Declare the input lists of 2 classes and sort both the lists. Find the length of the each list and store the values. Initialize two indexes with 0 for each of the list. Crate an empty arrays for common and uncommon students. Using condition statements (if,else,elseif,while) iterate both the list with one of the static and second list go through all elements. When ever a common element is found, append that particular name based on the list index in to the output array we created. Follow vice versa for uncommon elements.

```
# sample input
webStudents=["Revanth","Swathi","Sravya","Vineet","Rakul"]
pythonStudents=["Revanth","Vineet","Samanta","Bunny","Swathi","Rakul"]

# sort both the lists
pythonStudents.sort()
webStudents.sort()

# Calculate length of both lists
wLen=len(webStudents) # 5
pLen=len(pythonStudents) # 6

# initialize the output variables
pIndex,wIndex=0,0
commonStudents=[]
uncommonStudent=[]
```

```
while (wIndex < wLen): # 0 < 5 true case
    if webStudents[wIndex] == pythonStudents[pIndex]:
        commonStudents.append(pythonStudents[pIndex])
        wIndex += 1
        pIndex += 1
    elif webStudents[wIndex] > pythonStudents[pIndex]:
        uncommonStudent.append(pythonStudents[pIndex])
        pIndex+=1
    else:
        uncommonStudent.append(webStudents[wIndex])
        wIndex += 1
for i in range(pIndex,pLen):
    uncommonStudent.append(webStudents[i])
```

Output:

```
C:\Users\revan\AppData\Local\Programs\Python\Python36-32\pyth
Students enrolled in Web class
['Rakul', 'Revanth', 'Sravya', 'Swathi', 'Vineet']
Students enrolled in python class
['Bunny', 'Rakul', 'Revanth', 'Samanta', 'Swathi', 'Vineet']
Common Students
['Rakul', 'Revanth', 'Swathi', 'Vineet']
Uncommon Students
['Bunny', 'Samanta', 'Sravya']
```

Task 5: Class design, multiple class creation, oops features etc

Approach/Method

I have created 6 classes employee, fulltime employee, manager, recruiter, job applicants, interview schedule. Classemployee is the parent/base class which has parameters like name, dept, salary etc. Using constructor setter methods, I have assigned values for the parameters and using getter methods displaying those values. Created child classes fulltime employee which extends parent class classemployee features and additionally employee city. In the same way another child class class Manager which has additional attribute experience. Scheduled an interview class which inherits from multiple classes called Recruiter and Applicants and displayed the interview schedule for the job applicant. In addition, I calculated the count of non full time employees and full time employees and their average salaries too.

```
# Base/Parent Class - Employee
#class 1
class classEmployee:

    # Create a data members to count the number of Employees and average salary
  print("---Non-Fulltime employees list---")
  empcount=0
  Totalsalary=0

# Create a constructor to initialize name, family, salary, department
  def __init__(self,ename,esal,edept): # Use of self
      self.ename=ename      # Use of self
      self.esal=esal
      self.edept=edept
      classEmployee.empcount+=1
      classEmployee.Totalsalary+=esal
# Member method
  def display(self):

      print("name:",self.ename,"salary:",self.esal,"dept:",self.edept)

class Fulltime(classEmployee):

    fulltimecount=0
    fulltimetotalsalary=0
    def __init__(self,ename,esal,edept,ecity):
        super().__init__(ename,esal,edept) # Use of super call
        self.ecity=ecity
        Fulltime.fulltimecount+=1
        Fulltime.fulltimetotalsalary+=esal
    def display(self):

        classEmployee.display(self)
        print("Employment City:",self.ecity) # Use of self


# class 6 :defines the interviewing schedule by extending Recuiter class and Applicants class
# Multiple Inheritence
class Interview(Recruiter,Applicants):
    def __init__(self,rname,rdept,aname,ajobid,alocation):
        Recruiter.__init__(self,rname,rdept)
        Applicants.__init__(self,aname,ajobid,alocation)
    def display(self):
        Recruiter.display(self)
        Applicants.display(self)
```

Output:

```
---Non-Fulltime employees list---
name: Revanth salary: 1000 dept: IT
name: Swathi salary: 1500 dept: HR
name: Sravya salary: 2000 dept: Development

---Fulltime employees list---
name: Vineeth salary: 3000 dept: Android applications
Employment City: Kansas City
name: Ajay salary: 2000 dept: IOS applications
Employment City: California

---Manager details---
name: Michael Adams salary: 10000 dept: Recruitment RegionalHead
Manager Experience: 10 years

---Job Applicant details---
Applicant name: Maheshbabu
Applicant jobid: 785489
Applicant job location: Pittsburgh

---Interview Schedule---
Recruiter name: Narasimham
Recruiter dept: Senior Recruiter
Applicant name: MaheshBabu
Applicant jobid: 785489
Applicant job location: Pittsburgh

Total number of employees: 6 | Average salary of all employees: 3250

Total number of fulltime employees: 2 | The average salary of fulltime employees: 2500

Total number of Managers: 1 | Total number of Recruiters: 1
```

Task 6: Generate random array of size 15 and find min and max values from those array

Approach/Method

Using numpy library importing it, I have declared the vector size of 15 (using random class) consisting of values ranging between 0 and 20. Using the predefined method "bincount" by passing "maximum repeated" as argument we can display the most frequent item from that vector.

```
1  import numpy as np
2  # take random values ranging between 0 and 20 of specified size 15
3  x=np.random.randint(0,20,15)
4  print(x)
5  maxRepeated = np.bincount(x)
6  print (str("Most frequent item in the list is: ")+(str((np.argmax(maxRepeated)))))
```

Output:

```
C:\Users\revan\AppData\Local\Programs\Python\Python
[17 12 16  2  2  0  4  0  9  1 16 18 10 12  6]
Most frequent item in the list is: 0
```

Parameters: N/A

Evaluation/Discussion:

The program logic has been tested with various kinds of inputs and successfully executed in all cases.

Conclusion:

The above tasks were implemented from knowledge gained via class lectures and In class programming.

References:

https://stackoverflow.com/ (for logics and errors)

https://www.geeksforgeeks.org/ (programming basics and libraries)