VIVIYAN RICHARDS_33

January 13, 2021

Mam i missed this last 6th question to submit in previous lab assignment so i have added this in.

0.1 QUESTION6:

Other String Functions a) Define a function first() that receives a tuple and returns its first

[8]:
```
def first(num):
    for j in num:
        a,b,c=j
        print(a)
def sort_first(num):
    new_list=sorted(num)
    return new_list

def lists(num):
    return sorted(num, key=None, reverse=0)


def middle(num):
    for j in num:
        a,b,c=j
        print(b)

def sort_middle(num):
    return sorted(num,key=lambda mid:mid[1])

num=[(1,2,3), (2,1,4), (10,7,15), (20,4,50), (30, 6, 40)]
```

[9]:
```
first(num)
```

```
1
2
10
20
30
```

[10]:
```
sort_first(num)
```

[10]:
```
[(1, 2, 3), (2, 1, 4), (10, 7, 15), (20, 4, 50), (30, 6, 40)]
```

[11]:
```
middle(num)
```

```
2
1
```

```
7
4
6
```

[12]:
```
sort_middle(num)
```

[12]:
```
[(2, 1, 4), (1, 2, 3), (20, 4, 50), (30, 6, 40), (10, 7, 15)]
```

0.2 QUESTION7:

Develop a function remove_adjacent(). Given a list of numbers, return a list where all adjacent Test Cases:
1.Input:[1,2,2,3] and Output:[1,2,3]
2.Input:[2,2,3,3,3,] and Output:[2,3]
3.Input:[].Output:[].
4.Input:[2,5,5,6,6,7] and Output:[2,5,6,7]
5.Input:[6,7,7,8,9,9] and Output:[6,7,8,9]

[6]:
```
def remove_adjacent(lst):
    a = []
    for item in lst:
        if len(a):
            if a[-1] != item:
                a.append(item)
        else:
            a.append(item)
    return a
```

[7]:
```
remove_adjacent([1,2,2,3])
```

[7]:
```
[1, 2, 3]
```

[8]:
```
remove_adjacent([2,2,3,3,3])
```

[8]:
```
[2, 3]
```

[9]:
```
remove_adjacent([2,5,5,6,6,7])
```

[9]:
```
[2, 5, 6, 7]
```

[10]:
```
remove_adjacent([6,7,7,8,9,9])
```

[10]:
```
[6, 7, 8, 9]
```

[11]:
```
remove_adjacent([])
```

[11]:
```
[]
```

0.3 QUESTION8:

Write a function verbing(). Given a string, if its length is at least 3, add'ing' to its end. U

[12]:
```
def verbing(str):
    length = len(str)
    if length > 2:
        if str[-3:] == 'ing':
            str += 'ly'
        else:
            str += 'ing'
    return str
```

[13]:
```
verbing("hail")
```

[13]: 'hailing'

[14]:
```
verbing("swimming")
```

[14]: 'swimmingly'

[15]:
```
verbing("do")
```

[15]: 'do'

0.4 QUESTION9:

Develop a function not_bad(). Given a string, find the first appearance of the substring 'not'

[24]:
```
def not_bad(s):
    snot = s.find('not')
    sbad = s.find('bad')
    if sbad>snot:
        s=s.replace(s[snot:(sbad+3)], 'good')
    return s
```

[27]:
```
not_bad("This dinner is not really that bad! ")
```

[27]: 'This dinner is good!'

0.5 QUESTION1:

Write a program for Password Management System

File creation: Ask user to enter N user names and their passwords. Store usernames and passwords into a file named “loginfile.txt”. Store each user and password in one line. File Processing: Write a program that opens your “loginfile.txt” file and reads usernames and passwords from it. Store user names in one list and passwords in another lists. Querying: ask user to enter user name and password for verification. If they match the values stored in the lists, print a message “Login Successful”. Otherwise print a message “Login Failed, try again”

[1]:
```
def register():
    username = input("Please input the first 2 letters of your first name and your birth year ")
    password = input("Please input your desired password ")
    file = open("loginfile.txt","a")
    file.write(username)
    file.write(" ")
    file.write(password)
    file.write("\n")
    file.close()
    if login():
        print("You are now logged in... ")
    else:
        print("You aren't logged in!")

def login():
    username = input("Please enter your username: ")
    password = input("Please enter your password: ")
    for line in open("loginfile.txt","r").readlines(): # Read the lines
        login_info = line.split() # Split on the space, and store the results in a list of two strings
        if username == login_info[0] and password == login_info[1]:
            print("Correct credentials!")
            return True
    print("Incorrect credentials.")
    return False
```

[6]:
```
register()
```

```
Please input the first 2 letters of your first name and your birth year
Viviyan Richards W 2000
Please input your desired password keerthana16
Please enter your username: Viviyan
Richards Please enter your password:
Viviyan Richards W 2000 Correct
credentials!
You are now logged in…
```

[7]:
```
login()
```

```
Please enter your username:Viviyan
Richards  Please enter your
password:Viviyan Richards33 Correct
credentials!
```

[7]: True

0.6 QUESTION2:

Write a program for Student Performance Analysis

Create a text file, ‘marks.txt’, with N marks as floating point numbers. Open the file, read marks from it and compute and print the highest mark. If the user runs the program more than once you should not overwrite the previous text file – simply append the marks to the end of the file. Modify the above program so that it also prints Top-3 highest marks (Note: you may need to use list concept) Modify the above program so that it also prints the Lowest-3 marks.

[11]:
```
marks= [99.0,100.0,95.0,96.0,97.0]
with open('marks1.txt', 'a') as file:
    for mark in marks:
        file.write("%.1f\n" % mark)
number_list=[]
with open('marks1.txt', 'r') as fp:
    number_list = [float(item) for item in fp.readlines()]
print(max(number_list))

def Nmaxelements(list1, N):
    final_list = []
    for i in range(0, N):
        max1 = 0
        for j in range(len(list1)):
            if list1[j] > max1:
                max1 = list1[j];
        list1.remove(max1);
        final_list.append(max1)
    print(final_list)
Nmaxelements(number_list,3)

def Nminelements(list1, N):
    final_list =[];
    for i in range(0, N):
        min1 = 9999999;
        for j in range(len(list1)):
            if list1[j]<min1:
                min1 = list1[j];
        list1.remove(min1);
        final_list.append(min1)
    print(final_list)
Nminelements(number_list,3)
```

```
100.0
[100.0, 100.0, 100.0]
[95.0, 95.0, 95.0]
```

[ ]:

Question7. Develop a function **remove_adjacent()**. Given a list of numbers, return a list where all adjacent same elements have been reduced to a single element. You may create a new list or modify the passed in list.

Test Cases:

1. Input: [1, 2, 2, 3] and output: [1, 2, 3]
2. Input: [2, 2, 3, 3, 3] and output: [2, 3]
3. Input: [ ]. Output: [ ].
4. Input: [2,5,5,6,6,7]
   Output: [2,5,6,7]
5. Input: [6,7,7,8,9,9]
   Output: [6,7,8,9]

```
def remove_adjacent(nums):
    result = []
    for num in nums:
        if len(result) == 0 or num != result[-1]:
            result.append(num)
    return result.

nums = [1,2,2,3]
remove_adjacent(nums)

nums = [2,2,3,3]
remove_adjacent(nums)

nums = [2,2,3]
remove_adjacent(nums)

nums = []
remove_adjacent(nums)

nums = [6,7,7,8,9,9]
remove_adjacent(nums)
```

Question8. Write a function **verbing()**. Given a string, if its length is at least 3, add 'ing' to its end. Unless it already ends in 'ing', in which case add 'ly' instead. If the string length is less than 3, leave it unchanged. Return the resulting string. So 'hail' yields: hailing; 'swimming' yields: swimmingly; 'do' yields: do.

Source code:

```
def verbing(s):
    length = len(s)
if length >= 3:
    if s[-3:] == 'ing':
        s += 'ly'
else:
    s += 'ing'
return s
```

Question9. Develop a function **not_bad()**. Given a string, find the first appearance of the substring 'not' and 'bad'. If the 'bad' follows the 'not', replace the whole 'not'...'bad' substring with 'good'.
Return the resulting string. So 'This dinner is not that bad!' yields: This dinner is good!

```
def not_bad(s):
    snot = s.find('not')
    sbad = s.find('bad')
    if sbad > snot:
        s = s.replace(s[snot:(sbad+3)], 'good')
return s
```

Problem Solving Using Python and R Lab

Lab6. Python File Processing

Question1. Write a program for *Password Management System*

- File creation: Ask user to enter N user names and their passwords. Store usernames and passwords into a file named **"loginfile.txt".** Store each user and password in one line.
- File Processing: Write a program that opens your "security.txt" file and reads usernames and passwords from it. Store user names in one list and passwords in another lists.
- Querying: ask user to enter user name and password for verification. If they match the values stored in the lists, print a message "Login Successful". Otherwise print a message "Login Failed, try again".

source code:

```
def regis():
    username = input ("please input the first 2 letters of yourfirst
    name and your birth year")
    password = input("please input your desired password")
        file =open("loginfile.txt", "a")
        file = write (username)
        file.write(" ")
        file.write(password)
        file.writ ("\n")
        file.close()
        if login():
            print("you are logged in...")
    else:
        print ("you aren't logged in!")
    def login():
            username = input("please Enter your Name :")
            password = input("please Enter your password:")
            for line in open("loginfile.txt", "r").readlines():
                login_info = line.split()
    if username == login_info[0] and password == login_info.
    [1]:
            print ("Correct Credentials!")
            return True
    print ("Incorrect Credentials")
```

Question2. Write a program for *Student Performance Analysis*

- Create a text file, 'marks.txt', with **N** marks as floating point numbers. Open the file, read marks from it and compute and print the highest mark.
- If the user runs the program more than once you should not overwrite the previous text file – simply append the marks to the end of the file.
- Modify the above program so that it also prints **Top-3** highest marks (Note: you may need to use list concept)
- Modify the above program so that it also prints the **Lowest-3** marks.

```
Marks = [99.0, 100.0, 95.0, 96.0, 97.0]
with open('marks1.txt', 'a') as file:
    for mark in marks:
        file.write("%.1f\n" % mark)

number list = []
with open('marks1.txt', 'r') as fp:
    number list = [float(item) for item in fp.readlines()]
print(max(number list))
def Nmaxelements(list1, N):
    final list = []
    for i in range(0, N):
        max1 = 0
        for j in range(len(list1)):
            if list1[j] > max1:
                max1 = list1[j];
        list1.remove(max1);
        final.list.append(max1)
print(final_list)
```

```
final list.append(min1)
print(final list)
Nminelements(number list, 3)
```