JLUFE Spring 2021 (Feb-July)

Final Assignment Report

JILIN UNIVERSITY OF FINANCE AND ECONOMICS

Department of College of Managment Science and Information Engineering

Bachelor of information management and information systems

(2021)

Final Assignment: Part 01

13/07/2021

MODULE: Data Mining

Submitted by: Frost(王凯平) 0314021805403 (1854)
QQ: 577572168
Github ID: Frost57757

Instructions:

1. I have added tips and required learning resources for each question, which helps you to solve the exercise.
2. Finish the assignment on your **OWN** (**Any student find copying/sharing from classmates or internet will get '0' points!!!**)
3. Open the assignment from the Github Clasroom link (https://classroom.github.com/a/yFXO50A4) This will create private repository of the assignment in your Github account.
4. In your repository `Clone` -> `Download ZIP` in your computer.
5. Change your: **Major**, **Name**, **Student number**, **Class number**, **QQ number** and **GitHub ID**
6. Once you finish the Assignment convert your .ipynb file into PDF (https://github.com/milaan9/91_Python_Tips/blob/main/000_Convert_Jupyter_Notebook_to_PDF.ipynl (both .pynb and .pdf file will be required!)
7. Create Folder name "Solution" and copy your 3 files:
   A. Your Jupyter Notebook file (**.ipynb**).
   B. Your PDF converted file (**.pdf**).
   C. **.zip** file containing both .ipynb and .pdf files and name your .zip file as your student number and name. For example: **0318021907632 Milan(米兰).zip**
8. Finally, in your repository `Add files` -> `upload files` upload the "**Solution**" folder and `Commit changes`.

Python Assignment 01

Question 1:

Write a python program that generates a list containing only common elements between the two lists (without duplicates). Make sure your program works on two lists of different sizes.

Expected Output:

```
List 1: [0, 2, 4, 6, 12, 13, 14, 18, 20, 24, 25, 26, 27]
List 2: [0, 4, 7, 9, 10, 11, 13, 14, 17, 18, 20, 33, 39]
List of common elements are:  [0, 4, 13, 14, 18, 20]
```

For extra points:

1. Generate the two list randomly to test this
2. Generate each list in one line of Python.

In [8]:

```
# Solution 1:
import numpy as np
list1 = np.random.randint(1,30,13)
print("List 1:"+str(list1))
list2 = np.random.randint(1,30,13)
print("List 2:"+str(list2))
# 找出两个列表的相同元素
print("List of common elements are: ",end=" ")
for i in list1:
    for j in list2:
        if i == j:
            print(i, end=' ')
```

```
List 1:[16 20  8 25 18  1  7  4 14 28 10  6  4]
List 2:[ 3 19 20  3  5 18  4 19  7 24 28  7  1]
List of common elements are:  20 18 1 7 7 4 28 4
```

Question 2:

Write a python program to find the gravitational force acting between two objects.

$$F = G\frac{m_1 m_2}{r^2}$$

Expected Output:

```
Enter the first mass (m1): 5000000
Enter the second mass (m2): 900000
Enter the distance between the centres of the masses (N): 30
Hence, the Gravitational Force is:  0.33 N
```

In [9]:

```
# Solution 2:
# 万有引力的G=6.67×10-11N•m²/kg²
G=G=6.67*10**-11
m1 = 5000000
m2 = 900000
r = 30
F = m1*m2/r*2
F
```

Out[9]:

```
300000000000.0
```

Question 3:

Write a python program that generates a new list that contains only even elements from the randomly generated list.

Expected Output:

```
Randomly generated list: [64, 63, 90, 13, 38, 27, 19, 51, 97, 32, 18, 75]
List of even elements: [64, 90, 38, 32, 18]
```

In [22]:

```
# Solution 3:
list = np.random.randint(1,100,12)
print("Randomly generated list:",list)
list_2 = []
# 寻找元素里面的偶数
for i in list:
    if(i%2==0):
        list_2.append(i)
list_2
```

```
Randomly generated list: [68 56 26 12 52 75 28 86 15 94 17 69]
```

Out[22]:

```
[68, 56, 26, 12, 52, 28, 86, 94]
```

Question 4:

Write a python program to check if a substring is present in a given string.

Expected Output:

```
Enter string: Hello world
Enter word: world
Substring in string!
```

In [23]:

```
# Solution 4:

str = "Hello world"
if(str.index("world")):
    print("Substring in string!")
else:
    print("Substring not in string!")
```

```
Substring in string!
```

Question 5:

Write a python program that asks the user last 2 digit of (your) student number and generates Fibonacci series.

Expected Output:

```
How many numbers that generates?: 12
Fibonacci series:
 [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144]
```

In [ ]:

```
# Solution 5:
print('What are the last two digits of your student number?')

num = int(input("How many numbers that generates?:"))


n1 = 1
n2 = 1
count = 2
lists = []

if num <= 0:
    print("How many numbers that generates?:")
elif num == 1:
    print("Fibonacci series:")
    print(n1)
else:
    print("Fibonacci series:")
    lists.append(n1)
    lists.append(n2)
    while count < num:
        nth = n1 + n2
        lists.append(nth)

        n1 = n2
        n2 = nth
        count += 1
    print(lists)
```

Question 6:

Write a python program using function that generates a new list that contains all the elements of the first list and removing all the duplicates.

Expected Output:

```
List:  [1, 2, 3, 4, 3, 2, 1]
Result List using loop:  [1, 2, 3, 4]
Result List using sets:  [1, 2, 3, 4]
```

For extra points:

1. Generate the result using two different functions using:
   - one using a loop and constructing a list
   - sets

In [ ]:

```python
# Solution 6:
List=[1, 2, 3, 4, 3, 2, 1]
loop=[]
for i in List:
    if i not in loop:
        loop.append(i)
print("Result List using loop: %s"%loop)

def func1(one_list):
    return list(set(one_list))
print('Result List using sets:',func1(List))
```

Question 7:

Write a python program using functions that asks the user for a string containing multiple words and print back to the user the same string, except with the words in reverse order.

Expected Output:

```
Please enter a sentence: My name is Milaan
The reverse sentence is: Milaan is name My
```

In [ ]:

```python
# Solution 7:
str1 = str(input("Please enter a sentence:"))
for i in str1:
    if not i.isalpha():
        str1  = str1.replace(i,' ')

str1 = str1.split()
b = str1[::-1]
print("The reverse sentence is:"+' '.join(b))
```

Question 8:

Write a python program using function that encrypts a given input with these steps:

Input: "apple"

- Step 1: Reverse the input: "elppa"
- Step 2: Replace all vowels using the following chart:

```
a => 0
e => 1
i => 2
o => 2
u => 3
# 1lpp0
```

- Step 3: Add "aca" to the end of the word: "1lpp0aca"

Expected Output:

```
Word:  apple
Encrypted word: 1lpp0aca
```

More Examples:

```
encrypt("banana") ➞ "0n0n0baca"
encrypt("karaca") ➞ "0c0r0kaca"
encrypt("burak") ➞ "k0r3baca"
encrypt("alpaca") ➞ "0c0pl0aca"
```

In [ ]:

```
# Solution 8:
a=input("请输入原文")
d=a[::-1]
d=d.replace('a','0')
d=d.replace('e','1')
d=d.replace('i','2')
d=d.replace('o','2')
d=d.replace('u','3')
d+="aca"
print(d)
```

Question 9:

Write a python program using function that takes a number num and returns its length.

Expected Output:

```
Enter number: 963969
Total digits in given number:  6
```

In [ ]:

```
# Solution 9:
str1 = input('Enter number:')
len_str1 = len(str1)
print('Total digits in given number:%s'%len_str1)
```

Question 10:

Write a python program using function that takes a string and returns the number (count) of vowels contained within it.

Expected Output:

```
Enter string: Celebration
Total vowels in the string: 5
Identified vowels are:  ['e', 'e', 'a', 'i', 'o']
```

More examples:

```
count_vowels("Palm") ➞ 1
count_vowels("Prediction") ➞ 4
```

In [ ]:

```
# Solution 10:
str = input('Enter string:')
list=[]

count = 0


for i in str:
    if( i=='A' or i=='a' or i=='E' or i=='e' or i=='I' or i=='i' or i=='O' or i=='o' or i=='U' or
        count += 1
        list.append(i)


print("Total vowels in the string: ", count)
print('Identified vowels are:',list)
```

Question 11:

Write a python program to draw pattern as below:

Expected Output:

```
 --- --- ---
|   |   |   |
 --- --- ---
|   |   |   |
 --- --- ---
|   |   |   |
 --- --- ---
```

For extra point:

1. Generate solution by asking the user what size game board they want to draw, and draw it for them to the screen using Python's print statement.

Expected Output:

```
Enter the size of board you want to draw: 4
 --- --- --- ---
|   |   |   |   |
 --- --- --- ---
|   |   |   |   |
 --- --- --- ---
|   |   |   |   |
 --- --- --- ---
|   |   |   |   |
 --- --- --- ---
```

In [ ]:

```
# Solution 11:
n=int(input("Enter the size of board you want to draw:"))
format_=""
for i in range(n+1):
    format_+="|   "
for i in range(n):
    print(" --- --- --- ---")
    print(format_)
print(" --- --- --- ---")
```

Question 12:

Write a python program to ask user for a string and then perform following operations:

1. Calculate the num of digits
2. Calculate the num of characters
3. Calculate the num of vowels
4. Calculate the num of lowercase letters
5. replace ' ' with '_' in the string
6. Print and Store the ouput to 'output.txt' file.

Expected Output:

```
Enter string: Hello World 123
Output printed in'output.txt'
```

Expected Output in output.txt:

```
The entered string is: Hello World 123
The number of digits is: 3
The number of characters is: 15
The number of vowels is: 3
The number of lowercase letters is: 8
The modified string is: Hello_World_123
```

In [ ]:

```
# Solution 12:
digits = 0          # 数字计数器计数器  默认为0个
characters = 0        # 字符计数器  默认为0(字符包含空格)
vowels = 0           # 元音字符计算
lowercase_letters = 0       # 小写字母计数器  默认为0
new_string = ""
string = str(input("The entered string is:"))
# new_string = string.replace('','_')
for i in string:
    if i == ' ':
        i = '_'
    new_string += i

with open("output.txt", 'w') as f:
    f.write(string)  #文件的写操作
for i in string:
    if i.isdigit():             # 如果遇到数字
        digits  += 1
    elif ((i == "a") or (i == "e") or (i == "i") or (i == "o") or (i == "u") or (i == "A") or (i
        vowels += 1
    elif ord(i) in range(97, 123):          # 如果遇到小写字母
        lowercase_letters += 1
    characters += 1

print("The entered string is: %s" %(string))
print("The number of digits is: %d" %(digits))
print("The number of characters is: %d" %(characters))
print("The number of vowels is: %d" %(vowels))
print("The number of lowercase letters is: %d" %(lowercase_letters))
print("The modified string is: %s" %(new_string))
```

Question 13:

Write a python program using function that takes as input three variables from user, and returns the largest of the three. Do this without using the Python `max()` function!

Expected Output:

```
Please enter three integers separated by comma: 12, 66, 31
The maximum value is: 66
```

In [ ]:

```python
# Solution 13:
def zuida(num):

    temp = num.split(',')
    a = temp[0]
    b = temp[1]
    c = temp[2]
    max = a

    if a < b:
        if b > c:
            max = b
        else:
            max = c
    print(max)

n = input('Please enter three integers separated by comma:')
zuida(n)
```

Question 14:

Write a python program where user, will have a number in head between 0 and 100. The program will guess a number, and you, the user, will say whether it is too "high", too "low", or your number. Also, in the end program should print out how many guesses it took to get your number.

Expected Output:

```
Guess a number between 0 and 100 and tell whether high or low when prompted!
My guess is 50. Is that high, low or same? low
My guess is 75. Is that high, low or same? low
My guess is 88. Is that high, low or same? low
My guess is 94. Is that high, low or same? low
My guess is 97. Is that high, low or same? low
My guess is 99. Is that high, low or same? same
Congrats to me! I guessed it in 6 tries.
```

In [ ]:

```
# Solution 14:
import random
csz = random.randint(0, 100)
print("Guess a number between 0 and 100 and tell whether high or low when prompted!")
count=0
while (1):
    count+=1
    sz = input("Enter a number: ")
    if sz.isdigit():
        sz = int(sz)

        if sz == csz:
            print("Congrats to me! I guessed it in %s tries."%count)
            break
        elif sz < csz:
            print("My guess is {0}. Is that high, low or same? low".format(sz))
        else:
            print("My guess is {0}. Is that high, low or same? high".format(sz))
    else:
        print("What you entered is not an integer, please re-enter.")
```

Question 15:

Write a python program using function that takes an list(ordered) of numbers (from smallest to largest) and another number. The function decides whether or not the given number is inside the list and returns (then prints) an appropriate boolean.

> Hint: Use binary search.

Expected Output:

```
List:  [2, 4, 6, 8, 10]
Find  '5': False
Find '10': True
Find '-1': False
Find  '2': True
```

For extra point:

1. Generate list randomly and select he number randomly to be search from the list.

In [ ]:

```python
# Solution 15:
def aaa(num):

    num=num.split(',')
    list=[]
    for i in num:
        list.append(i)
    print(list)
    a=input('Find:')
    if a in num:
        print('True')
    else:
        print('False')
n=input('List:')
aaa(n)
```

Question 16:

Write a python program to generate password. Be creative with how you generate passwords - strong passwords have a mix of lowercase letters, uppercase letters, numbers, and symbols. The passwords should be random, generating a new password every time the user asks for a new password. Include your code in a main method.

Expected Output:

```
Please choose strong or weak:
strong
password: 6|Av.0T^9
do you want a new password? y/n
n
```

For extra points:

1. Ask the user if they want password to be strong(9 characters) or weak(6 characters)?

In [ ]:

```
# Solution 16:
from random import choice
import string

all_chs = string.ascii_letters + string.digits

def gen_pass(n=8):  #默认是循环8次，也就是生成8位密码
    result = ''
    for i in range(n):
        ch = choice(all_chs)
        result +=ch
    return result

if __name__ == '__main__':
    mima=input('Please choose strong or weak:')
    if mima == 'strong':
        print(gen_pass())
        huida=input('do you want a new password? y/n:')
        if huida=='y':
            print('Password generated successfully')
        else:
            print('Password generation failed')
    else:
        print('Password is 0000')
```

Question 17:

Write a python program using function that picks a random word from a list of words from the [dictionary (https://github.com/milaan9/92_Python_Assignments/blob/main/sowpods.txt)](https://github.com/milaan9/92_Python_Assignments/blob/main/sowpods.txt). Each line in the file contains a single word.

> Hint: use the Python random library for picking a random word.

Expected Output:

```
Random word: POTENTIATING
```

In [ ]:

```
# Solution 17:
import random


def cidain():
    danci = open('danci.txt', 'r')
    ls = danci.readlines()
    n = random.randint(0, len(ls))
    print(ls[n])


cidain()
```

Question 18:

Write a python program where a text(.txt) file is given **nameslist.txt (https://github.com/milaan9/92_Python_Assignments/blob/main/nameslist.txt)** that contains list of a bunch of names, count how many of each name there are in the file, and print out the results to the screen.

Expected Output:

```
{'Darth': 31, 'Luke': 15, 'Leia': 54}
```

For extra point:

1. Instead of using the **nameslist.txt (https://github.com/milaan9/92_Python_Assignments/blob/main/nameslist.txt)** file from above (or instead of, if you want the challenge), take this **SUN_Database.txt (https://github.com/milaan9/92_Python_Assignments/blob/main/SUN_Database.txt)** file, and count how many of each “category” of each image there are. This text file is actually a list of files corresponding to the SUN database scene recognition database, and lists the file directory hierarchy for the images. Once you take a look at the first line or two of the file, it will be clear which part represents the scene category.

Expected Output:

```
abbey: 50
airplane_cabin: 50
airport_terminal: 50
alley: 50
amphitheater: 50
...
...
...
wrestling_ring: 50
yard: 50
youth_hostel: 50
```

In [ ]:

```
# Solution 18:

filename = "nameslist.txt"
names_in_file = {}

with open(filename, 'r') as f:
    line = f.readline().strip()
    while line:
        if line in names_in_file.keys():
            names_in_file[line] += 1
        else:
            names_in_file[line] = 1
        line = f.readline().strip()

print(names_in_file)



filename = "Training_01.txt"
categories = {}

with open(filename, 'r') as f:
    line = f.readline().strip().split("/sun_")
    category = line[0][3:]
    while len(category)>0:
        if category in categories.keys():
            categories[category] += 1
        else:
            categories[category] = 1
        line = f.readline().strip().split("/sun_")
        category = line[0][3:]

print(categories)
```

Question 19:

Write a python program where two .txt files are given that have lists of numbers in them, find the numbers that are overlapping. One '**primenumbers1_1000.txt (https://github.com/milaan9/92_Python_Assignments/blob/main/primenumbers1_1000.txt)**' file has a list of all prime numbers under 1000, and the other '**happynumbers1_1000.txt (https://github.com/milaan9/92_Python_Assignments/blob/main/happynumbers1_1000.txt)**' file has a list of **happy numbers (https://en.wikipedia.org/wiki/Happy_number)** up to 1000.

Expected Output:

```
The list of overlapping numbers:
 [7, 13, 19, 23, 31, 79, 97, 103, 109, 139, 167, 193, 239, 263, 293, 313, 331, 367, 379, 3
83, 397, 409, 487, 563, 617, 653, 673, 683, 709, 739, 761, 863, 881, 907, 937]
```

For extra point:

1. Generate solution with functions using list comprehensions

In [ ]:

```
# Solution 19:


def read_numbers ( f ):
    print("Reading file", f, "...", end=" ")
    numbers = []
    with open(f, 'r') as f:
        for line in f:
            numbers += [int(line.strip())]
    print("done.")
    return numbers

def overlapping_elements ( list1, list2 ):
    print("Finding overlapping elements in these two lists ...", end=" ")
    result = []
    for i in list1:
        if i in list2:
            result += [i]
    print("done.")
    return result

if __name__=="__main__":
    prime_numbers = read_numbers("primenumbers.txt")
    happy_numbers = read_numbers("happynumbers.txt")
    overlapping = overlapping_elements(prime_numbers, happy_numbers)
    print(overlapping)
    print("'One line' solution:")
    print([n for n in prime_numbers if n in happy_numbers])
```

Question 20:

Create a function that takes a string as an argument and returns the Morse code equivalent.

For example:

```
encode_morse("HELP ME !") ➞ ".... . .-.. .--.   -- .   -.-.--"
```

Expected Output:

```
Enter a sentence: I love
..   .-.. --- ...- .
Enter morse code: .--. -.-- - .... --- -.
PYTHON
```

This dictionary can be used for coding:

```
    char_to_dots = {
        'A': '.-',    'B': '-...', 'C': '-.-.', 'D': '-..',  'E': '.',    'F': '..-.',
        'G': '--.',   'H': '....', 'I': '..',   'J': '.---', 'K': '-.-',  'L': '.-..',
        'M': '--',    'N': '-.',   'O': '---',  'P': '.--.', 'Q': '--.-', 'R': '.-.',
        'S': '...',   'T': '-',    'U': '..-',  'V': '...-', 'W': '.--',  'X': '-..-',
        'Y': '-.--', 'Z': '--..',

        '0': '-----', '1': '.----', '2': '..---', '3': '...--', '4': '....-',
        '5': '.....', '6': '-....', '7': '--...', '8': '---..', '9': '----.',

        ' ': ' ',        '&': '.-...',  "'": '.----.', '@': '.--.-.', ')': '-.--.-',
        '(': '-.--.',   ':': '---...', ',': '--..--', '=': '-...-',   '!': '-.-.--',
        '.': '.-.-.-', '-': '-....-', '+': '.-.-.', '"': '.-..-.',   '?': '..--..',
        '/': '-..-.'
    }
```

In [1]:

```
# Solution 20:
char_to_dots = {
  'A': '.-', 'B': '-...', 'C': '-.-.', 'D': '-..', 'E': '.', 'F': '..-.',
  'G': '--.', 'H': '....', 'I': '..', 'J': '.---', 'K': '-.-', 'L': '.-..',
  'M': '--', 'N': '-.', 'O': '---', 'P': '.--.', 'Q': '--.-', 'R': '.-.',
  'S': '...', 'T': '-', 'U': '..-', 'V': '...-', 'W': '.--', 'X': '-..-',
  'Y': '-.--', 'Z': '--..', ' ': ' ', '0': '-----',
  '1': '.----', '2': '..---', '3': '...--', '4': '....-', '5': '.....',
  '6': '-....', '7': '--...', '8': '---..', '9': '----.',
  '&': '.-...', "'": '.----.', '@': '.--.-.', ')': '-.--.-', '(': '-.--.',
  ':': '---...', ',': '--..--', '=': '-...-', '!': '-.-.--', '.': '.-.-.-',
  '-': '-....-', '+': '.-.-.', '"': '.-..-.', '?': '..--..', '/': '-..-.'
}

def encode_morse(message):
  nl = []
  global char_to_dots
  char_to_dots = {k.lower(): v for k, v in char_to_dots.items()}

  for w in message.lower():
    for c in w:
      if c in char_to_dots:
        nl.append(char_to_dots[c])
  return ' '.join(nl)
```

executed in 20ms, finished 12:11:36 2021-07-13