JLUFE Spring 2021 (Feb-July)

Final Assignment Report

JILIN UNIVERSITY OF FINANCE AND ECONOMICS

Department of College of Managment Science and Information Engineering

BSc in Data Science and Big Data Technology

(2020)

Final Assignment: Part 01

21/06/2021

MODULE: Data Mining

Submitted by: Milan(米兰) 0318021907632 (1907)

QQ: 3086215265

RULES:

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. Once you finish the Assignment **convert your .ipynb file into PDF** (both .ipynb and .pdf file will be required!)
4. Create **.zip** file and include your two files:
    A. Your Jupyter Notebook file (**001_Python_Assignment_01.ipynb**)
    B. Your PDF converted file of 001_Python_Assignment_01.ipynb (**001_Python_Assignment_01.pdf**)
5. Name your .zip file as your student number and name. example: **0318021907632 Milan(米兰).zip**

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 [6]:
```
# Solution 1:
import random

a = [random.randint(1,20) for i in range(10)]
print(a)
b = [random.randint(1,20) for i in range(15)]
print(b)
print(set(a) & set(b))#取交集
```

```
[15, 7, 1, 5, 14, 1, 7, 11, 10, 10]
[9, 18, 7, 7, 10, 15, 11, 19, 10, 6, 19, 14, 1, 5, 7]
{1, 5, 7, 10, 11, 14, 15}
```

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 [7]:
```
# Solution 2:
import math
from scipy.constants import G

print("Please enter your first mass:")
m1 = float(input())
print("Please enter your second mass:")
m2 = float(input())
print("Please enter the distance between the centres of the masses")
r = float(input())
F = G * m1 * m2 / (r * r)
print("the Gravitational Force is:" + str(F))#数据类型之间不匹配，需进行转换
```

```
Please enter your first mass:
5000000
Please enter your second mass:
900000
Please enter the distance between the centres of the masses
30
the Gravitational Force is:0.33370400000000006
```

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 [8]:
```
# Solution 3:
import random

A = random.sample(range(1,200),20)
print(A)

B = []
C = []

for i in A:
    if i % 2 == 0:
        B.append(i)
    else:
        C.append(i)
print("List of even number: ",B)
print("List of odd number: ",C)#奇偶数可以拉出来
```

```
[48, 199, 177, 70, 57, 175, 24, 195, 110, 68, 166, 39, 98, 154, 184, 163, 104, 33, 11, 152]
List of even number:  [48, 70, 24, 110, 68, 166, 98, 154, 184, 104, 152]
List of odd number:  [199, 177, 57, 175, 195, 39, 163, 33, 11]
```

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 [9]:
```
# Solution 4:
print("Please enter your first string:")
a = input()
print("Please enter your substring:")
b = input()
print(b in a)
```

```
Please enter your first string:
hello word
Please enter your substring:
word
True
```

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 [10]:
```
# Solution 5:
print("Please enter your generates:")
n = int(input())

list=[]
for m in range(0,n):
    if m == 0 or m == 1:
        list.append(1)
    else:
        list.append(list[m - 1]+list[m - 2])
for i in range(len(list)):
    print(list[i],end=' ')#输出一串的list值
```

```
Please enter your generates:
11
1 1 2 3 5 8 13 21 34 55 89
```

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 [11]:
```
# Solution 6:
import random

A = [1,2,3,4,3,2,1]
print(A)
B = []
for i in A:
    if not i in B:
        B.append(i)
print(B)#循环

print(set(A))#集合
```

```
[1, 2, 3, 4, 3, 2, 1]
[1, 2, 3, 4]
{1, 2, 3, 4}
```

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 [13]:
```
# Solution 7:
import re

n = 'My name is ZZC'
a = n.split()
a.reverse()
print(a)
list_add = []
for i in a:
    m = re.match('(\w+)(\W)' , i)#不进行非单词匹配会报错，no such group
    n = re.match('\w+' , i)
    if m :
        m1 = m.group(1)
        m2 = m.group(2)
        list_add.append(m2)
        list_add.append(m1)
    else:
        list_add.append(n.group())
newstr = ''
for j in list_add:
    newstr = newstr + j  +' '
print(newstr)
```

```
['ZZC', 'is', 'name', 'My']
ZZC is name My
```

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 [3]:
```
# Solution 8:
print("Please enter your word:")
A = input()

A1 = A[::-1]

A2 = ""#替换过程中原字符串A1不变，需新定义一个字符串存储
for i in A1:
    if (i == "a"):
        A2 += "0"
    elif (i == "e"):
        A2 +=  "1"
    elif (i == "i"):
        A2 +=  "2"
    elif (i == "o"):
        A2 +=  "2"
    elif (i == "u"):
        A2 +=  "4"
    else:
        A2 += i

A2 += "aca"

print(A2)
```

```
Please enter your word:
apple
1lpp0aca
```

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 [14]:
```
# Solution 9:
print("Please enter a string:")
a = input()
print("The string length is :",len(a))
```

```
Please enter a string:
hello word
The string length is : 10
```

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 [15]:
```
# Solution 10:
A = "The universe is big.It's vast and complicated and ridiculous."
print("a is:",A.count("a"))
print("e is:",A.count("e"))
print("i is:",A.count("i"))
print("o is:",A.count("o"))
print("u is:",A.count("u"))
```

```
a is: 4
e is: 4
i is: 6
o is: 2
u is: 3
```

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 [1]:
```
# Solution 11:
n = int(input("请输入您想要的大小："))
for i in range(2 * n + 1):
    if i % 2 == 0:
        print(" - " * n)
    else:
        print(" | " * (n + 1))
```

```
请输入您想要的大小：4
 -  -  -  - 
 |  |  |  |  | 
 -  -  -  - 
 |  |  |  |  | 
 -  -  -  - 
 |  |  |  |  | 
 -  -  -  - 
 |  |  |  |  | 
 -  -  -  - 
```

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 letters
3. Calculate the num of vowels
4. Calculate the num of lowercase characters
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 characters is: 8
The modified string is: Hello_World_123
```

In [2]:
```
# Solution 12:
print("Please enter your words:")
n = input()
snum = wnum = xwnum = ywnum = 0
other = none = 0
wnum = len(n)
for i in n:
    if 57 >= ord(i) >= 48:
        snum += 1
    elif 122 >= ord(i) >= 97:
        xwnum += 1
    else:
       other += 1
for i in n:
    if i == 'a':
        ywnum += 1
    elif i == 'e':
        ywnum += 1
    elif i == 'i':
        ywnum += 1
    elif i == 'o':
        ywnum += 1
    elif i == 'u':
        ywnum += 1
    else:
        none += 0
print("The number of digits is:",snum)
print("The number of characters is:",wnum)
print("The number of vowels is",ywnum)
print("The number of lowercase characters is",xwnum)

n1 = n.replace(" ","_")
n1
```

```
Please enter your words:
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 characters is 8
```

Out[2]: 'Hello_World_123'

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 [4]:
```
# Solution 13:
print("Please enter your numbers:")
a,b,c = map(int,input().split(","))

max = a
if max > b:
    max = max
else:
    max = b

if max > c:
    max = max
else:
    max = c
print(max)
```

```
Please enter your numbers:
21,8,98
98
```

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 [5]:
```
# Solution 14:
count = 0
min = 0
max = 100
pd = ' '
while pd != 'same':
    guess = (min + max) / 2
    print("My gess is %d " %guess ,"Is that high, low or same?")
    pd = input()
    if(pd == 'high'):
        max = guess
    else:
        min = guess
    if(pd == 'low'):
        min = guess
    else:
        max = guess
    count += 1
print("Congrats to me! I guessed it in %d tries." %count)
```

```
My gess is 50  Is that high, low or same?
low
My gess is 75  Is that high, low or same?
high
My gess is 62  Is that high, low or same?
low
My gess is 68  Is that high, low or same?
low
My gess is 71  Is that high, low or same?
same
Congrats to me! I guessed it in 5 tries.
```

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 [1]:
```
# Solution 15:
import numpy as np
import random

a = [random.randint(1,20) for i in range(10)]
print(a)
b = []
for i in a:
    if i not in b:
        b.append(i)
print(b)
b.sort()
print(b)

print("Please enter the number you want find:")
n = int(input())

low = 0
high = len(b) - 1
index = 1

while low <= high:
    mid = (low + high) // 2
    if n == b[mid]:
        print("一共查找了%d次,此数字在列表中的下标为:%d" %(index, mid))
        break
    elif n < b[mid]:
        low = mid + 1
    else:
        high = mid - 1
    index += 1
    print("一共找了%d次,找不到这样的值!" % index)
```

```
[17, 5, 1, 1, 13, 6, 16, 10, 15, 1]
[17, 5, 1, 13, 6, 16, 10, 15]
[1, 5, 6, 10, 13, 15, 16, 17]
Please enter the number you want find:
10
一共查找了1次,此数字在列表中的下标为:3
```

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 [2]:
```
# Solution 16:
import random
import re
import string

def getkey():
    if c1 == 'strong':
        a = string.ascii_letters + string.digits + string.punctuation
        pw = random.sample(a,8)
        pws = ''.join(pw)
        print(pws)
    else:
        a = string.ascii_letters + string.digits + string.punctuation
        pw = random.sample(a,6)
        pws = ''.join(pw)
        print(pws)

print("Please choose strong or weak?")

c1 = input()
print(getkey())

print("Do you want a new password? y/n")
c2 = input()
while c2 == 'y':
    print(getkey())
    print("Do you want a new password? y/n")
    c2 = input()#不知道再次生成密码的None为什么会出现？？？
```

```
Please choose strong or weak?
strong
vQ0`S)]q
None
Do you want a new password? y/n
y
z,Y5qZ$V
None
Do you want a new password? y/n
n
```

Question 17:

Write a python program using function that picks a random word from a list of words from the dictionary. 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 [3]:
```
# Solution 17:
import random
import linecache

a = random.randrange(1,1000)
print(a)
theline = linecache.getline(r'sowpods.txt', a)
print(theline)
```

```
753
ABSTENTIONIST
```

Question 18:

Write a python program where a text(.txt) file is given 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 file from above (or instead of, if you want the challenge), take this 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 [4]:
```
# Solution 18:
from collections import Counter
f = open("nameslist.txt")
words = []
readline = f.readlines()
words.extend(readline)
print(words)

#word1 = list(set(words))
#print(word1)
names = dict(Counter(words))
print ({ key : value for key , value in names.items() if value > 1})
```

```
['Darth\n', 'Luke\n', 'Darth\n', 'Leia\n', 'Darth\n', 'Leia\n', 'Leia\n', 'Luke\n', 'Darth\n', 'Leia
\n', 'Darth\n', 'Darth\n', 'Leia\n', 'Leia\n', 'Darth\n', 'Leia\n', 'Darth\n', 'Leia\n', 'Luke\n', 'D
arth\n', 'Leia\n', 'Leia\n', 'Darth\n', 'Leia\n', 'Darth\n', 'Darth\n', 'Leia\n', 'Leia\n', 'Luke\n',
'Luke\n', 'Leia\n', 'Darth\n', 'Darth\n', 'Luke\n', 'Leia\n', 'Darth\n', 'Darth\n', 'Leia\n', 'Leia
\n', 'Leia\n', 'Leia\n', 'Leia\n', 'Luke\n', 'Darth\n', 'Luke\n', 'Leia\n', 'Leia\n', 'Leia\n', 'Leia
\n', 'Luke\n', 'Leia\n', 'Darth\n', 'Leia\n', 'Leia\n', 'Darth\n', 'Leia\n', 'Leia\n', 'Darth\n', 'Da
rth\n', 'Leia\n', 'Darth\n', 'Leia\n', 'Darth\n', 'Luke\n', 'Leia\n', 'Luke\n', 'Darth\n', 'Darth\n',
'Luke\n', 'Darth\n', 'Leia\n', 'Darth\n', 'Leia\n', 'Luke\n', 'Leia\n', 'Leia\n', 'Leia\n', 'Leia\n',
'Leia\n', 'Darth\n', 'Leia\n', 'Leia\n', 'Leia\n', 'Leia\n', 'Leia\n', 'Leia\n', 'Leia\n', 'Luke\n',
'Leia\n', 'Leia\n', 'Leia\n', 'Leia\n', 'Leia\n', 'Leia\n', 'Darth\n', 'Luke\n', 'Darth\n', 'Leia\n',
'Leia\n', 'Darth']
{'Leia\n': 54, 'Darth\n': 30, 'Luke\n': 15}
```

Question 19:

Write a python program where given two .txt files that have lists of numbers in them, find the numbers that are overlapping. One 'primenumbers1_1000.txt' file has a list of all prime numbers under 1000, and the other 'happynumbers1_1000.txt' file has a list of happy numbers 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, 383, 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 [5]: # Solution 19:
file1 = open(r'happynumbers1_1000.txt')
file2 = open(r'primenumbers1_1000.txt')

a = file1.readlines()
b = file2.readlines()
c = [i for i in a if i in b]
#print(a)
#print(b)
print(c)
```

```
['7\n', '13\n', '19\n', '23\n', '31\n', '79\n', '97\n', '103\n', '109\n', '139\n', '167\n', '193\n',
'239\n', '263\n', '293\n', '313\n', '331\n', '367\n', '379\n', '383\n', '397\n', '487\n', '563\n', '6
17\n', '653\n', '673\n', '683\n', '709\n', '739\n', '761\n', '881\n', '907\n', '937\n']
```

Question 20:

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

Expected Output:

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

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 [9]: # Solution 20:
chars = ",.0123456789?abcdefghijklmnopqrstuvwxyz"

codes = """--..-- .-.-.- ----- .---- ..--- ...-- ....- ..... -.... --... ---..

----. ..--.. .- -... -.-. -... . ..-. --. .... .. .--- -.- .-.. --

-. --- .--. --.- .-. ... - ..- ...- .-- -..- -.-- --.."""
keys = dict(zip(chars, codes.split()))

def zhuanhuan(char):
    return keys.get(char.lower(), char)

print(''.join(zhuanhuan(c) for c in 'HELP ME !'))
print(''.join(zhuanhuan(c) for c in 'I FINISH THIS WORK!'))
```

```
......-...--. --. !
.. ..-...-.......... -......... .-----.-.-.-!
```

```
In [ ]:
```