NANDAN A.S.

1BG15CS059

6TH SEMESTER CSE

Report – O.D.U. Internship Selection Problems

# Problem 1:

Think about a Classmate device designed for your University, which can answer anything about your classmates quickly & efficiently. For example, Classmate starts its conversation by saying "Hello, What’s your name?" and a student responds to it by saying "Jon Snow”. Jon Snow (alias Checker) is a student of Class A, and he asks about Samwell Tarly (alias Mate) when prompted for "How can I help you, Jon Snow". Classmate should identify Jon Snow belongs to Class A and search Samwell Tarly with in Class A and respond if found by saying "Samwell Tarly is from Tennessee (home town) and he likes Basketball (likes)". If it cannot find Samwell Tarly, it should prompt "Samwell Tarly not found in Class A. Please ask about students in your class."

Each student has the following information stored at the beginning of the program.

First Name, Last Name, Likes, Home Town, Class

For example,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| First Name | Last Name | Likes | Home Town | Class |
| Jon | Snow | Football | New York | A |
| Samwell | Tarly | Basketball | Tennessee | A |
| Arya | Stark | Hunting | New York | B |
| Daenerys | Targaryen | Flying | Dragon Stone | A |
| .. | .. | .. | .. | .. |

System: Hello, What’s your name?

User: Jon Snow

System: Whose information would you like to know, Jon Snow?

User: Samwell Tarly

System: Samwell Tarly is from Tennessee and he likes Basketball.

User: Arya Stark

System: Arya Stark not found in Class A. Please ask about students in your class.

User: Daenerys Targaryen

…

…

…

User: Exit

System: Bye.

Tasks:

* Design an efficient way to store the information of all the students in the University.
* Matching Strategy to quickly & efficiently identify student (checker)
* Searching strategy to identify Mate with in the Class.

Note: Assume the strength of University is greater than 10,000 to test the efficiency of your algorithm.

Please comment your code and explain your methodology clearly

**Solution:**

Waterfall Model is used in the software development process for this problem.

1. Requirements – A simple program which is capable of getting details of a person’s classmate by the press of a button.
   1. An efficient way to store and read the information of about 10,000 students in the University
   2. Matching Strategy to quickly and efficiently identify the student
2. System Design – This program will be tested in a small scale. A simple computer with dual core processor and 2 GB of RAM with Linux or Windows is required.
3. Implementation –The solution is coded in Python 3.6. PyCharm was the IDE used in development. This project can be solved by storing all the records in a Comma Separated Values (CSV) file. All the contents of the file can be read in the beginning of the program and can be processed accordingly. A simple search algorithm can be used for finding the match.

Python Libraries Used - CSV

Another approach to the same problem when the number of students is as high as 10,000 is to use a data structure which supports hash table implementation for efficient searching techniques.

1. Testing – The test case mentioned in the question was executed successfully and the outputs were as expected.

Python Code – File Computing

The data is stored in the CSV file named “data.csv”

data.csv

First Name,Last Name,Likes,Home Town,Class

John,Snow,Football,New York,A

Samwell,Tarly,Basketball,Tennessee,A

Arya,Stark,Hunting,New York,B

Daenerys,Targaryen,Flying,Dragon Stone,A

“p1.py” is the driver program for the above file.

p1.py

import csv

# This module is used for reading and writing CSV files

user = input('System: Hello, What\'s your name?\nUser: ')

# Get the input from the user

file = open('data.csv')

# Open the file containing the data

#Read CSV file and split on ','

csv\_file = csv.reader(file,delimiter=',')

# A \_csv.reader object is created for handling the data

# Get the all the data stored in the file

data = [row for row in csv\_file]

# Delete the first row as it contains the column headers

del(data[0])

# Get the Class to which the user belongs

cls = None

class\_of\_user = None

# Unpack the data as lists

for row in data:

name = str(row[0]+' '+row[1])

# Combine the first name and last name

if(name==user):

class\_of\_user = row[4]

# Prompt string which will be displayed once

prompt = 'System: Whose information would you like to know ' + user + '?'

print(prompt)

# A loop which halts when 'Exit' is typed

while(True):

# Get the name that has to be searched

search\_element = input('User: ')

# Break the loop if the input is 'Exit'

if(search\_element == 'Exit'):

print('System: Bye.')

exit()

flag = 0

for row in data:

# Form the name string

name = str(row[0] + ' ' + row[1])

# Search for the name and print a success message if found

if(search\_element==name and row[4]==class\_of\_user):

print('System:',str(row[0]+' '+row[1]+' is from '+row[3]+' and he likes '+row[2]+'.'))

flag = 1

break

# In case of an unsuccessful search, this message will be displayed

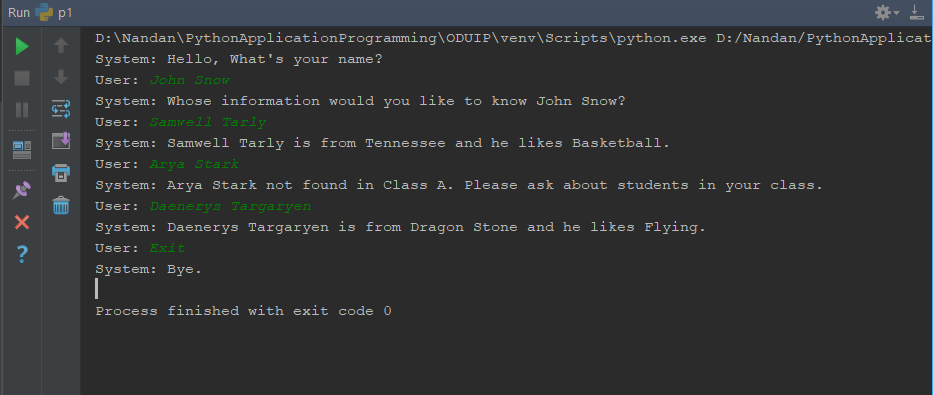
if( flag == 0):

print('System:',search\_element,'not found in Class A. Please ask about students in your class.')

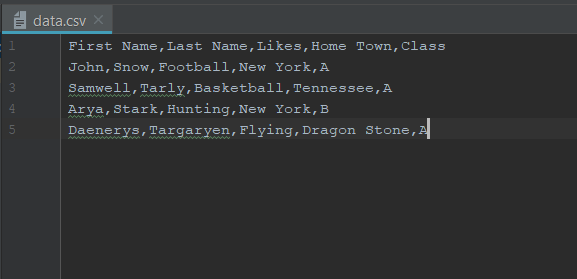
Screenshots of the Output-

File Computing –

Output Window



data.csv



Python Code -

Dictionaries in Python are implemented using hash tables. It is an array whose indexes are obtained using a hash function on the keys

Python Implementation of a Hash Table – Every dictionary in python is a hash table. Using a hash table makes these kind of computations fast.

import csv

# This module is used for reading and writing CSV files

user = input('System: Hello, What\'s your name?\nUser: ')

# Get the input from the user

file = open('data.csv')

# Open the file containing the data

#Read CSV file and split on ','

csv\_file = csv.reader(file,delimiter=',')

# A \_csv.reader object is created for handling the data

# Get the all the data stored in the file

data = [row for row in csv\_file]

del(data[0])

# Create a dictionary with names as key and hometown as values

name\_hometownn = { row[0]+' '+row[1]:row[3] for row in data}

# Create a dictionary with names as key and class as values

name\_class = { row[0]+' '+row[1]:row[4] for row in data}

# Create a dictionary with names as key and likes as values

name\_likes = { row[0]+' '+row[1]:row[2] for row in data}

class\_of\_user = name\_class.get(user)

prompt = 'System: Whose information would you like to know ' + user + '?'

print(prompt)

# A loop which halts when 'Exit' is typed

while(True):

# Get the name that has to be searched

search\_element = input('User: ')

# Break the loop if the input is 'Exit'

if(search\_element == 'Exit'):

print('System: Bye.')

exit()

# Find the details of the classmate's name entered by the student

if(not( name\_class.get(search\_element) is None)):

if(name\_class.get(search\_element)==class\_of\_user):

if(not(name\_likes.get(search\_element) is None and name\_hometown.get(search\_element) is None )):

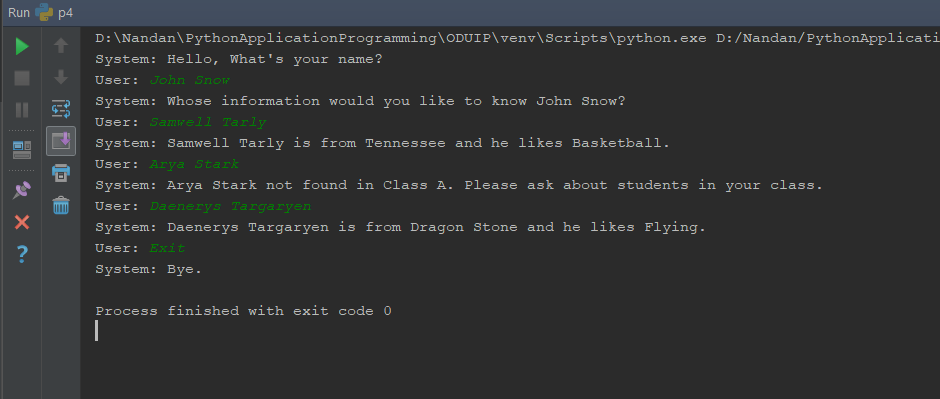
print('System: %s is from %s and he likes %s.'%(search\_element,name\_hometownn.get(search\_element),name\_likes.get(search\_element)))

# Print a message if the required data is not found in the same section

else:

print('System: %s not found in Class %s. Please ask about students in your class.'%(search\_element,class\_of\_user))

Output -



Python Code – A database approach

import pymysql

# Import the required SQL driver to connect to the Database

# Establish a working connection to the database

try:

conn = pymysql.connect(host='localhost',user='root',passwd='',db='ODUIP')

except:

print('Error! Can\'t connect to the Database at the moment ')

exit()

# Get the cursor object

cursor = conn.cursor()

user = input('System: Hello, What\'s your name?\nUser: ')

prompt = 'System: Whose information would you like to know ' + user + '?'

print(prompt)

# Get the first name and the last name

fname , lname = user.split(" ")

# SQL Query to find the class of the user

sqlstring = " SELECT CLASS FROM STUDENTDATA WHERE FIRST\_NAME = '%s' AND LAST\_NAME = '%s' "%(fname,lname)

# Execute the Query

cursor.execute(sqlstring)

# Convert it as a string for ease of usage

cls = ''.join(list(cursor.fetchone()))

# This loop stops when the user enters 'Exit'

while(True):

# Get the name that has to be searched

search\_element = input('User: ')

# Break the loop if the input is 'Exit'

if (search\_element == 'Exit'):

print('System: Bye.')

exit()

try:

# Split the search name into it's first name and last name

sfname , slname = search\_element.split(" ")

sqlstring = """

SELECT \* FROM STUDENTDATA

WHERE FIRST\_NAME = '%s' AND LAST\_NAME = '%s'

AND CLASS = ( SELECT CLASS FROM STUDENTDATA WHERE FIRST\_NAME = '%s' AND LAST\_NAME = '%s' )

""" %(sfname,slname,fname,lname)

cursor.execute(sqlstring)

result = cursor.fetchone()

# Check if the result object is empty and perform a set of actions accordingly

if(result==None):

print('System:',search\_element,'not found in Class',cls,'. Please ask about students in your class.')

else:

result = list(result)

print('System: %s %s is from %s and he likes %s.' %(result[0] , result[1] , result[3] , result[2]))

except:

print('Error! Can\'t connect to database at the moment')

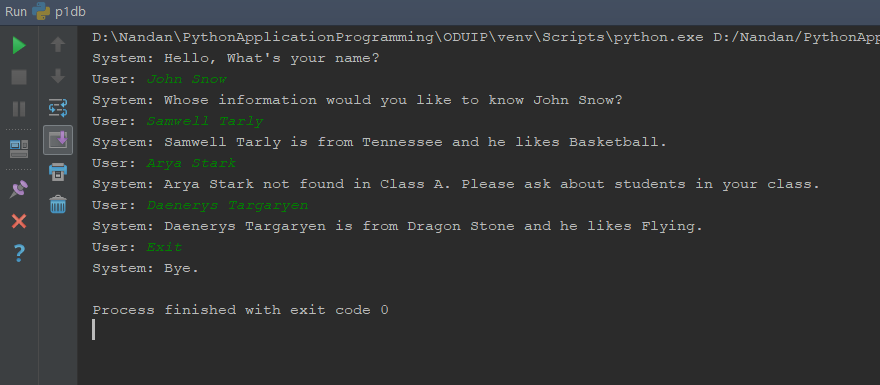
# Close the connection which was made to access the database

conn.close()

Output -   
Snapshot of the Database



Driver Code –



**Problem 2**

Given an N by N matrix, and a string (s1) (length may be less than NXN), fill the matrix with s1 row by row going left to right. Fill the matrix completely by repeating the string. Now, given another string (s2) from the user and display all the occurrences of s2 in the matrix by traversing only:

1. Left to right.
2. Top to bottom.

If there are no such occurrences, output “No matches”.

Here is an example, N = 4, s1 = “MISSISSIPPI” and s2 =”IS”

Matrix looks like this

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | | 1 | | |  |  | 2 | |  |  | 3 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 |  | M |  |  | I |  | |  | S | |  | S |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 |  | I |  |  | S |  |  |  | S | | | I |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  | P |  |  | P | | |  | I |  |  | M |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  | I |  |  | S | |  |  | S |  | | I |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

Output:

(0, 1) to (0, 2) -> left to right

(0, 1) to (1, 1) -> top to bottom

(1, 0) to (1, 1) -> left to right

(2, 2) to (3, 2) -> top to bottom

(3, 0) to (3, 1) -> left to right

**Solution –**

**Python Code -**

The answer to the problem is coded using Python 3.6

# Get the value of N , s1 , s2 from the user

N = int(input('Enter the value of N\n').strip())

s1 = input('Enter the value of s1\n').strip()

s2 = input('Enter the value of s2\n').strip()

# All the characters in the string are converted into a matrix

l1 = []

# The final solution list

solution = []

c = 0

for i in range(N):

temp = []

for j in range(N):

temp.append(s1[c % len(s1)])

c = c + 1

l1.append(temp)

# The nested list l1 contains the matrix

# Check row-wise for every occurrence of string s2

rowno = 0

for row in l1:

rowstring = ''.join(row)

# print(rowstring)

start = 0

if( rowstring.find(s2) != -1):

start = 0

while(start != -1):

temp = rowstring.find(s2,start)

if (temp != -1):

solution.append(str(('(%d , %d) to (%d , %d) -> left to right '%(rowno,temp,rowno,temp+len(s2)-1))))

start = temp + 1

else:

break

rowno += 1

# The transpose of the matrix is calculated

l1 = list(map(list, zip(\*l1)))

# Check column-wise for every occurrence of the string s2

colno = 0

for col in l1:

colstring = ''.join(col)

# print(rowstring)

start = 0

if( colstring.find(s2) != -1):

start = 0

while(start != -1):

temp = colstring.find(s2,start)

if (temp != -1):

solution.append(str(('(%d , %d) to (%d , %d) -> top to bottom '%(temp,colno,temp+N-len(s2)-1,colno))))

start = temp + 1

else:

break

colno += 1

# Sort the solution list, join them using new line escape sequence and print

print('\n'.join(sorted(solution)))

Output

