|  |  |  |
| --- | --- | --- |
|  |  | **ISM 6225**  **Distributed Information systems** |
| Prof Agrawal TA: Khaninder Kommagoni | | |

Assignment 1 – Programming Introduction

Primary objective: Develop familiarity with essential programming constructs

Secondary objective: develop comfort with using the IDE and GitHub

*Estimated time: 25 hours*

## Introduction

Full-stack application development is an essential skill needed to succeed and even survive in business analytics and/ or information systems roles, especially as AI takes over many rudimentary tasks formerly performed by analysts. This assignment introduces the essential programming constructs such as variables, selection, loops, methods and arrays used to build such applications. Specifically, this assignment avoids the use of API methods and object-oriented programming. Those tasks are left for later assignments. This assignment also does not check for efficiency in program implementation. That is something you will develop over a lifetime in the profession. Rather, the focus is on simple programming exercises for students to learn basic industry best practices. One design goal for this assignment was to focus tightly on introductory programming structures, with low probability that students would find ready-to-use solutions online.

This is an individual assignment, to give every student the opportunity to develop the necessary skills to become a productive contributor to project teams in this class and beyond.

## Activity

Use a programming language of your choice to define methods to do the operations specified in the method signatures and hints below. The methods are listed in the recommended sequence of development. A starter Program.cs file is included in the appendix.

## Submission

Push the code to GitHub and submit the URL to Canvas. Also, get the output from a sample run that shows the use of all required methods and upload/push a screenshot to GitHub. This serves as a quick check. Submit your self-reflection as a comment to the assignment.

## Grading scheme

Each method carries 1 point. You will be graded on the following aspects for each question:

Logic (including appropriate organization of logic into methods) : 0.5

Handling all reasonable corner cases : 0.2

Descriptive comments explaining the logic to reviewer : 0.2

Self-reflection (time taken, learning, and recommendations) : 0.1

## Method specifications

**QUESTION 1:**

/\*n – number of rows for the pattern, integer (int)

\* summary : This method prints a triangle pattern using \*.

\* For example n = 5 will display the output as:

\*

\* \*

\* \*\*\*

\* \*\*\*\*\*

\* \*\*\*\*\*\*\*

\* \*\*\*\*\*\*\*\*\*

\*

\* returns : N/A

\* return type : void

\*/

*# Function to demonstrate printing pattern triangle*

def triangle(n):

*# number of spaces*

k = 2\*n - 2

*# outer loop to handle number of rows*

for i in range(0, n):

*# inner loop to handle number spaces*

*# values changing acc. to requirement*

for j in range(0, k):

print(end=" ")

*# decrementing k after each loop*

k = k - 1

*# inner loop to handle number of columns*

*# values changing acc. to outer loop*

for j in range(0, i+1):

*# printing stars*

print("\* ", end="")

*# ending line after each row*

print("\r")

*# Driver Code*

n1=input("provide num")

n=int(n1)

triangle(n)

**QUESTION 2:**

/\* n2 – number of terms of the series, integer (int)

\* This method prints the following series(Odd numbers) till n terms: 1, 3, 5, 7, 9,

\* 11… and their sum.

\* For example, if n2 = 5, output will be

\*

\* The odd numbers are : 1, 3, 5, 7, 9

\* The sum is : 25

\*

\* Returns : N/A

\* Return type: void

\*

\*/

def suma(n):

lis=list*()#creating a list to place the numbers inside*

sum=0

for i in range(1,2\*n,2): *# setting the range for the loop*

lis.append(i)

sum=sum+int(i*)#adding the individual numbers in the loop*

print("The odd numbers are :",lis) *# printing as required in the assignment*

print("The sum of integers is :",sum) *# printing as required in the assignment*

n1=input("Provide the n value")

n=int(n1) *#passing values*

suma(n)

**QUESTION 3:**

/\* An array is *monotonic* if it is either monotone increasing or monotone decreasing.

\* An array A is monotone increasing if for all i <= j, A[i] <= A[j].  An array A is

\* monotone decreasing if for all i <= j, A[i] >= A[j].

\* Return true if and only if the given array A is monotonic

\* For example:

\* Input: A = [1,2,2,6] will return the output: true

\* Input: A = [3,3,2,1] will return the output : true

\* Input: A = [4,5,2,3] will return the output: false

\* Input: A = [1,1,1,1] will return the output : true

\* returns : Boolean Value

\* return type : bool

\*/

def isMonotonic(A):

return (all(A[i] <= A[i + 1] for i in range(len(A) - 1)) or

all(A[i] >= A[i + 1] for i in range(len(A) - 1)))

*# Driver program*

A = [1,2,2,6]

*# Print required result*

print(isMonotonic(A))

**QUESTION 4:**

/\* Given an array of integers and an integer n, you need to find the number of unique

\* n-diff pairs in the array. Here a n-diff pair is defined as an integer pair (i, j),

\* where i and j are both numbers in the array and their absolute difference is n.

\* Example 1:

\* Input: [3, 1, 4, 1, 5], k = 2

\* Output: 2

\* Explanation: There are two 2-diff pairs in the array, (1, 3) and (3, 5).

\* Although we have two 1s in the input, we should only return the number of unique

\* pairs.

\* Example 2:

\* Input:[1, 2, 3, 4, 5], k = 1

\* Output: 4

\* Explanation: There are four 1-diff pairs in the array, (1, 2), (2, 3), (3, 4) and

\* (4, 5).

\* Example 3:

\* Input: [1, 3, 1, 5, 4], k = 0

\* Output: 1

\* Explanation: There is one 0-diff pair in the array, (1, 1).

\* Note : The pairs (i,j) and (j,i) count as same.

def count(arr,k):

count = 0

n=len(arr)

# Pick all elements one by one

for i in range(0, n):

# See if there is a pair of this picked element

for j in range(i+1, n) :

if arr[i] - arr[j] == k or arr[j] - arr[i] == k:

count=count+ 1

print(count)

arr = [1, 3, 1, 5, 4]

k = 0

count(arr,k)

**QUESTION 5:**

/\* Imagine a special bulls keyboard with all keys in a single row.

\* Given a string keyboard of length 26 indicating the layout of the keyboard(indexed

\* from 0 to 25), initially your finger is at index 0. To type a character, you have

\* to move your finger to the index of the desired character.

\* The time taken to move your finger from index i to index j is |i – j|

\* You want to type a single word. Complete the function to calculate

\* how much time it takes to type it with one finger.

\* Example 1:

\* Input: keyboard = “abcdefghijklmnopqrstuvwxyz” word = “dis”

\* Output: 18

\* Explanation: Initial index 0 at a . The moves from 0 to 3 , then 3 to 8 and finally

\* from 8 to 18. Therefore total time = 3 + 5 + 10 = 18

\*

\* Example 2:

\* Input: keyboard = “hijklmnopqrstuvwxyzabcdefg” word = “gobulls”

\* Output: 79

\* returns : Integer

\* return type : int

\*/

def keyboard(a,b): *#takes entry of the keyboard and word*

la=list(a) *#convert to list*

ls=list(b) *# convert to list*

sum=0

si=0

total=0

for i in ls: *#Start taking one character at a time to compare*

if i in la: *#if the given chracter is in the keyboard*

index=la.index(i) *#find the index of it*

sum=abs(si-index) *# subtract the index with the previous index*

total=total+sum *#add the sum of index*

si=index *# re assign the si index as the previous character's index*

return total *# return the total value*

a1="hijklmnopqrstuvwxyzabcdefg"

b1="gobulls"

keyboard(a1,b1)

QUESTION 6:

/\* Given two strings str1 and str2 and below operations that can performed on str1.

\* Find minimum number of edits (operations) required to convert ‘str1’ into ‘str2’

\* 1.Insert

\* 2.Remove

\* 3.Replace

\*

\* All of the above operations are of equal cost.

\*

\* Example 1:

\* Input: str1 = “goulls” str2 = “gobulls”

\* Output: 1

\* Explanation: We can convert str1 to str2 by inserting ‘b’

\*

\* Example 2:

\* Input: str1 = “robky” str2 = “rocky”

\* Output: 1

\* Explanation: We can convert str1 to str2 by replacing ‘b’ with ‘u’

\*

\* Example 3:

\* Input: str1 = “sunday” str2= “saturday”

\* Output: 3

\* Explanation: We can convert by replacing ‘n’ with ‘r’ and inserting ‘t’ and ‘a’

\* returns : Integer

\* return type : int

\*/

def change(str1, str2, c, d):

*# If first string is empty, insert all characters of second string into first*

if c == 0:

return d

*# If second string is empty,remove all characters of first string*

if d == 0:

return c

*# If last characters of two strings are same, nothing*

*# much to do. Ignore last characters and get count for*

*# remaining strings.*

if str1[c-1]== str2[d-1]:

return change(str1, str2, c-1, d-1)

*# If last characters are not same, consider all three*

*# operations on last character of first string, recursively*

*# compute minimum cost for all three operations and take*

*# minimum of three values.*

return 1 + min(change(str1, str2, c, d-1), # Insert

change(str1, str2, c-1, d), # Remove

change(str1, str2, c-1, d-1) # Replace

)

*# Driver program to test the above function*

str1 = "sunday"

str2 = "saturday"

print(change(str1, str2, len(str1), len(str2)))