Q1 of 15outlined\_flag

What is the output of the below code snippet?

4 6

10

3 7

TypeError

**Answer -**

//-------------------------------------------------------------------------------

Q2 of 15outlined\_flag

Find the output of the below Python code.  
Note: Assume that necessary imports have been done.

[33, 45, 67, 79, 100]

[100, 79, 67, 45, 33]

[33, 44, 67, 78, 99]

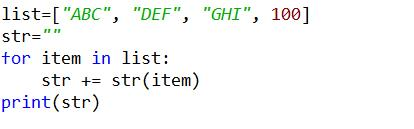
[99, 78, 67, 44, 33]

**Answer -**

//-------------------------------------------------------------------------------

Q3 of 15outlined\_flag

What is the output of the below code snippet?



TypeError

ABCDEFGHI100

ValueError

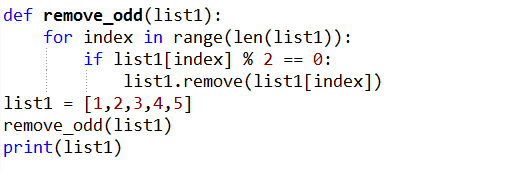
ABCDEFGHI

**Answer -**

//-------------------------------------------------------------------------------

Q4 of 15outlined\_flag

What is the output of the below code snippet?



[1,3,5]

IndexError

[1,2,4,5]

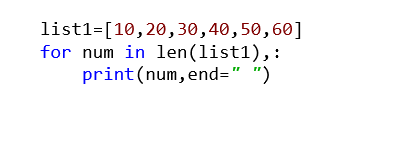
[1,2,3,4,5]

**Answer -**

//-------------------------------------------------------------------------------

Q5 of 15outlined\_flag

What is the output of the below code snippet?



6

10 20 30 40 50 60

60

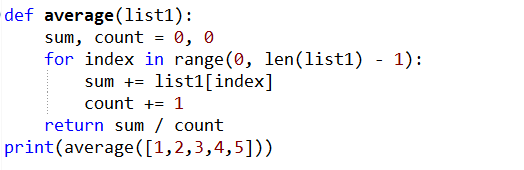
Error

**Answer -**

//-------------------------------------------------------------------------------

Q6 of 15outlined\_flag

What is the output of the below code snippet?



2.5

2

3.0

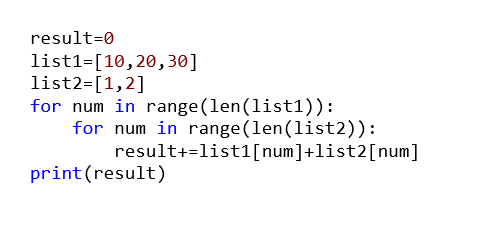
3

**Answer -**

//-------------------------------------------------------------------------------

Q7 of 15outlined\_flag

What is the output of the below code snippet?



22

Error

129

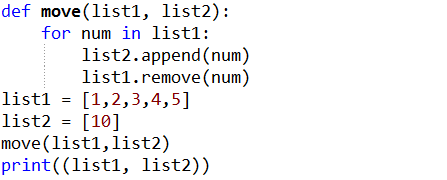
99

**Answer -**

//-------------------------------------------------------------------------------

Q8 of 15outlined\_flag

What is the output of the below code snippet?



 ([],[10,1,2,3,4,5])

([1,2,3,4,5],[10])

([2, 4], [10, 1, 3, 5])

ValueError

**Answer -**

//-------------------------------------------------------------------------------

Q9 of 15outlined\_flag

What would be the output of Python code given below?

elements=[2,5,6,0]  
try:  
    div=elements[4]/elements[3]  
except ZeroDivisionError:  
    print("Infinity")  
except IndexError:  
    print("Index Error")  
except Exception:  
    print("0")  
finally:  
    print("In finally block")

a. Infinity  
    In finally block

b. 0  
    In finally block

c. Index Error  
    In finally block

d. Index Error

a

b

c

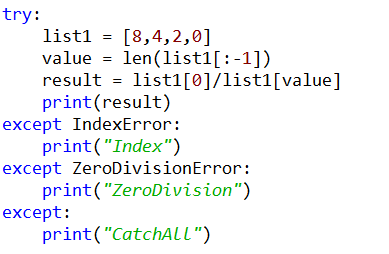
D

**Answer -**

//-------------------------------------------------------------------------------

Q10 of 15outlined\_flag

What is the output of the below code snippet?



ZeroDivision

Index

CatchAll

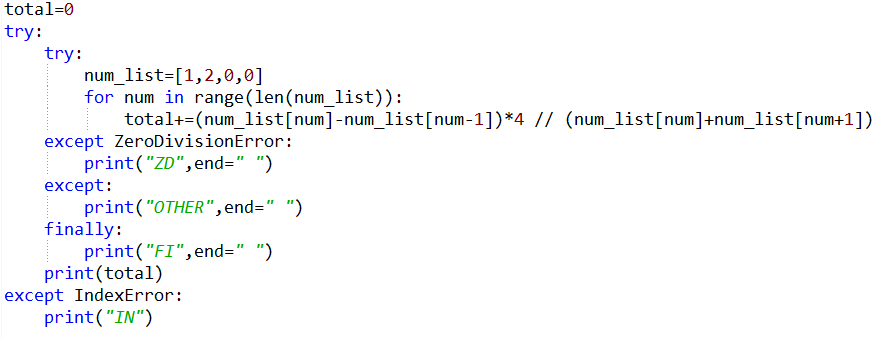
Index ZeroDivision

**Answer -**

//-------------------------------------------------------------------------------

Q11 of 15outlined\_flag

What is the output of the below code snippet?



 ZD FI 3

ZD FI

OTHER FI

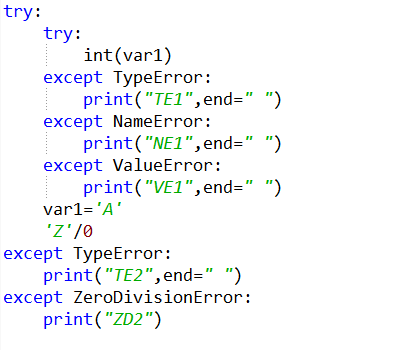
OTHER FI 3

**Answer -**

//-------------------------------------------------------------------------------

Q12 of 15outlined\_flag

What is the output of the below code snippet?



TE1 TE2

NE1 TE2

ZD2

TE2 ZD2

**Answer -**

//-------------------------------------------------------------------------------

Q13 of 15outlined\_flag

What is the output of the below code snippet?

VE

ZD

CA

VE CA

**Answer -**

//-------------------------------------------------------------------------------

Q14 of 15outlined\_flag

What is the output of the below code snippet?

VE

ZD

CA

TE

**Answer -**

//-------------------------------------------------------------------------------

Q15 of 15outlined\_flag

What is the output of the below code snippet?

1

2

3

Error

**Answer -**

//-------------------------------------------------------------------------------

Q2 of 15outlined\_flag Find the output of th... by Ashutosh (Unverified)Ashutosh (Unverified)9:04 AM

Q2 of 15outlined\_flag

Find the output of the below Python code.  
Note: Assume that necessary imports have been done.

[33, 45, 67, 79, 100]

[100, 79, 67, 45, 33]

[33, 44, 67, 78, 99]

[99, 78, 67, 44, 33]

**Answer -**

//-------------------------------------------------------------------------------

//---------------------------------------------------------------------------------------

**Regular Expression -**

It basically used for matches the patterns.

* A regular expression (RE) in a programming language is a special text string used for describing a search pattern. It is extremely useful for extracting information from text such as code, files, log, spreadsheets, or even documents.
* Regular expressions can contain both special and ordinary characters. Most ordinary characters such as 'A', 'a', or '0' are the simplest regular expressions. These characters simply match themselves.

**Regular Expression Syntax**

**RE**

import re

* "re" module included with Python primarily used for string searching and manipulation  Also used frequently for web page "Scraping" (extract large amount of data from websites)

**Using regular expression methods**

The "re" package provides several methods to actually perform queries on an input string. The method we going to see are

* re.match()
* re.search()
* re.findall()

**Precompiled patterns**

**compile()**

* re module provide compile () function to compile a pattern into RegexObject.

**pattern = re.compile("ab")**

**Iterating over matches using `re.ﬁnditer`**

**finditer():**

* You can use re.finditer to iterate over all matches in a string.
* This gives you (in comparison to re.findall extra information, such as information about the match location in the string (indexes):
* Returns an Iterator object which yields Match object for every Match

**def finditer Found at: re**

**def finditer(pattern, string, flags=0):**

* Return an iterator over all non-overlapping matches in the string.  For each match, the iterator returns a Match object.
* Empty matches are included in the result."""

                       return \_compile(pattern, flags).finditer(string)

**matcher = pattern.finditer("abaababa")**

**On Match object we can call the following methods.**

**start()**

Returns start index of the match

**end()**

Returns end+1 index of the match

**group()**

Returns the matched string

count=0

pattern=re.compile(*"python"*)

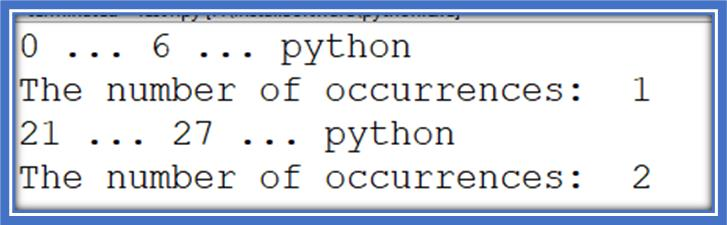
matcher=pattern.finditer(*"python is a prog and python is High level"*)

for match in matcher:

   count+=1

   print(match.start(),*"..."*,match.end(),*"..."*,match.group())

   print(*"The number of occurrences: "*,count)



**Note: We can pass pattern directly as argument to finditer() function.**

import re

count=0

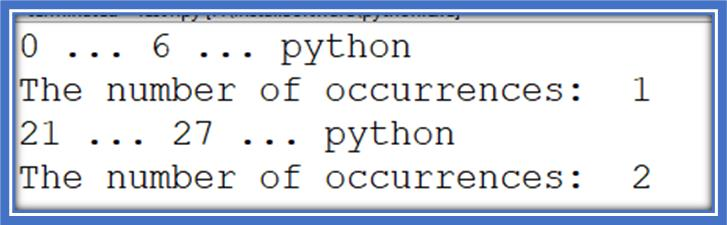
matcher=re.finditer(*"python"*,*"python is a prog and python is High level"*)

for match in matcher:

   count+=1

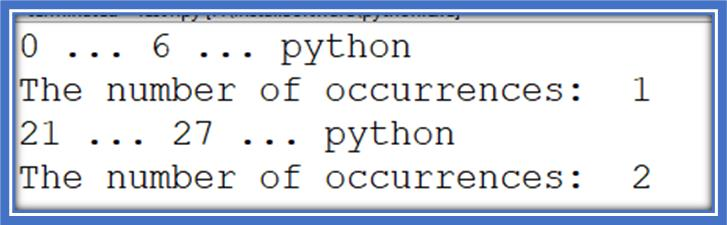
print(match.start(),*"..."*,match.end(),*"..."*,match.group())

   print(*"The number of occurrences: "*,count)



print(match.start(),*"..."*,match.end(),*"..."*,match.group())

   print(*"The number of occurrences: "*,count)



**Character classes:**

 We can use character classes to search a group of characters



import re

#RegEx pattern

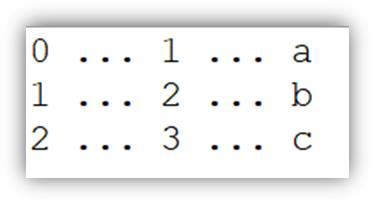
pattern = *'[abc]'*

matcher=re.finditer(pattern,*"abcdef"*)

for match in matcher:

[10:25 AM] Ashutosh (Unverified)

    print(match.start(),*"..."*,match.end(),*"..."*,match.group())



import re

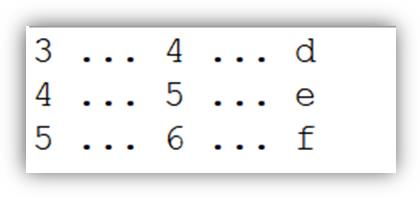
#RegEx pattern

pattern = *'[^abc]'*

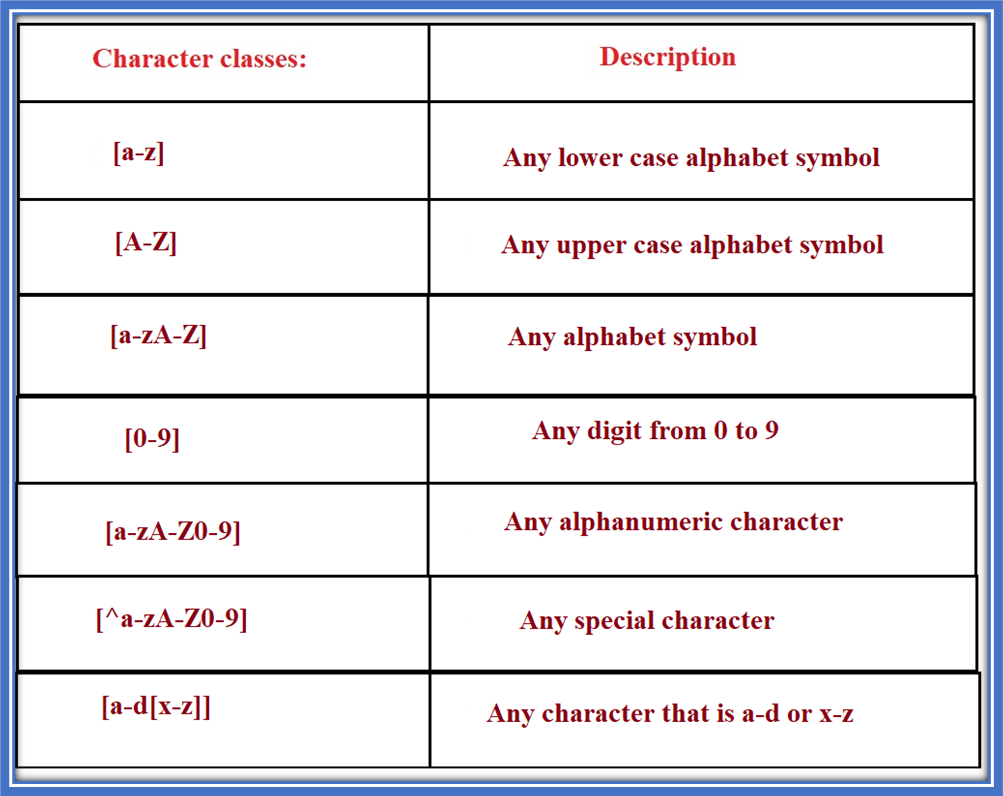
matcher=re.finditer(pattern,*"abcdef"*)

for match in matcher:

   print(match.start(),*"..."*,match.end(),*"..."*,match.group())



**How to use range and union Regex character classes**



[10:25 AM] Ashutosh (Unverified)

import re

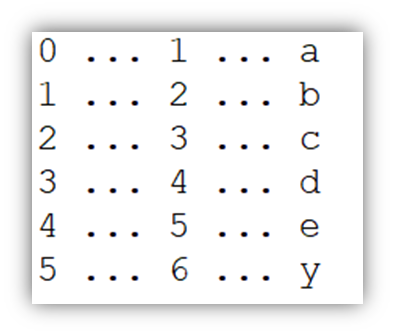
#RegEx pattern

pattern = *'[a-z]'*

matcher=re.finditer(pattern,*"abcdeyAB12"*)

for match in matcher:

   print(match.start(),*"..."*,match.end(),*"..."*,match.group())



[10:25 AM] Ashutosh (Unverified)

import re

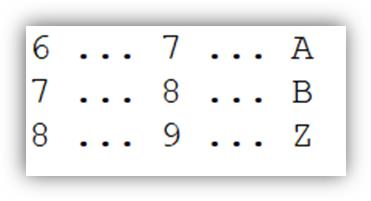
#RegEx pattern

pattern = *'[A-Z]'*

matcher=re.finditer(pattern,*"abcdeyABZ12"*)

for match in matcher:

   print(match.start(),*"..."*,match.end(),*"..."*,match.group())



import re

#RegEx pattern

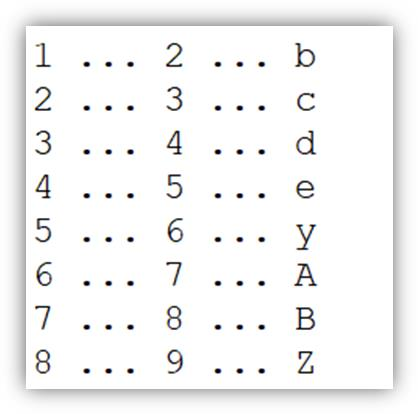
pattern = *'[a-zA-Z]'*

matcher=re.finditer(pattern,*"abcdeyABZ12"*)

[10:25 AM] Ashutosh (Unverified)

for match in matcher:

   print(match.start(),*"..."*,match.end(),*"..."*,match.group())



import re

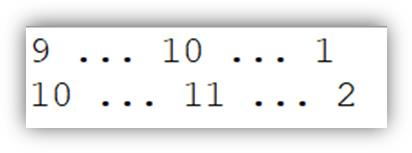
#RegEx pattern

pattern = *'[0-9]'*

matcher=re.finditer(pattern,*"abcdeyABZ12"*)

for match in matcher:

   print(match.start(),*"..."*,match.end(),*"..."*,match.group())



[10:26 AM] Ashutosh (Unverified)

import re

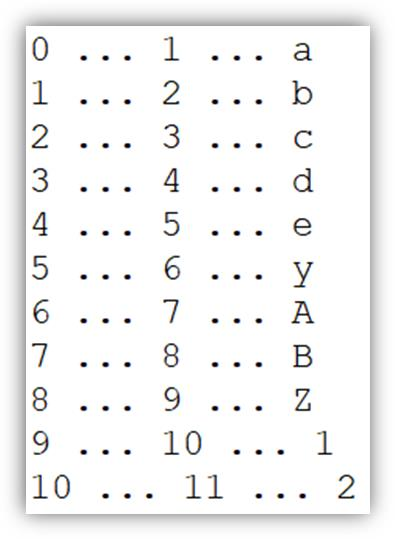
#RegEx pattern

pattern = *'[a-zA-Z0-9]'*

matcher=re.finditer(pattern,*"abcdeyABZ12"*)

for match in matcher:

   print(match.start(),*"..."*,match.end(),*"..."*,match.group())



import re

#RegEx pattern

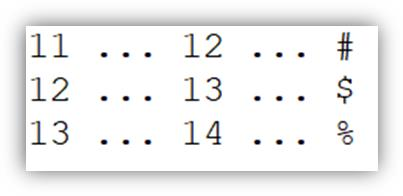
pattern = *'[^a-zA-Z0-9]'*

[10:26 AM] Ashutosh (Unverified)

matcher=re.finditer(pattern,*"abcdeyABZ12#$%"*)

for match in matcher:

   print(match.start(),*"..."*,match.end(),*"..."*,match.group())



1. [abc]===>Either a or b or c

2. [^abc] ===>Except a and b and c

3. [a-z]==>Any Lower case alphabet symbol

4. [A-Z]===>Any upper case alphabet symbol

5. [a-zA-Z]==>Any alphabet symbol

6. [0-9] Any digit from 0 to 9

7. [a-zA-Z0-9]==>Any alphanumeric character

8. [^a-zA-Z0-9]==>Except alphanumeric characters(Special Characters)

**Pre defined Character classes:**

\s  Space character

\S  Any character except space character

\d  Any digit from 0 to 9

\D  Any character except digit

\w  Any word character [a-zA-Z0-9]

\W  Any character except word character (Special Characters)

.   Any character including special characters

**Qunatifiers:**

* We can use quantifiers to specify the number of occurrences to match.

a  Exactly one 'a'

 a+  Atleast one 'a'

a\*  Any number of a's including zero number

 a?  Atmost one 'a' ie either zero number or one number

a{m}  Exactly m number of a's  a{m,n} 

a{m}  Exactly m number of a's

a{m,n}  Minimum m number of a's and Maximum n number of a's

 //-------------------------------------------------------------------

import re count=0 pattern=re.compile("pytho... by Ashutosh (Unverified)Ashutosh (Unverified)10:26 AM

import re

count=0

pattern=re.compile("python")

print(pattern)

sent="python is a prog and python is High level"

matcher=pattern.finditer(sent)

for match in matcher:

   print(match.start(),"...",match.end(),"...",match.group())

 //-------------------------------------------------------------------

import re #RegEx pattern pattern = '[abc]' ... by Ashutosh (Unverified)Ashutosh (Unverified)10:27 AM

import re

#RegEx pattern

pattern = '[abc]'

matcher=re.finditer(pattern,'abcdef')

for match in matcher:

   print(match.start(),"...",match.end(),"...",match.group())

  //-------------------------------------------------------------------

import re #RegEx pattern pattern = '[^abc]'... by Ashutosh (Unverified)Ashutosh (Unverified)10:27 AM

import re

#RegEx pattern

pattern = '[^abc]'

matcher=re.finditer(pattern,'abcdef')

for match in matcher:

   print(match.start(),"...",match.end(),"...",match.group())

  //-------------------------------------------------------------------

import re #RegEx pattern pattern = '[a-z]' ... by Ashutosh (Unverified)Ashutosh (Unverified)10:27 AM

import re

#RegEx pattern

pattern = '[a-z]'

matcher=re.finditer(pattern,"abcdeyAB12")

for match in matcher:

    print(match.start(),"...",match.end(),"...",match.group())

  //-------------------------------------------------------------------

import re pattern='[a]{1}' #returns a Match... by Ashutosh (Unverified)Ashutosh (Unverified)10:27 AM

import re

pattern='[a]{1}'

#returns a Match object if there is a match anywhere in the string

match=re.search(pattern,'Ajaay')

print(match)

if match:

    print('pattern matching')

else:

print('Not matching')

 //-------------------------------------------------------------------

import re txt="The rain in Spain" #findall(... by Ashutosh (Unverified)Ashutosh (Unverified)10:27 AM

import re

txt="The rain in Spain"

#findall() function returns a list containing all matcher

#The list contains the matches in the orderthey are found

#If nomatches are found, an emplty list is returned

x=re.findall('ai',txt)

print(x)

 //-------------------------------------------------------------------

import re txt="The rain in Spain" #checks f... by Ashutosh (Unverified)Ashutosh (Unverified)10:27 AM

import re

txt="The rain in Spain"

#checks for a match anywhere in the string

x=re.search('ai',txt)

print(x.group())

 //-------------------------------------------------------------------

import re txt="ai The rain in Spain" #check... by Ashutosh (Unverified)Ashutosh (Unverified)10:27 AM

import re

txt="ai The rain in Spain"

#checks for a match only at the beginining of the string

x=re.match('ai',txt)

print(x)

 //-------------------------------------------------------------------

import re txt="ai" #checks for a entire str... by Ashutosh (Unverified)Ashutosh (Unverified)10:27 AM

import re

txt="ai"

#checks for a entire string to be a match

x=re.fullmatch('ai',txt)

print(x)

 //-------------------------------------------------------------------

import re s1='bob has a birthday on Feb 25t... by Ashutosh (Unverified)Ashutosh (Unverified)10:28 AM

import re

s1='bob has a birthday on Feb 25th'

s2='sara has a birthday on March 3rd'

s3='12eup 586iu'

s4='0turt'

pattern1=re.compile('\w+ \d+\w+')

pattern2=re.compile('\w+ \w+')

x=pattern1.search(s1)

print(x)

x=pattern2.search(s2)

print(x)

pattern3=re.compile('\d\d\w+ \d\d')

x=pattern3.search(s3)

print(x)

 //-------------------------------------------------------------------

import re txt='The rain in Spain' x=re.spli... by Ashutosh (Unverified)Ashutosh (Unverified)10:28 AM

import re

txt='The rain in Spain'

x=re.split("\s",txt)

y=re.split("\s",txt,1)

print(x)

 //-------------------------------------------------------------------

import re txt='The rain in Spain' #sub() fu... by Ashutosh (Unverified)Ashutosh (Unverified)10:28 AM

import re

txt='The rain in Spain'

#sub() function replaces the matches with the text of your choice

x=re.sub("\s","9",txt)

print(x)

  //-------------------------------------------------------------------

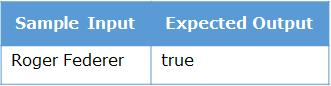
Problem Statement

Validate the name based on the below conditions using regular expression in the checkNameValidity() method.

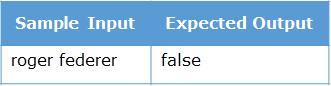
* The length of the name should be between 2 and 30 characters (both inclusive)
* The name can contain only alphabets and spaces
* The first character of each word of the name should be an upper case alphabet
* Each word should be separated by a space
* The name should not start or end with a space
* Special characters should not be allowed

Return true if the name is valid, else return false.

**Sample Input and Output**



IMG_257



IMG_259

//--------------------------------------------------------------------------------------

**MCQ Examples -**

Q1 of 6outlined\_flag

Which of the following strings will match the regular expression given below?

Hello (World|Program) .?

Hello World !

Hello Program !!!!

Hello World!

Hello Program

Hello World Program

Hello Program .

**Answer - Hello World !**

**//-----------------------------------------**

Q2 of 6outlined\_flag

Which of the following strings will match the regular expression given below?

[0-9]+ Pre?mium or{3}der

3445 Premium orrrder

57696 Prmium orrrder

Premium orrrder

57696 Prmium orrrrder

**Answer -**

**3445 Premium orrrder**

**57696 Prmium orrrrder**

**//-----------------------------------------**

Q3 of 6outlined\_flag

Which of the following strings will match the regular expression given below?

[\\w]+ [\\w]+

Steve FoodOrder\_12

Steve!Smith FoodOrder1

John\_Smith FoodOrder=

Steve 12 FoodOrder\_1\_12

**Answer -**

**Steve FoodOrder\_12**

**John\_Smith FoodOrder=**

**//-----------------------------------------**

Q4 of 6outlined\_flag

Which of the following strings will match the regular expression given below?

[\\D]+-[A-Z]+Ab\\\*[0-9]{3,5}

Steve-FOODORDERAbc\*1123

Steve-FOODORDERAb\*1123678

Steve1\*-FOODORDERAb\*11236

Steve-FOODORDERAb\*1123

**Answer -**

**Steve-FOODORDERAbc\*1123**

**Steve1\*-FOODORDERAb\*11236**

**//-----------------------------------------**

Q5 of 6outlined\_flag

Which of the following regular expressions will match the string **Steve@abc.co.in**?

\\w+@abc\\.(co\\.in)

\\w+\\d+@\\w+\\.co.in

[\\w+@]+[\\D]+

[a-zA-Z@]+[0-9]+

[\\.\*Steve\\.\*](\\\\.*Steve\\.*)

**Answer - \\w+@abc\\.(co\\.in)**

**//-----------------------------------------**

Q6 of 6outlined\_flag

Which of the following regular expressions will match CD\*56FT\*A\*54?

\\w\\\*\\w+\\\*A\\\*\\d+

CD\\\*\\W\\w+\\W\\w+\\\*[0-9]4

[^\\d]\\\*[\\w-]+

[\\w{2}\\\*\\d{2}\\w{2}\\\*\\w{1}\\\*\\d{2}](\\\\w{2}\\*\\d{2}\\w{2}\\*\\w{1}\\*\\d{2})

**Answer - [\\w{2}\\\*\\d{2}\\w{2}\\\*\\w{1}\\\*\\d{2}](\\\\w{2}\\*\\d{2}\\w{2}\\*\\w{1}\\*\\d{2})**

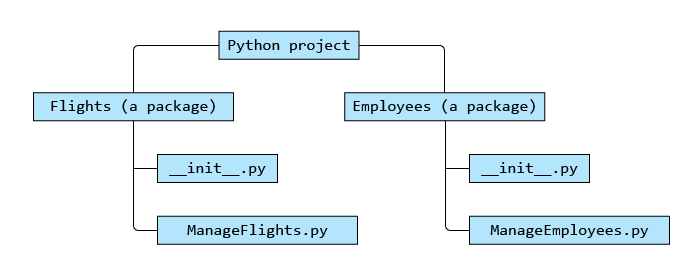
//------------------------------------------------------------------------------------------------

//----------------------------------------------------------------------------------------

**Modules and Packeges -**

Code organization is done through modules and packages!

* A module is nothing but a normal python file with a .py extension.
* A package is just a normal folder which contains an empty file who’s name is \_\_init\_\_.py. Since package is a folder it can store many python modules/files in it.



Let's assume that the ManageFlights.py inside Flights package has the below code:

airline="Smart Airlines"def add(no\_of\_flights):    print(no\_of\_flights," flights added to the fleet")

If we need to access the function add(), inside the ManageFlights module in some other module, then we can import the ManageFlights module and use it.

Import can be done in two ways:

**Method 1:**

from Flights import ManageFlights #from packagename import modulenameManageFlights.add(10)

**Method 2:**

import Flights.ManageFlights #import packagename.modulenameFlights.ManageFlights.add(10)

//---------------------------------------------------------------------

//----------------------------------------------------------------------

**Non Linear Data Structure -**

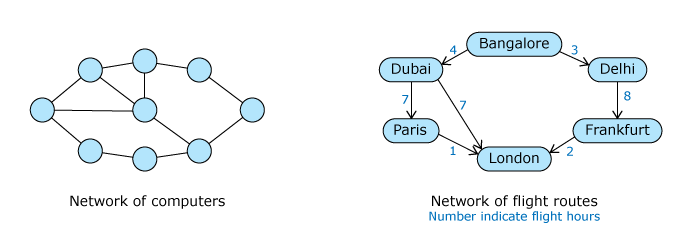
1. **Graph**

# Non - Linear Data Structures - Introduction

# ****Scenario-1:****

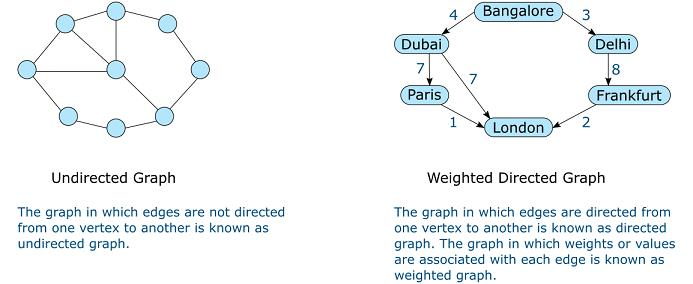
The network engineers of a company are trying to connect all the computers (numbered 1 to 9) in the company network. They also need to provide a path to traverse from one computer to the other.

**Scenario-2:**  
Maria is planning a vacation trip to Europe and is trying to choose the best air route from the available options based on the travel time in hrs.  
Option-1: Bangalore(4hrs) -> Dubai(7hrs)-> Paris(1hr)-> London  
Option-2: Bangalore(3hrs)->Delhi(8hrs)->Frankfurt(2hrs)->London  
Option-3: Bangalore(4hrs)->Dubai(7hrs)->London

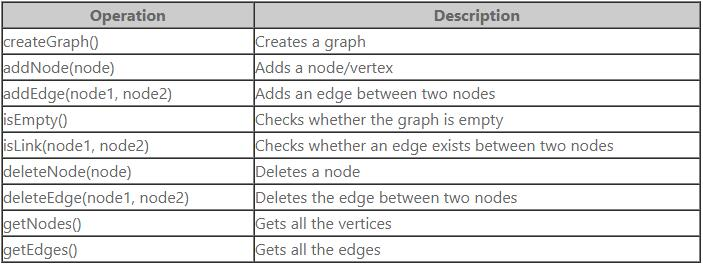


Graph                                                            In these scenarios, we understand that we cannot use any of the linear data structures like array, linked list, stack or queue to represent it. Here, we need an arrangement which allows to have a set of vertices and edges between them. Such a data structure is known as **graph**.

**Graph** is a non-linear data structure having a set of **vertices**(or **nodes**) and edges between vertices. It can have any number of edges and nodes and any node can be connected to any other node by an edge. It can be implemented using arrays or linked lists.



Common operations possible on graph are listed below:



//---------------------------------------------------------------------------------------------------

1. **Tree -**

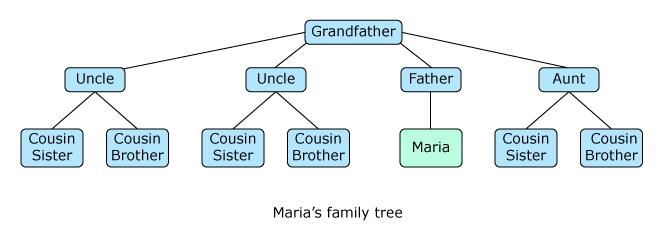
# ****Scenario-1:****

Maria is preparing her family tree. She has her grandfather as the head of the family. Her grandfather has three sons and one daughter. Her aunt and uncles have one son and one daughter each. Maria is the only child of her father.

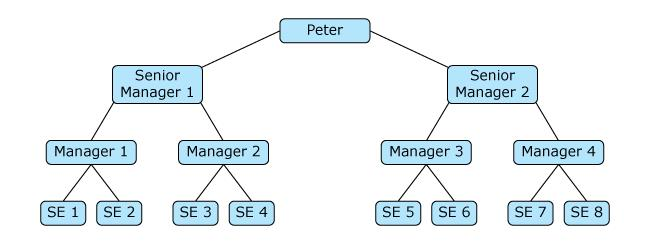
**Scenario-2:**

Peter who has become the head of his team is preparing the organization structure. He has two senior managers reporting into him. To each of the senior managers, there are two managers reporting and to each of the managers, there are again two software engineers reporting.

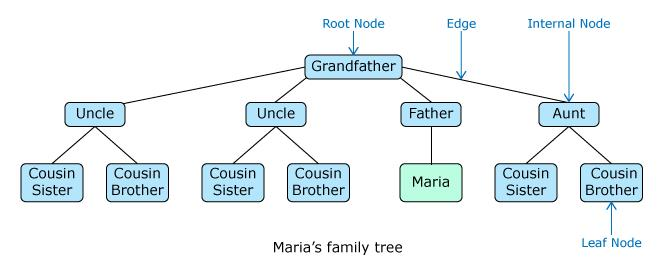
**Scenario-1: Family Tree**



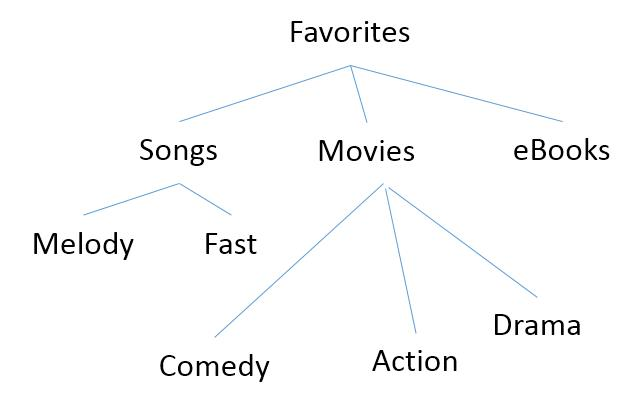
**Scenario-2: Organization Structure**



This type of non-linear arrangement where a node is attached to one or more nodes directly beneath it, is a special type of graph known as **tree**. In this data structure, the top most node is called the root and the connections between nodes are called **edges**. Nodes that have no children are called **leaf nodes** and non-root and non-leaf nodes are called **internal nodes**.



Trees are used extensively in searching and sorting applications.



//-------------------------------------------------------------------------------

**MCQ Example -**

Q1 of 4outlined\_flag

A travel agent plans trips for tourists from Chicago to Miami. He gives them three ways to get from town to town: airplane, bus, train.  
Once the tourists arrive, there are two ways to get to the hotel: hotel van or taxi. The cost of each type of transportation is given in the table below.

Which data structure can be used to represent all the possible ways to go from Chicago to Miami?

|  |  |
| --- | --- |
| **Transportation Type** | **Cost** |
| Airplane | $ 350 |
| Bus | $ 150 |
| Train | $ 225 |
| Hotel Van | $ 60 |
| Taxi | $ 40 |

Tree

Graph

Queue

Stack

**Answer - Graph**

//-----------------------------------------------------

Q2 of 4outlined\_flag

In an ATM, a customer can get a mini statement of his/her last 5 transactions.  
Which is the most appropriate data structure to model the mini statement?

List

Queue

Stack

Graph

**Answer - Queue**

//-----------------------------------------------------

Q3 of 4outlined\_flag

Which is the most appropriate data structure to model the token for cash withdrawal in a bank?  
1st token is provided to the first person who has arrived to withdraw money. The next person is provided the 2nd token etc. The person who got the first token will be serviced first.

List

Queue

Stack

Graph

**Answer - Queue**

//-----------------------------------------------------

Q4 of 4outlined\_flag

Mary wants to model all the items in a retail store under various categories and sub-categories.  
Which of the below data structures can be used to optimally represent it?

Tree

List

Queue

Stack

Graph

**Answer - Tree**

//-----------------------------------------------------

//--------------------------------------------------------------------------------

**Non Linear Data Structure -**

1. **Stack -**

**Algorithms -**

**push(data):**

1. 1. Check whether the stack is full. If full, display appropriate message
2. 2. If not,
3. a. increment top by one
4. b. Add the element at top position in the elements array

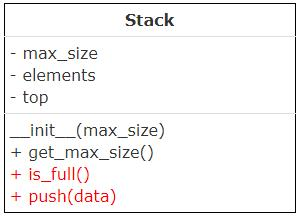
push(data):  
1. Check whether the stack is full. If full, display appropriate message  
2. If not,  
   a. increment top by one  
   b. Add the element at top position in the elements array

**Stack - Pop operation  pop:**  
1. Check whether the stack is empty. If empty, display appropriate message  
2. If not,  
   a. Retrieve data at the top of the stack  
   b. Decrement top by 1  
   c. Return the retrieved data

//---------------------

# Stack - Push Operation - Try outProblem Statement

Write a program to represent a Stack class as discussed earlier and implement push(data) and is\_full() methods to implement the push operation based on the algorithm discussed.  
  
Once done, represent Peter’s shirts as a stack of maximum size 5 and push the following shirts into it.



|  |
| --- |
| 1. Shirt1  2. Shirt2  3. Shirt3  4. Shirt4  5. Shirt5  6. Shirt6 |

class Stack:  
    def \_\_init\_\_(self,max\_size):  
        self.\_\_max\_size=max\_size  
        self.\_\_elements=[None]\*self.\_\_max\_size  
        self.\_\_top=-1

    def get\_max\_size(self):  
        return self.\_\_max\_size

    def is\_full(self):  
        pass  
        #Remove pass and write the logic to check whether stack is full. If the stack is full, return true else return false.

    def push(self,data):  
        pass  
        #Remove pass and write the logic to push element into the stack. Element should be pushed only if the stack is not full. Otherwise, display appropriate message  
  
    #You can use the below \_\_str\_\_() to print the elements of the DS object while debugging  
    def \_\_str\_\_(self):  
        msg=[]  
        index=self.\_\_top  
        while(index>=0):  
            msg.append((str)(self.\_\_elements[index]))  
            index-=1  
        msg=" ".join(msg)  
        msg="Stack data(Top to Bottom): "+msg  
        return msg

stack1=Stack(5)  
#Push all the required element(s).  
stack1.push("Shirt1")

//---------------------------------------