**\*\*PYTHON TEST-1\*\***

“AITB”

* **ENG. Mohab Essam Aboud**

QUESTION 1: Answer the following questions:

1. What are lists and tuples? what is the key difference between the two?

List and Tuple are **a class of data structures that can store one or more objects or values**.

A list is used to store multiple items in one variable and can be created using square brackets.

Similarly, tuples can store multiple items in a single variable and be declared using parentheses and we can use it without parentheses.

1. What is break, continue and pass in python?

-Break: Terminate the current loop. Use the break statement to come out of the loop instantly.

- Continue: Skip the current iteration of a loop and move to the next iteration.

-Pass: Do nothing. Ignore the condition in which it occurred and proceed to run the program as usual.

1. What is the use of self in Python?

The self is used to represent the [instance](https://www.edureka.co/blog/isinstance-in-python/) of the class. With this keyword, you can access the attributes and methods of the [class in python](https://www.edureka.co/blog/python-class/). It binds the attributes with the given arguments. The reason why we use self is that Python does not use the ‘@’ syntax to refer to instance attributes.

1. What is docstring in Python?

Python documentation strings (or docstrings) **provide a convenient way of associating documentation with Python modules, functions, classes, and methods**. It's specified in source code that is used, like a comment, to document a specific segment of code.

1. Python supports multiple inheritances. explain it with an example.

Inheritance is the mechanism to achieve the re-usability of code as one class(child class) can derive the properties of another class(parent class). It also provides transitivity ie. if class C inherits from P then all the sub-classes of C would also inherit from P.

**Multiple Inheritance**   
When a class is derived from more than one base class it is called multiple Inheritance. The derived class inherits all the features of the base case.

class Class1:

    def m(self):

        print("In Class1")

class Class2(Class1):

    def m(self):

        print("In Class2")

class Class3(Class1):

    def m(self):

        print("In Class3")

class Class4(Class2, Class3):

    pass

obj = Class4()

obj.m()

Question2 : True or false and

TRUE

FALSE

TRUE

FALSE

TRUE

Question3: Write Python Code to solve these problems  
1- Create a function that takes a string and returns the number  
(count) of vowels contained within it.  
Example:  
count\_vowels("Celebration") ➞ 5--------------------------------

def isVowel(ch):

    return ch.upper() in ['A', 'E', 'I', 'O', 'U']

# to count total number of

# vowel from 0 to n

def countVovels(str, n):

    if (n == 1):

        return isVowel(str[n - 1]);

    return (countVovels(str, n - 1) +

                isVowel(str[n - 1]));

# Driver Code

# string object

str = "celebration";

# Total numbers of Vowel

print("Total numbers of Vowel =",countVovels(str, len(str)))

2- Write a function that finds the sum of the first n natural  
numbers. Make your function recursive.  
Example:  
sum\_numbers(5)

def recur\_sum(n):

if n <= 1:

return n

else:

return n + recur\_sum(n-1)

# change this value for a different result

num = 16

if num < 0:

print("Enter a positive number")

else:

print("The sum is",recur\_sum(num))

Write a Python program to check if value 200 exists in the  
following dictionary.  
Given:  
sample\_dict = {'a': 100, 'b': 200, 'c': 300}

# Function to print sum

def checkKey(dict, key):

    if key in dict.keys():

        print("Present, ", end =" ")

        print("value =", dict[key])

    else:

        print("Not present")

# Driver Code

dict = {'a': 100, 'b':200, 'c':300}

key = 'b'

checkKey(dict, key)

key = 'w'

checkKey(dict, key)

`

Question 5 : Explain only by example  
1- Default constructor vs parameterized constructor

* **Default constructor**

class DemoClass:

# constructor

def \_\_init\_\_(self):

# initializing instance variable

self.num=100

# a method

def read\_number(self):

print(self.num)

# creating object of the class. This invokes constructor

obj = DemoClass()

# calling the instance method using the object obj

obj.read\_number()

* **Parameterized constructor:**

class DemoClass:

num = 101

# parameterized constructor

def \_\_init\_\_(self, data):

self.num = data

# a method

def read\_number(self):

print(self.num)

# creating object of the class

# this will invoke parameterized constructor

obj = DemoClass(55)

# calling the instance method using the object obj

obj.read\_number()

# creating another object of the class

obj2 = DemoClass(66)

# calling the instance method using the object obj

obj2.read\_number()

2- Class vs object.

* **Class:**

class Person:

"This is a person class"

age = 10

def greet(self):

print('Hello')

# Output: 10

print(Person.age)

# Output: <function Person.greet>

print(Person.greet)

# Output: "This is a person class"

print(Person.\_\_doc\_\_)

* **Object:**

class Person:

"This is a person class"

age = 10

def greet(self):

print('Hello')

# create a new object of Person class

harry = Person()

# Output: <function Person.greet>

print(Person.greet)

# Output: <bound method Person.greet of <\_\_main\_\_.Person object>>

print(harry.greet)

# Calling object's greet() method

# Output: Hello

harry.greet()