Python Programming Question Bank

Unit 1 -

▼ Explain any five features of Python Programming

1. Easy to Use
2. Expressive Language (Understandable)
3. Open-Source
4. Cross Platform
5. Supports OOP
6. Have Many Libraries
7. GUI can be made

▼ Explain nested if-else statement with an example

▼ Explain Logical & Membership operators in python with an example

▼ Describe control statements in python.

▼ Write a short note on break & continue statements with an example

break is used to exit a for loop or a while loop, whereas continue is used to skip the current block, and return to the "for" or "while" statement.

▼ Write a program that accepts a number from the user and prints whether the number is even or odd

```
num = int(input("Enter any integer number: "))
mod = num%2

if mod > 0:
    print("Your number is odd")
else:
    print("Your number is even")
```

▼ Write a python program to calculate the area of a triangle and print the result. Take input from user

Area of a Triangle = A = 1/2 × b × h

▼ Explain the if & elif statements with an example

▼ Explain arithmetic operators with example

▼ Explain range() function with example

▼ Explain Relational & Assignment operators in python

▼ List the datatypes in python. Explain any two datatypes with an example

▼ Write a python program to calculate the area of a circle and print the result. Take input from user

Area of Circle = A = πr^2

▼ Explain for loop with an example

▼ Write a short note on range function in python with example

▼ Write a python program to print factorial of a number. Take input from user

```
a = int(input("Enter any number: "))
fact = 1

for x in range(1, a+1):
    fact = fact*x
print(fact)
```

▼ Write a program that accepts the name and age of a user. Print a message that tells them the year that they will turn 100 years

```
name = input("Enter your name: ")
age = int(input("Enter your age: "))
year = 2022 + (100-age)

print(name, "you will turn 100 in", year, "years")
```

▼ If the ages of Ankit, Raj and Minal are input through the keyboard, write a program to determine the eldest of the three.

Unit 2 -

▼ What is a function. Write syntax to define a function with an example.

▼ Discuss the local and global variable scope in python

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope. This means that local variables can be accessed only inside the function in which they are declared, whereas global variables can be accessed throughout the program body by all functions.

▼ Explain following string functions with example i) isdigit() ii) upper() iii) rstrip() iv) find() v) split()

1. returns True if all the characters are digits, otherwise False
2. returns a string where all characters are in upper case
3. removes any trailing characters (characters at the end a string)
4. returns the lowest index or first occurrence of the substring if it is found in a given string. If it is not found, then it returns -1.
5. splits a string into a list

▼ Explain Boolean function with an example.

The Python Boolean type is one of Python's built-in data types. It's used to represent the truth value of an expression. For example, the expression 1 <= 2 is True , while the expression 0 == 1 is False . Understanding how Python Boolean values behave is important to programming well in Python.

▼ Justify the statement “The Strings in Python are immutable”.

In Python, strings are made immutable so that programmers cannot alter the contents of the object (even by mistake). This avoids unnecessary bugs. Some other immutable objects are integer, float, tuple, and bool. More on mutable and immutable objects in Python.

▼ Explain the following math functions with example i)pow() ii)sqrt() iii)floor() iv)ceil() v)fabs()

1. math.pow(x, y) - Return x raised to the power y
2. math.sqrt(x) - Return the square root of x.
3. math.floor(x) - Return the floor of x, the largest integer less than or equal to x. If x is not a float, delegates to x.**floor**(), which should return an Integral value.
4. math.ceil(x) - Return the ceiling of x, the smallest integer greater than or equal to x. If x is not a float, delegates to x.**ceil**(), which should return an Integral value.
5. math.fabs(x) - Return the absolute value of x.

▼ Write a note on “import” in python with an example

Import in python is similar to #include header_file in C/C++. Python modules can get access to code from another module by importing the file/function using import. The import statement is the most common way of invoking the import machinery, but it is not the only way.

ex. import module_name

Code -
import math

```
pie = math.pi
print("The value of pi is : ",pie)
```

▼ Write a program to traverse a string in python.

▼ Write a program to calculate area of rectangle using function.

Area of Rectangle = A = wl

▼ Define a function. What are the features of functions? Write a program to explain a function.

Features -
Functions Are Objects.
Functions Can Be Stored in Data Structures.
Functions Can Be Passed to Other Functions.
Functions Can Be Nested.
Objects Can Behave Like Functions.

▼ Describe string slicing with an example

```
arr[start:stop]      # items start through stop-1
arr[start:]          # items start through the rest of the array
arr[:stop]           # items from the beginning through stop-1
arr[:]               # a copy of the whole array
arr[start:stop:step] # start through not past stop, by step
```

▼ Explain any five string functions.

1. lower(): Converts all uppercase characters in a string into lowercase
2. upper(): Converts all lowercase characters in a string into uppercase
3. title(): Convert string to title case
4. center(): Pad the string with the specified character
5. find(): Returns the lowest index of the substring if it is found

▼ Write a python program to reverse a number. Take input from user.

▼ Write a note on “composition”.

Composition is a concept that models a has a relationship. It enables creating complex types by combining objects of other types. This means that a class Composite can contain an object of another class Component . This relationship means that a Composite has a Component .

▼ Write a python program to check whether a number is a Palindrome or not. Take input from user.

▼ Write the output for following string operations. s = 'HelloWorld' i) s[::-1] ii) s[2:8:2] iii) s[8:1:-1] iv) s[8:1:-2] v) s[-4:-2]

▼ Write a program to generate the Fibonacci series up to 9 numbers.

```
def fib(n):
    a = 0
    b = 1

    print(a)
    print(b)

    for i in range(2, n):
        c = a+b
        a = b
        b = c

        print(c)

n = int(input("How many terms you wants: "))
fib(n)
```

▼ Write a program to count number of alphabets and number of digits in the string ‘VSIT0011’

```
def collen(n):
    count = 0
    for i in n:
        count += 1
    return count
```

```
print("The lenght of the list is ", collen(
‘VSIT0011'))
```

Unit 3 -

▼ What are lists? Give an example to access the elements of list.

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets.

For example −
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5, 6, 7]

print ("list1[0]: ", list1[0])
print ("list2[1:5]: ", list2[1:5])

When the above code is executed, it produces the following result −
list1[0]: physics
list2[1:5]: [2, 3, 4, 5]

▼ Explain del(), clear() and remove() methods of list with example

del(): Remove items by index or slice
clear(): Remove all items in the list
remove(): Remove an item by value

del() -
l = [0, 10, 20, 30, 40, 50]
del l[:]
print(l)

[ ]

clear() -
l = [0, 1, 2]
l.clear()
print(l)

[ ]

remove():
l = ['Alice', 'Bob', 'Charlie', 'Bob', 'Dave']
l.remove('Alice')
print(l)

['Bob', 'Charlie', 'Bob', 'Dave']

▼ What are Tuples? How to define and access the elements of tuple.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also

For example -
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5, 6, 7)

print ("tup1[0]: ", tup1[0])
print ("tup2[1:5]: ", tup2[1:5])

When the above code is executed, it produces the following result −
tup1[0]: physics
tup2[1:5]: [2, 3, 4, 5]

▼ Write a short note on Tuple operations.

Tuples are used to store multiple items in a single variable.

Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Set, and Dictionary, all with different qualities and usage.

A tuple is a collection which is ordered and **unchangeable**.

Tuples are written with round brackets.

▼ Tuples are immutable. Justify with an example.

In Python, strings are made immutable so that programmers cannot alter the contents of the object (even by mistake). This avoids unnecessary bugs. Some other immutable objects are integer, float, tuple, and bool. More on mutable and immutable objects in Python.

▼ Explain exception handling block in Python with the help of an example

▼ Define List. Explain the following functions of list with example: append() ii) extend() iii)insert()

1. inserts a single element into an existing list
2. adds the specified list elements (or any iterable) to the end of the current list.
3. inserts the specified value at the specified position.

▼ How to create dictionary in python? Give example

Python dictionary is an unordered collection of items. While other compound data types have only value as an element, a dictionary has a key: value pair. Dictionaries are optimized to retrieve values when the key is known.

Creating a dictionary is as simple as placing items inside curly braces {} separated by comma. An item has a key and the corresponding value expressed as a pair, key: value.

empty dictionary

my_dict = {}

dictionary with integer keys

my_dict = {1: 'apple', 2: 'ball'}

dictionary with mixed keys

```
my_dict = {'name': 'John', 1: [2, 4, 3]}
```

▼ Describe different access modes of file handling.

1. Read Only (‘r’) : Open text file for reading. The handle is positioned at the beginning of the file. If the file does not exists, raises the I/O error. This is also the default mode in which a file is opened.
2. Read and Write (‘r+’): Open the file for reading and writing. The handle is positioned at the beginning of the file. Raises I/O error if the file does not exist.
3. Write Only (‘w’) : Open the file for writing. For the existing files, the data is truncated and over-written. The handle is positioned at the beginning of the file. Creates the file if the file does not exist.
4. Write and Read (‘w+’) : Open the file for reading and writing. For an existing file, data is truncated and over-written. The handle is positioned at the beginning of the file.
5. Append Only (‘a’): Open the file for writing. The file is created if it does not exist. The handle is positioned at the end of the file. The data being written will be inserted at the end, after the existing data.
6. Append and Read (‘a+’) : Open the file for reading and writing. The file is created if it does not exist. The handle is positioned at the end of the file. The data being written will be inserted at the end, after the existing data.

▼ Explain mkdir(), chdir() and rmdir() methods in python

1. used to create a directory named path with the specified numeric mode
2. used to change the current working directory to specified path
3. used to remove or delete a empty directory

▼ Explain following dictionary methods with example: keys(), values() & update()

▼ Write a program to sort a dictionary in ascending and descending order of values.

```
from audioop import reverse
import operator

dic = {1:'c', 2:'b', 3:"a"}
print(dic)

sort = sorted(dic.items(), key=operator.itemgetter(0))
print(sort)

sort = sorted(dic.items(), key=operator.itemgetter(0), reverse = True)
print(sort)
```

▼ Write a Python program to print a specified list after removing the 0th, 2nd, 4th and 5th elements.

```
list = ['Kamal', 'Nishant', "Om", 'Santosh', 'Amit']
newlist = []

for i in list:
    if list.index(i) == 1 or list.index(i) == 3:
        newlist.append(i)

print(newlist)
```

▼ Write a program that prints out all the elements from the list that are less than 5.

▼ Write a Python program to sum all the items in a list.

```
list1 = [11, 5, 17, 18, 23]

for e in range(0, len(list1)):
    total = total + list1[e]

print("Sum of all elements in given list: ", total)
```

▼ Write a python code to get the following dictionary as output {1:1,3:9,5:25,7:49,9:81}

Unit 4 -

▼ Explain various types of regular expressions with example.

A regular expression is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern.

A Regular Expression (RegEx) is a sequence of characters that defines a search pattern.

For example - ^a...s$

The above code defines a RegEx pattern. The pattern is: any five-letter string starting with a and ending with s.

▼ Explain match function with example.

▼ Explain search function with example.

▼ What is constructor? Explain the rules for creating a constructor.

A constructor is the first method that is called on object creation (a concept from Object Orientated Programming). It is always part of a class (an objects methods are defined in a class).

The constructor is always called when creating a new object. It can be used to initialize class variables and startup routines.

There are a few rules for constructors -

1. A constructor must be defined with the name **init**.
2. A constructor must be defined with the self keyword in its parameters.
3. A constructor cannot return any value, except None.
4. Only one constructor is allowed for a class.
5. Only for object initialization

▼ Write a short note on thread module.

▼ Explain Multithreading with example.

A thread is the smallest unit of a program or process executed independently or scheduled by the Operating System. In the computer system, an Operating System achieves multitasking by dividing the process into threads. A thread is a lightweight

process that ensures the execution of the process separately on the system. In Python 3, when multiple processors are running on a program, each processor runs simultaneously to execute its tasks separately.

Multithreading is a threading technique in Python programming to run multiple threads concurrently by rapidly switching between threads with a CPU help (called context switching). Besides, it allows sharing of its data space with the main threads inside a process that share information and communication with other threads easier than individual processes. Multithreading aims to perform multiple tasks simultaneously, which increases performance, speed and improves the rendering of the application.

Benefits of Multithreading in Python -

Following are the benefits to create a multithreaded application in Python, as follows:

It ensures effective utilization of computer system resources.

Multithreaded applications are more responsive.

It shares resources and its state with sub-threads (child) which makes it more economical.

It makes the multiprocessor architecture more effective due to similarity.

It saves time by executing multiple threads at the same time.

The system does not require too much memory to store multiple threads.

▼ With the help of program demonstrate the concept of method overriding

Method overriding is an ability of any object-oriented programming language that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes. When a method in a subclass has the same name, same parameters or signature and same return type(or sub-type) as a method in its super-class, then the method in the subclass is said to **override**
the method in the super-class.

```
# Python program to demonstrate
# method overriding
```

```
# Defining parent class
class Parent():

    # Constructor
    def __init__(self):
        self.value = "Inside Parent"

    # Parent's show method
    def show(self):
        print(self.value)

# Defining child class
class Child(Parent):

    # Constructor
    def __init__(self):
        self.value = "Inside Child"

    # Child's show method
    def show(self):
        print(self.value)


# Driver's code
obj1 = Parent()
obj2 = Child()

obj1.show()
obj2.show()
```

▼ What is Inheritance? Explain Multilevel inheritance with a suitable example

The capability of a **class** to derive properties and characteristics from another class is called **Inheritance**. Inheritance is one of the most important features of Object-Oriented Programming.

Inheritance is a feature or a process in which, new classes are created from the existing classes. The new class created is called “derived class” or “child class” and the existing class is known as the “base class” or “parent class”. The derived class now is said to be inherited from the base class.

When we say derived class inherits the base class, it means, the derived class inherits all the properties of the base class, without changing the properties of base class and may add new features to its own. These new features in the derived class

will not affect the base class. The derived class is the specialized class for the base class.

- **Sub Class:** The class that inherits properties from another class is called Subclass or Derived Class.
- **Super Class:** The class whose properties are inherited by a subclass is called Base Class or Superclass.

Multiple Inheritance is a feature of C++ where a class can inherit from more than one class. i.e one **subclass** is inherited from more than one **base class**.

```
Class Base1:
       Body of the class

Class Base2:
     Body of the class

Class Derived(Base1, Base2):
     Body of the class
```

```
# Python Program to depict multiple inheritance
# when method is overridden in both classes

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()
```

▼ Demonstrate the use of random module functions with suitable example

▼ Write a Python program that matches a string that has an a followed by zero or more b's

▼ Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle

▼ Write a Python program that matches a word containing 'z'

▼ Write a Python program where a string will start with a specific number.

▼ Explain class attribute with example.

▼ Write a Python class named Rectangle constructed by a length and width and a method which will compute the area of a rectangle

▼ Define Inheritance. Explain Hybrid inheritance with a suitable example

Inheritance consisting of multiple types of inheritance is called hybrid inheritance.

```
# Python program to demonstrate
# hybrid inheritance


class School:
  def func1(self):
    print("This function is in school.")


class Student1(School):
  def func2(self):
    print("This function is in student 1. ")


class Student2(School):
  def func3(self):
    print("This function is in student 2.")


class Student3(Student1, School):
  def func4(self):
    print("This function is in student 3.")


# Driver's code
object = Student3()
object.func1()
object.func2()
```

▼ Write a Python class which has two methods get_String and print_String. get_String accept a string

▼ rom the user and print_String print the string in upper case

Unit 5 -

▼ Explain the place layout manager in python with the help of an example

https://itvoyagers.in/layout-manager-geometry-manager-in-python/

▼ How to enhance the look and feel of GUI Application

1. Keep the inaterface simple.
2. Create consistency and use common UI elements.
3. Be purposeful in page layout.
4. Strategically use color and texture.
5. Use typography to create hierarchy and clarity.
6. Make sure that the system communicates what's happening.
7. Think about the defaults.

▼ Explain the grid layout manager in python with the help of an example

▼ Write a short note on scale widget.

▼ Write a short note on spin box.

https://www.computerhope.com/jargon/s/spinbox.htm

▼ Explain Check button widget with example.

https://www.geeksforgeeks.org/python-tkinter-checkbutton-widget/