**Q1. List out different OOPS principles and explain?**

We have 4 important principles for Python language to make it an Object-Oriented Programming Language.

**1. Encapsulation:**

We keep each object private inside a **class** keyword, so other objects don’t have direct access to it. Instead, they can call a list of public functions called methods.

class Circle:

pi = 3.14

# Circle gets instantiated with a radius (default is 1)

def \_\_init\_\_(self, radius=1):

self.radius = radius

self.area = radius \* radius \* Circle.pi

# Method for resetting Radius

def setRadius(self, new\_radius):

self.radius = new\_radius

self.area = new\_radius \* new\_radius \* self.pi

c = Circle()

print('Radius is: ',c.radius)

**2. Abstraction:**

Abstraction in Python is the process of hiding the real implementation of an application from the user and emphasizing only how to use the application.

class rectangle(Circle):

# Method to get size of rectangle

def setRadius(self, l, w):

self.area = l \* w

**3. Inheritance:**

Inheritance is a way to form new classes using classes that have already been defined. The newly formed classes are called derived classes, the classes that we derive from are called base classes. Important benefits of inheritance are code reuse and reduction of complexity of a program.

class Animal:

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

print("Animal created")

def whoAmI(self):

print("Animal")

def eat(self):

print("Eating")

class Dog(Animal):

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

Animal.\_\_init\_\_(self)

print("Dog created")

def whoAmI(self):

print("Dog")

def bark(self):

print("Woof!")

# If we intanciate it and capture in a variable d:

d = Dog()

print(d)

# Output:

*Animal created*

*Dog created*

**4. Polymorphism:**

In Python, polymorphism refers to the way in which different object classes can share the same method name, and those methods can be called from the same place even though a variety of different objects might be passed in.

class Dog:

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

self.name = name

def speak(self):

return self.name+' says Woof!'

class Cat:

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

self.name = name

def speak(self):

return self.name+' says Meow!'

# Lets capture methods of these two different objects in variables:

niko = Dog('Niko')

felix = Cat('Felix')

print(niko.speak())

print(felix.speak())

# Output:

*Niko says Woof!*

*Felix says Meow!*

**Q2. List out Layers of TCP/IP Model and explain?**

There are four layers which are as follows:

**1. Application layer:**

It is a Top most layer which is equivalent to Application, Presentation and Session layer.

It uses protocols like SMTP, HTTP, HTTPS, FTP, NTP, SSH, Telnet SNMP etc.

**2. Host-to-Host layer:**

It is equivalent to the transport layer of the OSI model and is responsible for delivery of data. It depends on which type of protocol is used whether it’s Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). If reliable delivery of data is needed, then TCP will be used. UDP is used, if fast delivery without reliability is needed. In this layer, TCP and UDP the protocols which are used and data acts as a Protocol Data Unit (PDU).

**3. Internet layer:**

Logical addressing and routing is performed at this layer by using the IP address of a device. At this layer the routers forward the packet on the basis of IP addresses of devices. IP is the most important protocol at this layer.

**4. Network Access layer:**

It is equivalent to the Data link layer and Physical layer. Hardware addressing is done at this layer. It defines protocols for physical transmission of data. Ethernet is the most important protocol used at this layer. Fast Ethernet, Token Ring and Fiber Distributed Data Interface (FDDI) are other protocols at this layer.

**Q3. Q3. Construct a binary tree by using postorder and inorder sequences given below. Inorder: N, M, P, O, Q Postorder: N, P, Q, O, M**



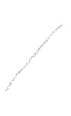
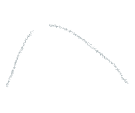
**Inorder:**



In this order traversal happens on the basis of **L Root R**

So, the traversal will be: **N, M, P, O, Q**







**Postorder:**

In this order traversal happens on the basis of **L R Root**

So, the traversal will be: **N, P, Q, O, M**



**Q5. Explain LRU cache and its implementation:**

LRU (Least Recently Used) cache clean-up algorithm is a common strategy. According to the name, the latest used data should be useful. Hence, when the memory cache is full, we should prioritize removing those data that haven't been used for long are not useful.

**Q6. Virtual Memory:**

Virtual memory is a memory management technique that is implemented by using both hardware and software. It makes the application think it has a contiguous and available storage space or address space. However, in fact, virtual memory usually is divided into several physical memory fragments, and some of them are stored on the external disk storage which can be used to exchange data when needed. Currently, most operating systems use virtual memory, like the Windows family's “virtual memory”, or the Linux “swap space”.

**Q7. Explain Deadlock and its characteristics:**

Deadlocks are a set of blocked processes each holding a resource and waiting to acquire a resource held by another process.