Data Structures and OOPS topic:

1. Why choose oops?
   * Developers often choose to use OOP in their Python programs because it makes code more reusable and makes it easier to work with larger programs.
   * OOP programs prevent you from repeating code because a class can be defined once and reused many times

2. What are the alternatives to oops?

**Functional programming:**

* Functional programming is a way of thinking about software construction by creating pure functions
* functions have two main properties. First, they always produce the same output for the same arguments irrespective of anything else.
* Secondly, they have no side-effects i.e. they do modify any argument or global variables or output something

**Advantages**:

* + Simple to understand
  + Making debugging and testing easier
  + readability of the code

**Disadavnatge:**

* + Low performance
  + Writing programs is a daunting task
  + Low readability of the code

**Procedural programming:**

* + In Procedure Oriented programming paradigms, series of computational steps are divided modules which means that the code is grouped in functions and
  + the code is serially executed step by step so basically, it combines the serial code to instruct a computer with each step to perform a certain task.

**Advantages:**

* + General-purpose programming
  + Code reusability

**Disadvantages:**

* + Data protection
  + Not suitable for real-world objects
  + Harder to write

**Logical programming:**

* + Logical Programming is a type of programming paradigm that uses logic circuits to control how facts and
  + rules about the problems within the system are represented or expressed.

**Advantages:**

* + Logic programming can be used to express knowledge in a way that does not depend on the implementation making programs more flexible and understandable.

**Disadvantages:**

* + The programs execution can be slow

3. What is the advantage and disadvantages for oops and datastructures?

**Advantage of OOPS:**

* + re-usability:“Write once and use it multiple times” you can achieve this by using class
  + OOP language allows to break the program into the bit-sized problems that can be solved easily
  + The principle of data hiding helps the programmer to build secure programs which cannot be invaded by the code in other parts of the program.
  + By using inheritance, we can eliminate redundant code and extend the use of existing classes
  + Inheritance is the good feature for data redundancy. If you need a same functionality in multiple class you can write a common class for the same functionality and inherit that class to sub class.

**Disadvantages of OOPS:**

* + Size: Object Oriented Programs are much larger than other programs.
  + Effort:Object Oriented Programs require a lot of work to create.
  + Speed:Object Oriented Programs are slower than other programs, because of their size.

**Advantages of DataStruture:**

* + Data structures allow information storage on hard disks.
  + allows the data use and processing on a software system.
  + Allows easier processing of data.
  + Using internet, we can access the data anytime from any connected machine (computer, laptop, tablet, phone, etc.)

**Disadvantages of DataStruture:**

* + Only advanced users can make changes to data structures
  + Any problem involving data structure will need an expert's help, i.e. basic users can not help themselves.

4. What is object with brief real time answer?

* + Object is the concept which represents the class.
  + It is the instance of class. With the help of a new operator we may easily create object of class and memory is created in the HEAP and the object is called an instance of class.

**REAL TIME EXAMPLE:**

If animal is the class then dog is the object, if human is the class then man is the object.

A dog has legs and eyes, then eyes is the variable in the technical concept,this is the property and

the dog may run or may walk, these are methods, and the same concept we used in OOPS concept

5. What is classes?

* + Class creates a user-defined data structure, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class.
  + A class is like a blueprint for an object. Classes are created by keyword class.

**Example:**

Class dog:

pass

6. What is encapsulation and advantage, disadvantage and real time example with brief answer?[practice a simple example]

**Encapsulation:** Binding

* + Bundling of data along with the method to operate on that data into a single unit
  + Used to hide the value of a structured data inside the class, preventing unauthorized parties

**Real time Example:** Bank balance

If we have a account in the bank. Some asking my balance means they wont tell to them, why because that is my data. They hide my details from the other customers.

**Advantage:**

* Protects an unwanted access by client
* We can hide the implementation details behind a public programming interface

**Disadvantage:**

* Encapsulation increase the length of the code

**Example:**

class Base:

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

self.\_a = 2

class Derived(Base):

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

Base.\_\_init\_\_(self)

print("Calling protected member of base class: ")

print(self.\_a)

obj1 = Derived()

obj2 = Base()

print(obj2.a)

7. What is polymorphism and advantage, disadvantage and real time example with brief answer?[practice a simple example]

**Polymorphism:**

* + poly means means
  + morphism means forms

Polymorphism means the condition of occurring in several different forms

**Example:** + operator

i)when ‘+’ operator used in number it perform addition

ii)the same ‘+’ operator is used in the string it performs concatenation

**Types:**

1. static (compile time)
2. dynamic(runtime)🡪

**Compile time:**

Refers to the time duration in which the programming code is converted into machine code, it occurs before runtime

**Runtime:**

When the programming is running when you start a program running in a computer it is runtime for that program

**Example:**

class Cat:

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

self.name = name

self.age = age

def info(self):

print(f"I am a cat. My name is {self.name}. I am {self.age} years old.")

def make\_sound(self):

print("Meow")

class Dog:

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

self.name = name

self.age = age

def info(self):

print(f"I am a dog. My name is {self.name}. I am {self.age} years old.")

def make\_sound(self):

print("Bark")

cat1 = Cat("Kitty", 2.5)

dog1 = Dog("Fluffy", 4)

for animal in (cat1, dog1):

animal.make\_sound()

animal.info()

animal.make\_sound()