Data Structures and OOPS topic:

1. **Why choose oops?**

A language like C has an amazing legacy in programming history, but writing software in a top-down language is complex. The more complex it gets, the greater the chance it will collapse. Meanwhile, writing a functional-style program in a language will offer stability and can be debugged easily.

Object-oriented programming is often the most natural and pragmatic approach, once you get the hang of it. OOP languages allow you to break down your software into bite-sized problems that you then can solve — one object at a time.

1. What are the alternatives to oops?

FP - Functional Programming is an extremely popular programming paradigm that has been around for a very long time and has, in more recent years, started becoming more and more prominent. FP favors immutability over mutability, recursion, and functions with no side effects.

1. What are the advantages and disadvantages for oops and data structures?

* It models the real world very well
* With OOP, programs are easy to understand and maintain
* OOP offers code reusability
* OOP facilitates the quick development of programs where parallel development of classes is possible
* With OOP, programs are easier to test, manage and debug

Data structure helps in efficient storage of data in the storage device. Data structure usage provides convenience while retrieving the data from storage device. Data structure provides effective and efficient processing of small as well as large amount of data.

1. What is object with brief real time answer?

An object in OOPS is nothing but a self-contained component which consists of methods and properties to make a particular type of data useful. For example color name, table, bag, barking. When you send a message to an object, you are asking the object to invoke or execute one of its methods as defined in the class.

1. What is class?

The building block of Object-Oriented programming is a Class. It is a user-defined data type, 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.

Example:

 Consider the Class of Cars. There may be many cars with different names and brand but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range etc. So here, Car is the class and wheels, speed limits, mileage are their properties.

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

Encapsulation basically means binding up of data in a single class.

class employee(object):

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

self.name = 1234

self.\_age = 1234

self.\_\_salary = 1234

object1 = employee()

print(object1.name)

print(object1.\_age)

print(object1.\_\_salary)

Advantages of Encapsulation  
  
Encapsulation **protects an object from unwanted access by clients**. Encapsulation allows access to a level without revealing the complex details below that level. It reduces human errors. Simplifies the maintenance of the application. Makes the application easier to understand

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

Polymorphism refers to the ability to exist in multiple forms. Multiple definitions can be given to a single interface.

**For example**, if you have a class named Vehicle, it can have a method named speed but you cannot define it because different vehicles have different speed. This method will be defined in the subclasses with different definitions for different vehicles.

Advantage:

It helps programmers reuse the code and classes once written, tested and implemented. They can be reused in many ways.

Disadvantage:

One of the disadvantages of polymorphism is that developers find it difficult to implement polymorphism in codes.Run time polymorphism can lead to the performance issue as machine needs to decide which method or variable to invoke so it basically degrades the performances as decisions are taken at run time.

1. What is data abstraction and advantage, disadvantage and real time example with brief answer?[practice a simple example]

Data abstraction is a very important feature of OOPs that allows displaying only the important information and hiding the implementation details. For example, while riding a bike, you know that if you raise the accelerator, the speed will increase, but you don’t know how it actually happens. This is as the implementation details are hidden from the rider.

Example:

real life example of Abstraction is ATM Machine; All are performing operations on the ATM machine like cash withdrawal, money transfer, retrieve mini-statement…etc. but we can't know internal details about ATM. Note: Data abstraction can be used to provide security for the data from the unauthorized methods.

Advantages of Abstraction:

- Abstraction makes code as readable and more simpler. Because complexity of code will be hidden.

- Abstraction makes code from longer to small , because abstraction doesn't show unnecessary things.

Disadvantage of Abstraction:

- A disadvantage is that abstract classes cannot be instantiated, but most of the time it is logical not to create a object of an abstract class

1. What is inheritance and types, advantages, disadvantages, difference or comparison between each types and real time example with brief answer?[practice a simple examples for each types]

INHERITANCE; Inheritance in simple terms means,,,getting some characteristics from the older ones. For example, when child takes birth it inherit some qualities from his parents. Same in the case of programming., the derived class takes some characteristics from the base class. The inheritance is beneficial for programming because with the help of this we can make our code as short as possible and there will be not any kind of redundancy.

TYPES:

Inheritance is the capability of one class to inherit capabilities or properties from another class in Java.

We inherit certain properties from the class ‘Human’ such as the ability to speak, breathe, eat, drink, etc

1. Single Inheritance

In single inheritance, there is a single child class that inherits properties from one parent class

2. Multilevel

In this type of inheritance, the child or derived class inherits the features of the superclass and simultaneously this child class acts as a superclass for another derived class.

3. Hierarchical Inheritance

In Hierarchical Inheritance, one class acts as a superclass (base class) for more than one subclass. More than one subclass can inherit the features of a base class.

4. Multiple Inheritance

In Multiple Inheritance, one child or subclass class can have more than one base class or superclass and inherit features from every parent class which it inherits.

5. Hybrid Inheritance

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

1. What is constructor, destructor and types , advantages, disadvantages?[practice a simple example for each types]

A constructor is a special type of method (function) which is used to initialize the instance members of the class.

In C++ or Java, the constructor has the same name as its class, but it treats constructor differently in Python. It is used to create an object.

The users call Destructor for destroying the object. In Python, developers might not need destructors as much it is needed in the C++ language. This is because Python has a garbage collector whose function is handling memory management automatically.

Destructors are called when an object gets destroyed. In Python, destructors are not needed as much needed in C++ because Python has a garbage collector that handles memory management automatically.

Advantage and disadvantage:

The constructor is used to allocate the memory if required and constructing the object of class whereas, a destructor is used to perform required clean-up when an object is destroyed. The destructor is called automatically by the compiler when an object gets destroyed.

1. What is access specifier and it’s types, advantages, disadvantages with real time explanation?

That determine the accessibility of methods, classes, etc in OOPs. These access specifiers allow the implementation of encapsulation. The most common access specifiers are public, private and protected.

• Private: access level is only within the class. It cannot be accessed from outside the class.

• Default: access level is only within the package. It cannot be accessed from outside the package. If you do not specify anyaccess level, it will be the default.

• Protected: access level is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.

• Public: access level is everywhere.

1. What is overloading, overriding and its type, advantages and disadvantages with real time example in brief answer and sample short programs for that.

When two or more methods in the same class have the same name but different parameters, it's called Overloading

When the method signature (name and parameters) are the same in the superclass and the child class, it's called Overriding.

Advantages: It allows reusability and saves time. Saves memory space. Faster execution .

Disadvantages: Function declarations that differ only in the return type cannot be overloaded

1. What is sorting and types(internal sorting and external sorting) and advantages.

A sorting algorithm is just a series of orders or instructions. In this, an array is an input, on which the sorting algorithm performs operations to give out a sorted array

**Internal sorting**: If the input data is such that it can be adjusted in the main memory at once, it is called internal sorting.

**External sorting**: If the input data is such that it cannot be adjusted in the memory entirely at once, it needs to be stored in a hard disk, floppy disk, or any other storage device. This is called external sorting.

1. Bubble sort – principle, advantages and disadvantages and with simple short example

It is the easiest and simplest of all the sorting algorithms. It works on the principle of repeatedly swapping adjacent elements in case they are not in the right order.

In simpler terms, if the input is to be sorted in ascending order, the bubble sort will first compare the first two elements in the array. In case the second one is smaller than the first, it will swap the two, and move on to the next element, and so on.

def bubbleSort(a):

    n = len(a)

    for i in range(n):

        for j in range(0, n-i-1):

            if a[j] > a[j+1] :

                a[j], a[j+1] = a[j+1], a[j]

a = [64, 34, 25, 12, 22, 11, 90]

bubbleSort(a)

print (“The sorted array is:”)

for i in range(len(a)):

    print (“%d” %a[i])

This algorithm has several advantages. It is simple to write, easy to understand and it only takes a few lines of code. The data is sorted in place so there is little memory overhead and, once sorted, the data is in memory, ready for processing. The major disadvantage is the amount of time it takes to sort.

1. Insertion sort– principle, advantages and disadvantages and with simple short example

The insertion sorts repeatedly scans the list of items, each time inserting the item in the unordered sequence into its correct position.The main advantage of the insertion sort is its simplicity. It also exhibits a good performance when dealing with a small list. The insertion sort is an in-place sorting algorithm so the space requirement is minimal. The disadvantage of the insertion sort is that it does not perform as well as other, better sorting algorithms.

def insertionSort(arr):

    for i in range(1, len(arr)):

        key = arr[i]

        j = i-1

        while j >= 0 and key < arr[j] :

                arr[j + 1] = arr[j]

                j -= 1

        arr[j + 1] = key

arr = [12, 11, 13, 5, 6]

insertionSort(arr)

for i in range(len(arr)):

    print ("% d" % arr[i])

1. Selection sort– principle, advantages and disadvantages and with simple short example

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array. The subarray which is already sorted.Remaining subarray which is unsorted.

def selectionSort( itemsList ):

n = len( itemsList )

for i in range( n - 1 ):

minValueIndex = i

for j in range( i + 1, n ):

if itemsList[j] < itemsList[minValueIndex] :

minValueIndex = j

if minValueIndex != i :

temp = itemsList[i]

itemsList[i] = itemsList[minValueIndex]

itemsList[minValueIndex] = temp

return itemsList

el = [21,6,9,33,3]

print(selectionSort(el))

**Advantages of Selection Sort**

It performs very well on small lists.

It is an in-place algorithm. It does not require a lot of space for sorting. Only one extra space is required for holding the temporal variable.

It performs well on items that have already been sorted

1. Quick sort– principle, advantages and disadvantages and with simple short example
2. Merge sort-principle, advantages and disadvantages and with simple short example
3. Shell sort– principle, advantages and disadvantages and with simple short example
4. Which is most powerful sorting algorithm with brief answer and why?

The time complexity of **Quicksort** is O(n log n) in the best case, O(n log n) in the average case, and O(n^2) in the worst case. But because it has the best performance in the average case for most inputs, Quicksort is generally considered the “fastest” sorting algorithm.

1. Types of complexity: time and space complexity

**Time complexity** is the amount of time taken by an algorithm to run, as a function of the length of the input. It measures the time taken to execute each statement of code in an algorithm.

***Space complexity****is the amount of memory used by the algorithm (including the input values to the algorithm) to execute and produce the result.*

1. Time complexity types and how to apply in coding with short examples

**Time complexity** is the amount of time taken by an algorithm to run, as a function of the length of the input. It measures the time taken to execute each statement of code in an algorithm

1.Constant time – O (1)

2. Linear time – O (n)

3.Logarithmic time – O (log n)

4.Quadratic time – O (n^2)

5.Cubic time – O (n^3)

Example:

Void fun(int n)

{

int i,j,k,c=0;----------🡪O(1)

for(i=0;i<n;i++)----🡪O(n)

for(j=1;j<n;j++)----🡪O(n)

for(k=1;k<n;k\*2)-🡪O(log n)

c++;

}

**Time complexity:n\*n\*n\*logn=O(n2logn)**

1. Linked list -short example with advantages and real time application

Advantage of Linked List

LinkedLists does not require contiguous blocks of memory and therefore **can** help reduce memory fragmentation.

LinkedLists support efficient removal of elements (dynamic arrays usually force a shift in all of the elements).

1. Types of linked list- Singly, Doubly, Circular, Doubly circular(**Just learn what is the principle or concept to just explain with real time example, advantages by comparing all its types and no need coding but if possible learn some short coding for its types**)

Singly Linked List:

The singly linked list is a data structure that contains two parts, i.e., one is the data part, and the other one is the address part, which contains the address of the next or the successor node. The address part in a node is also known as a **pointer**.

Doubly Linked List:

We can define the doubly linked list as a linear data structure with three parts: the data part and the other two address part. In other words, a doubly linked list is a list that has three parts in a single node, includes one data part, a pointer to its previous node, and a pointer to the next node.

Circular Linked list:

A circular linked list is a variation of a singly linked list. The only difference between the ***singly linked list*** and a ***circular linked*** list is that the last node does not point to any node in a singly linked list, so its link part contains a NULL value. On the other hand, the circular linked list is a list in which the last node connects to the first node, so the link part of the last node holds the first node's address. The circular linked list has no starting and ending node. We can traverse in any direction, i.e., either backward or forward.

Doubly circular linked list:

The doubly circular linked list has the features of both the **circular linked list** and **doubly linked list**. Similar to circular linked list the root node and the last node will be connected by a pointer and it is done using double linked list.

1. Stack-simple short code with advantages with real time application

Stack is **a linear data structure which follows a particular order in which the operations are performed**. The order may be LIFO(Last In First Out) or FILO(First In Last Out). There are many real-life examples of a stack. Consider an example of plates stacked over one another in the canteen.

1. Reverse linked list with short program
2. Hash table- with short coding, advantage, disadvantage with real time examples

In computing, a hash table (hash map) is **a data structure that implements an associative array abstract data type**, a structure that can map keys to values. A hash table uses a hash function to compute an index, also called a hash code, into an array of buckets or slots, from which the desired value can be found.

1. Difference between tree and graph with real time example, advantage and practice short code for simple example
   * Graph can have any number of nodes in it, while in tree can also have any number of child node but in case of binary tree a node can have at most of 2 child nodes
   * There is no unique node in graph while there is a unique node called “root” in tree
   * Graphs can form cyclic pattern while tree cannot
   * Applications of graphs are finding the shortest path in routing , for tree File Folders in the system follows tree structure
2. Binary tree vs B tree (not expected coding but only in concept wise learn)

* In B tree the nodes are self balanced while the binary tree is not
* B tree can have N number of child nodes while the binary tree can have at most 2 child nodes
* All the leaf nodes should be at the same level in B tree and above the leaf nodes there should not be any empty tree, while there is no such condition in Binary tree
* Inserting elements or keys in B tree is more complicated than inserting in Binary tree

1. Binary search tree with real time example, advantage and practice short code for simple example

Binary search is used when there is primarily a criterion of efficiency. It involves working on the already ordered data, which is sorted either in ascending or in descending order. To begin with, the middle element of the array is found out, and the search begins from there. The array is searched in two parts based on the search value being higher or lower than the middle element. It is key to know the order of the arrangement to help search the value accordingly.

1. Linear search and Binary search with real time example, advantage and practice short code for simple example

A linear search scans one item at a time, without jumping to any item .The worst case complexity is  O(n), sometimes known an O(n) search.Time taken to search elements keep increasing as the number of elements are increased.

A binary search however, cut down your search to half as soon as you find middle of a sorted list. The middle element is looked to check if it is greater than or less than the value to be searched. Accordingly, search is done to either half of the given list

1. Graph and its types(BFS and DFS algorithm) with short code as example, advantages and real time application

**1. Weighted Graph**

Graphs whose edges or paths have values. All the values seen associated with the edges are called weights. Edges value can represent weight/cost/length.

**2. Unweighted Graph**

Where there is no value or weight associated with the edge. By default, all the graphs are unweighted unless there is a value associated.

**3. Undirected Graph**

Where a set of objects are connected, and all the edges are bidirectional.

**4. Directed Graph**

Also called a digraph, where a set of objects (N, E) are connected, and all the edges are directed from one node to another. The above image showcases the directed graph.

1. Recursion

Recursion is the process of defining something in terms of itself.A physical world example would be to place two parallel mirrors facing each other. Any object in between them would be reflected recursively.

def factorial(x):

if x == 1:

return 1

else:

return (x \* factorial(x-1))

num = 3

print("The factorial of", num, "is", factorial(num))