Specialisation theory questions

1. How does Object Oriented Programming differ from Process Oriented Programming?

Object orientated programming:

1. Program is divided into classes and objects/instances where you are able to reuse the objects in other programmes
2. Uses encapsulation such as access modifiers private, public and protected
3. Due to abstraction able to hide data so it is more secure
4. Programs are easy to modify as new data objects can be created easily from already made objects
5. Easy to add new data and functions
6. Uses a bottom-up approach as after defining the class and methods you create the object. The execution takes place in a bottom-up approach where the object is initialised and then it looks for the declared class and functions.
7. Overloading is possible in the form of function overloading and operator overloading
8. Uses mutable data
9. Not able to be used for parallel programming
10. Conditional statements and loops are used to help flow control
11. Uses loop to iterate lists, sets, dictionaries
12. OOPH is slower than procedural and use up a larger amount of memory

Procedural orientated programming:

1. Uses a sequence of instructions to break the task into a collection of variables where each step is done in a systematic order
2. Procedural programming separates code from the processed data so the data and procedures are treated separately
3. It divides the program into procedures or functions to tell you how things should be carried out whilst the data structures are built separately which maintain the processed data
4. Data moves freely from function to function
5. No access modifiers are used hence is less secure than OOP
6. Not as simple to add new data and functions without having to go through the whole program
7. Top- down approach where you implement the most basic modules first and then work towards those that provide a specific functionality hence breaking bigger problems into smaller ones. You start coding at the top of the file and work your way down to the bottom.
8. Not able to overload functions
9. Uses immutable data
10. Able to be used for parallel programming
11. Functional calls are used to help flow control
12. Uses recursion to iterate lists, sets, dictionaries
13. What's polymorphism in OOP? **(See Python file)** -> method overriding/method overloading
14. It allows a variable, object or function to have many different forms
15. It is the ability of a class instance to behave as if it were the instance of another class
16. Runtime polymorphism involves method overriding which allows you to change the code that the method inherits in the child class from what it originally was in the base class. This allows you to use the method in the parent class without duplicating and you can adapt it to how you want. If the object/instance made is based on the child class and then then method is run. Although the methods in the parent and child class have the same name it will use the adapted method in the child class as that is the class it is derived from and them same would be vice versa.
17. The advantages of method overriding is that you can adapt a method and use the parameters without changing it in the parent class.
18. Compile time polymorphism involves method overloading where it is the ability of a function to act differently based on the number of parameters that have been passed into it for example zero or more than one. Therefore, instead of writing many different methods that differ by the number of parameters being used you can write one method and simply overload it. As a result, it helps to make the code simpler and improves its transparency as we are calling the same method but in different ways.
19. Advantage of overloading: allow programmers to reuse code once they have been written, tested, and used
20. Advantage of overloading: A variable can be used to stored variables that have different data types such as float, double, long, int.
21. Advantage of polymorphism: Able to make powerful complex abstractions
22. Advantage of polymorphism: Makes it is easier to debug code
23. What's inheritance in OOP? **(See Python file)**
24. A child class can inherit attributes or methods from the parent class by extending from it.
25. As well as being able to inherit the properties from a parent class the child class can have its own properties and methods.
26. To inherit from the parent class in parenthesis for the child class you simple insert the name of the parent class.
27. You can have a single form of inheritance as well as multiple inheritance where you are inheriting from many classes instead of just one and therefore having access to their methods and variables.
28. You can also have hierarchical inheritance where multiple child classes are inheriting from the parent class. Child classes can also inherit from other child classes that originally inherit from the parent class which displays a type of multilevel inheritance.
29. Hybrid inheritance is when you can put together more than one type of inheritance such as hierarchical and multiple.
30. Advantages of Inheritance means we can reuse the code, improve its readability, easily manage it as it is divided into the parent and child classes, allows functions to be modified in the child class.
31. Can use the super().method\_name\_parent\_class() allows you to have access to the contents of the method in the parent class without rewriting the code in the child class. Or use parent\_class\_name.init() which allows you to use the init function from the parent class in the child class and have access to all code.
32. If you had to make a program that could vote for the top three funniest people in the office, how would you do that? How would you make it possible to vote on those people? **(See Python file)**
33. What's the software development cycle (SDLC)?
34. This process is used to help produce high quality software whilst trying to minimise the use of fundamental resources such as cost and production time.
35. Requirement analysis involves communicating with the stakeholders e.g., product manager/owner, customers, salespeople, operators’ developers and testers to do a SWOT analysis and creating user stories. The stakeholders will look at the terms of the agreement will sign the deal and accept the project.
36. Planning is where you determine the cost, time, and energy for implementing the requirements as well as being able to determine how they will carry this project out whilst minimising risk. You can consider factors such as the user registration, login, logout, and dashboard landing page… Once the requirements are outlined the product owner will create tickets in a product management system using e.g. Jira.
37. Software design e.g. architectural design such as the database and components and all the software requirements are actually implemented into a design specification where the rest of the team and stakeholders will review. This is very important as this needs to be solidifies before any changes can be implemented. The design team will also consider factors such as the layout e.g. responsive web design and colour scheme, mobile support and programming languages to use.
38. Software development is where the implementation of the plan takes place once it has been agreed to by developers. It is fundamental to have rules about the proper instructions about practises of the code to help formulate a clear, organised, and readable structure. Testers are very valuable as they will write test cases that they imagine the usability of the application and check for any faults early on.
39. Testing where you identify and fix bugs by implementing a bug tracking system in order to make sure the code has met the required guidelines e.g. if the browser session does not clear when logging out of the homepage and if for personal banking details are on their it could be a problem.
40. Deployment is where you want to deploy the software so it can officially be in us, however more testing may take place here during the maintenance part before the application goes live to real users. If millions of users are using this application, then need to find a way to maintain the servers in the environment as we do not want the system to be brought down. As a result, will endure that larger databases, servers and faster computer will be used. If more bugs are encountered then we use production support where the users may email problems, get the bug fixed and do another deployment to production.
41. The SDLC is repetitive, and you must ensure that quality code is being produced at every cycle where new requirements and features will be added
42. Advantages of the SDLC: Can provide the greatest level of management control, provides a framework for planning, decreases the chance of certain risks taking place, helps to improve the clarity of the project and overall, the relationship with the client.
43. What's the difference between agile and waterfall?
44. Waterfall model is a type of SDLC model with a sequential approach where one phase is complete and waterfalls into the next phase.
45. Agile model is a type of SDLC model that separates the product into cycles/builds so there is a continuous iteration of development and testing which delivers a working product very quickly.
46. Agile uses an adaptive software development method whereas Waterfall uses a predictive approach where detailed planning, requirement analysis and planning take place at the very beginning.
47. Agile uses an adaptive method where there is not much detailed planning and the future tasks being planned are focused on the features that will be explicitly used. Customer interaction is very important as the agile teams work closely to one another so it important that they regularly communicate with one another.
48. Agile is more flexible in comparison to the waterfall methodology which is more structured hence agile is better when adapting to a change in situations and hence provides more scope for changing environments.
49. Agile completes its tests simultaneously and the test plan is reviewed after each sprint whereas Waterfall does it after the software development phase.
50. Agile ensures that the client is constantly being involved during each stage of the process and the lifecycle is separated into sprints whereas Waterfall uses specific phases.
51. Agile can take place multiple different projects whereas Waterfall software development usually takes place on one project.
52. Within agile methodology the stages may take place more than once whereas in Waterfall once they are completed, they move on.
53. Agile has a product mindset where the product must satisfy the customer demands whereas the waterfall must satisfy the project targets.
54. During agile the product owner will revaluate the requirements roughly every day during the project but with Waterfall the business analysts will only prepare the requirements before the beginning of the project.
55. The project manager has an essential role during the Waterfall whereas they do not for Agile as the project is managed by the whole team. By having team members that can have multiple roles they are able to work more quickly.
56. What is a reduced function used for? **(See Python file)**
57. Reduce must be imported from functools
58. The reduce function allows you to concisely reduce the list instead of using a for loop to go through it
59. The reduce function works its way from left to right and where the function takes two arguments performs the function operand on them. The first parameter is now the result of these two arguments, and you move through the next element in the list to perform the operand on with the result.
60. This is completed in a recursive manner as the reduce continues to call the function/same operation (callable object) on the next element in the list until it has been completely iterated through and an item is returned.
61. In conjunction with the reduce function lambda is often used to help make the code more concise.
62. The Lambda function can take any number of arguments but only one operation.
63. Lambda is an anonymous function as it does not have a name.
64. In Python 3 the reduce() works with 3 parameters as well as 2.
65. How does merge sort work? **(See Python file)**
66. Merge sort is sorting algorithm that has a top-down approach
67. Merge sort takes place recursively as you divide the pointers of the unsorted array by two the LHS and RHS.
68. You have a function to recursively break the array into two or more subarrays and keep going until you reach a single element/cannot be further divided which for the LHS is when the left == middle and for the RHS the middle == right.
69. During the recursive steps to get to a single entity in the list you check the left branch / left side of the element pointer and the right branch / right side of the element pointer before starting the merge.
70. The merge function will then sort the individual elements into pairs and then will compare with the first number within each pair. If the element from the LHS > element from the RHS then the element from the RHS will be appended into the sorted list first and vice versa. If there is another number on either side, then these elements will also be compared. So, the individual items are evaluated by comparing the values and merging them within temporary arrays.
71. The list is doubled/elements which become subarrays keep pairing up until you reach the end of the list and produce a sorted list.
72. You keep going up the recursion stack where the smaller arrays are merged into a larger one whilst the elements are inserted in order. One final merge is required to have the sorted list and in the example a for loop is used to convert sorted array to the initial array.
73. The pros of merge sort it that it works more efficiently for larger lists as it does not go through the list several times and the run times are similar within the stages. It is also useful for sorting linked lists.
74. The cons of merge sort are that it requires more memory to store two lists and even if the whole list is sorted is goes through it again hence it is a bit slower than other algorithms when working on smaller tasks.
75. The time complexity of merge sort is O(nlog(n)) which is faster than bubble, selection and insertion sort which is O(n^2). At every level the number of operations performed is equivalent to the length of the array. So, the recursive tree generated is a balanced binary tree where the height/levels of the tree is log(n) where we perform n operations on each level hence number of levels x number of operation per level is O(nlog(n)).
76. Generators - Generator functions allow you to declare a function that behaves like an iterator, i.e. it can be used in a for loop. What is the use case? **(See Python file)**
77. An iterator is an object that allows you to go through all go the values in a list, tuple, dictionary using the functions \_\_inter\_\_() and \_\_next\_\_() where you do not need to use a for loop to go through the elements.
78. As using an iterator can be a bit time consuming generators can be used to create iterators. As it is a function that will return the iterator which will allow you to go through each element one at a time.
79. A generator is made very simply where a yield statement is used instead of a return statement which instead of completing the function it pauses it temporarily pauses it until where the variables and states are saved and can be continued from that point when next used.
80. A generator can also have multiple yield statements where a generator is an iterator but not vice versa. As functions like \_\_inter\_\_() and \_\_next\_\_() don’t need to be used as the generator already performs these under the hood.
81. You should use generators because they are shorter and are able to return the items in an object one by one in a lazy manner where it does not return something until the user specifically asks for it.
82. Generators are great when used to calculate large sets of data for example fixed size containers or where the stream is an unknown size as well as in cases if you don’t know if you need results or will use all your results at the same time hence can create an infinite list.
83. Generators can be used instead of call-back functions and are able to generate data on demand
84. A stack of generators can create a processing pipeline which will in turn help with the readability and precision of the code.
85. If the data is cheap and storing it is expensive because the data set is extremely large it is better to use a generator
86. Generators can help with two way data transfer hence they can support send() and throw() methods.
87. Generators help to improve the memory performance/is more efficient as it does not hold the entire records in memory and one item is yielded at a time whereas iterators will load the entire records in memory and then go through each process.
88. Generators can encapsulate code as they are able to give you the values when it is called within your code.
89. Generators can be used for expensive information processing operations as you able to efficiently get data when needed and do not need to store them for long periods of time.
90. Decorators - A page for useful (or potentially abusive?) decorator ideas. What is the return type of the decorator?

The decorator requires a callable and then a different callable is returned hence the return type of the decorator is a callable.

Useful decorators:

Object orientated programming:

1. @unittest.mock.patch.object -> patch the main member with an object that is mocked
2. @abc.abstractmethod -> abstract methods
3. @unittest.mock.patch.multiple -> one call that uses many patches
4. @asyncio.coroutine -> helps to mark coroutines that made by a generator
5. @unittest.mock.patch.dict -> patches the dictionary
6. @contextlib.contextmanager -> set up a function that returns an object with context managers
7. @types.coroutine ->returns a coroutine based on a generator
8. @atexit.register -> provides an interface to register the functions
9. @static.method-> gives you a static method in a class
10. @classmethod-> gives you a method in a class
11. @functools.cached\_property
12. @property -> a way to get getters and setters
13. @functools.lru\_cache -> uses a cache dictionary
14. @functools.wraps -> updates the wrapper function to look like the wrapped function