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

OOP divides a program into smaller objects whereas POP divides a program into smaller functions

OOP is a process for modelling real world objects and relationships between different entities. It sees code as objects rather than a set of procedures. These objects are used for encapsulation, so any functions needed for the object to work are within a ‘class’. It relies heavily on data and the data in each object is controlled on its own. It is easy to add new data and functions. It divides uses the concepts of objects and classes. It uses a bottom-up approach. Using Encapsulation, you can hide data. OOP is better for solving big problems.

Process oriented programming is how most beginners are taught how to code. POP is functional; looking to perform a task. This can be long-winded and what programmers do instead of having multiple functions is that they generalise the functions that they already have (which is what is done in OOP because the existing code can be reused and is more concise). It uses a top-down approach. It has no control over data, every function has its own data. It can be difficult to expand data and functions. Data cannot be hidden and is accessible globally. This type of programming is not good for solving big problems. Mainly used for medium-sized applications. All the functions can access the global data involved. This can make its use unsafe because the data is not secure.

Overall, it is argued by many that OOP will give a better way to organise a programmer’s code. OOP is easier to troubleshoot because of encapsulation as objects are self-contained and do not interfere with different parts of the code. And inheritance in OOP means that code can be reused.

1. **What's polymorphism in OOP?**

The literal definition of ‘polymorphism’ is to exist in different forms. In programming this is when you use a single type of entity to represent different types in different scenarios.

Polymorphism is essential for every OOP language. In OOP is lets an object take many different forms at different instances. So, the same function name can be used for different types.

A child class inherits all the methods from the parent class. But sometimes a method inherited from the parent class doesn’t fit into the child class. So, in this case the method needs to be re-implemented in the child class. Polymorphism allows you to define methods in the child class with the same name as defined in their parent class.

Here are some examples:

* Polymorphism in the + (plus) operator

Polymorphism exists with the + operator because it can work with numbers (for example the integer data type) by adding them, or it can work on strings (so the string data type) by concatenating them. So depending on the data type the + operator acts almost like a different operator in each case.

* There is also the len() function.

Polymorphism exists with len() because for list types len() can be used to output the number of items in a list. But for the string data type it will return the number of letters in the string.

These are some simple examples. In fact, polymorphism extends to class methods too. This means that different classes can contain methods with the same name.

1. **What's inheritance in OOP?**

Child classes in Python inherit methods and attributes from the parent class. So inheritance is what allows us to define this child class that takes all the functionality from the parent class and then add more functionality. In fact, methods and attributes can be redefined specifically for the child class.

Inheritance looks like in in python:

Graphical user interface, text

Description automatically generated

Essentially, the child class calls the parent class within it allowing it to inherit as follows:

Both of these give the same outputs.

class Parent:  
 def \_\_init\_\_(self, name, age):  
 self.name = name  
 self.age = age  
  
 def printname(self):  
 print(self.name, self.age)  
  
  
Abe\_as\_father = Parent("Abe", 40)  
Abe\_as\_father.printname()  
  
class Child(Parent):  
 pass  
  
Abe\_as\_son = Parent("Abe", 40)  
Abe\_as\_son.printname()

1. **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?**

I will answer this question using pseudocode:

First input: each person in the office’s name

Send these to a database and store them

Now everyone in the office should be able to vote on who they think is the funniest. So, I will take three inputs form each person. 1st input is the who they thin is the funniest person – they will get a value of 3 added next to their name on the database. Second input is who they think is second funnies - they will get a value of 2 added next to their name on the database. And third input is who they think is the third funniest - they will get a value of 1 added next to their name on the database.

These entries will be sent to the database and stored. The database will update depending on these inputs.

I will then output the information from the database as a list is an inerrable over which I can loop. I will loop over these entries to see who got what score and in which order. The person with the highest total will thus be the funniest in the office, etc.

I will call upon the database to extract the top three people – i.e., the three with the highest scores and this will solve the question.

In order to actually work on this ‘app’ for finding the three funniest people in the office I would work using the Agile approach as it is good for working on small scale projects.

1. **What's the software development cycle? SDLC (software development life cycle)**

This is a process with clearly defined steps that software developers use to design, develop and test good quality software. This process is designed to meet the customer’s requirements, deadlines, and cost restrictions. The steps within SDLC are as follows:

* Requirement analysis – this is where the current problems are considered. This requires getting input from all the stakeholders such as the programming team, the customers, and the salespeople for example.
* Planning – this is the stage where what is required from the stakeholders is considered alongside the available budgets and resources. The risks involved are also considered allowing the team to decide on how best to proceed with bringing the project to life.
* Architecture design – the plan is turned into a design. This can be reviewed by all stakeholders and their feedback is considered.
* Software development – this is where the programming team brings the project to life. The actual project is built. Developers must follow the guidelines laid out by their organisation.
* Testing – this is where any defects within the project are noted and fixed. Things are then retested until the project fulfils its quality requirements. It is also an opportunity to check that the code does what it is required to do.
* Deployment – this is where the stakeholders can start using the software before releasing it to the market for users, so that any mistakes that have not been spotted up until this point can be pinpointed. After this the product is officially released into the desired market. This may be done in stages with a small collection of users being able to use it before it is released into the mainstream.

SDLC is considered as ‘best practice’ for programmers and it is argued that following this method of work will allow for the best results from a program, as well as the most efficient work process for the development team.

1. **What's the difference between agile and waterfall?**

These are both ways for software development teams to complete projects. Software development teams decide at the beginning of a project which of these methods they will adopt.

Agile uses cyclic and collaborative processes. To complete a project, it will move through a series of cycles. This means that tasks are completed in increments, and each involves all the SDLC phases. This means that after each increment an updated product is delivered. Once all the iterations are completed the final product is ready. Agile is better for completing many small projects and changes to requirements are allowed at any time.

Waterfall uses sequencies methods so it is generally linear but can allow for collaborative work. Requirements are prepared once at the start of the process. The tasks from each stage are reviewed and verified before moving onto the next stage. It is argued that this is a simpler and more traditional process. Waterfall is better when completing one single project, and changes to requirements are not generally welcome.

1. **What is a reduce() function used for?**

The reduce() function takes in a function and a sequence, so we have reduce(fun,seq). You must pass a function into its argument and what reduce() does is that it applies this function to all of the elements of the list of the sequence which is passed along. The reduce() function ultimately returns a single value and this is calculated in a specific way. This is because there is a maths technique known as ‘reduction’. The reduce() function in Python implements this mathematical technique. This is does in the following steps:

* A function and a sequence are input into reduce()
* The function is called with the first two terms in the sequence
* The appropriate result is returned
* The function is called again, this time with the result that we have just obtained and what would be the ’third’ term of the sequence
* The appropriate result is returned
* This loop continues
* It stops when all the terms of the sequence have been ‘used up’

Here is an example:

Graphical user interface, text

Description automatically generated

1. **How does merge sort work?**

This is a solving algorithm. It sorts values in any traversable data structure (e.g. a list). It works in the following way (using the example of a list of 8 items):

* First it splits the input list into two halves (4 and 4)
* Then it halves the halves (2,2 and 2,2)
* It sorts each of the quarters separately (orders the first 2, then the second 2, then the third 2 and finally the fourth 2)
* Then it sorts the two halves – putting each of the quarters in the correct order (so it sorts the first 2,2 and then the second 2,2 – leaving two lists of ordered 4s)
* Finally, it merges the two sorted halves together (now it just needs to put the two sets of 4 in the correct order – leaving an ordered list of 8 items)

This same process can be repeated for any length array. Even where there is an odd number of entries in the array the process still works (with one ‘half’ having one less item in it). It is often referred to as a ‘divide-and-conquer’ algorithm.

1. **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?**

By ‘use case’ I will assume this to mean ways in which we can use generators. The uses of generators stem from the use of iterators because generators allow programmers to make an iterator in a fast and easy way (the type of iterator they produce is often referred to as a ‘lazy’ iterator). An iterator is any object that can be looped upon.

Iterators allow programmers to save memory space, so generators do too.

Generators introduce the ‘yield’ statement (rather than a ‘return’ statement) into Python, this statement returns a value and also saves the state of the function. Therefore, when the function is called later, execution of it continues from where it left off, with the same values that it had prior to yielding. In fact, if the ‘return’ keyword is used in a generator this forces the generator function to terminate. This is because ‘yield’ return a values and pauses execution while internal states are maintained, but ‘return’ returns a value and then terminates the function’s execution.

Generators are functions which do not return single values. Instead, they return iterator objects with a sequence of values.

1. **Decorators - A page for useful (or potentially abusive?) decorator ideas. What is the return type of the decorator?**

In python decorators are used to modify a function or class’ behaviour – without directly changing any of the source code of the original function. The decorator itself is a function that returns another function. Essentially decorators modify a function, but this modification is non-permanent.

Decorators are useful when you have different functions with common features/ behaviours.

Decorators were designed to make code easier to read and write. Thus, they focus on the software engineers who *read* and *write* code.

Unfortunately, decorators can cause many issues in Python. This can be due to the fact that the code written is dependent on the software engineer in charge of calling that code and decorators do not consider this as they do not focus on the software engineer calling the source code. Although decorators are useful for counting function calls.

A decorator takes in a function as its input, but its return type does not have to be a function.