THEORY QUESTIONS ASSIGNMENT

Software Stream

Maximum score: 100

KEY NOTES

- This assignment to be completed at student's own pace and submitted before given deadline.
- There are 10 questions in total and each question is marked on a scale 1 to 10. The maximum possible grade for this assignment is 100 points.
- Students are welcome to use any online or written resources to answer these questions.
- The answers need to be explained clearly and illustrated with relevant examples where necessary. Your examples can include code snippets, diagrams or any other evidence-based representation of your answer.

Theory questions	**10 point each**

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

   In procedural programming, the program is divided into small parts called functions. In object-oriented programming, the program is divided into small pieces called objects. Procedural programming follows a top-down approach. Object-oriented programming follows a bottom-up approach

2. What's polymorphism in OOP?

   A polymorphic object is an object that is capable of taking on multiple forms to represent different types in different scenarios.

3. What's inheritance in OOP?

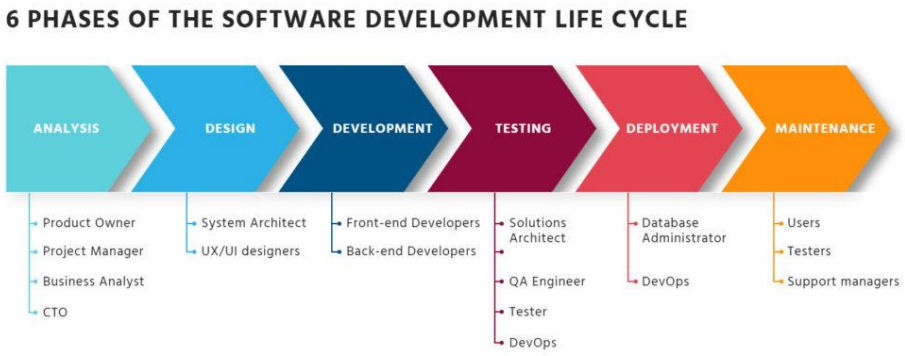
   Inheritance is the process by which one class takes on (inherit) the attributes and methods of another.

4. 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 would create a class named Person with the attributes has_voted and votes_received then I'd create a method vote. For example: if you have Person A and B. A would vote on B, first the method vote would check if has_voted is True, if not B. votes_received would be incremented by 1 and A.has_voted would be set to True. The method vote needs to also check if the person is voting for itself.

5. What's the software development cycle?

1.Plan
2.Requirements
3.Design
4.Implementation
5. Test&integrate
6.Deploy

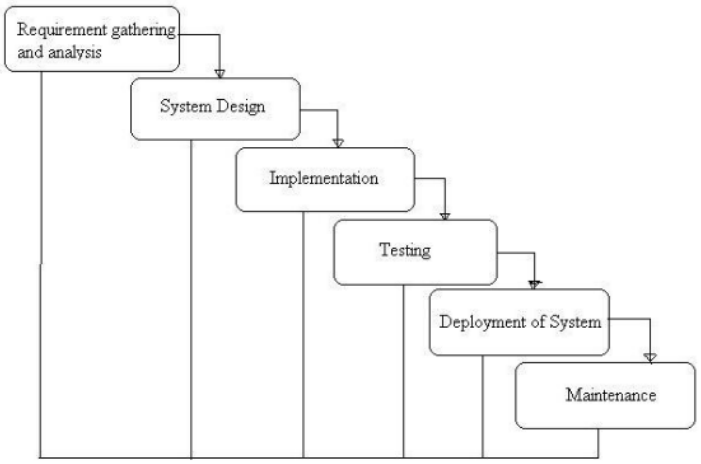


6. What's the difference between agile and waterfall?

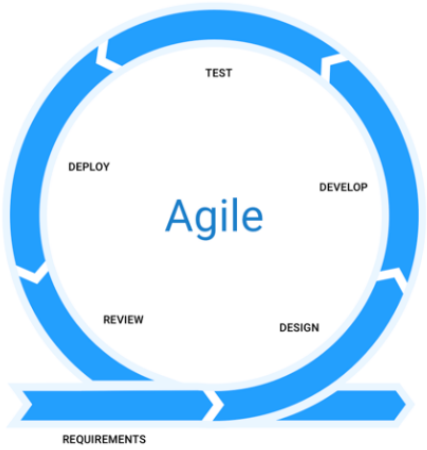
The main difference is that Waterfall is a linear system of working that requires the team to complete each project phase before moving on to the next one, while Agile encourages the team to work simultaneously on different stages of the project. Agile was developed as a flexible method that welcomes incorporating changes of direction even late in the process.

Waterfall methodology is a linear form of project management ideal for projects where the end product is clearly established from the beginning of the project. The expectations for the project and the deliverables of each stage are clear and are required in order to progress to the next phase.

Waterfall:



Agile:



7. What is a reduced function used for?

   The reduce() function accepts a function and a sequence and returns a single value calculated as follows:

   1. Initially, the function is called with the first two items from the sequence, and the result is returned.
   2. The function is then called again with the result obtained in step 1 and the next value in the sequence. This process keeps repeating until there are items in the sequence.

8. How does merge sort work

   Merge sort is a sorting algorithm and it works by splitting the input list into two halves, repeating the process on those halves, and finally merging the two sorted halves together.

9. 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?

   We could write a movie matching program to call an external API. We could perform the calls to collect all the results and then match them with a given parameter, for example, genre. Using a generator function, we can match the results while fetching them, making the program more memory efficient and faster.

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

    A decorator return type is a function.

    A decorator takes another function and extends the behaviour by adding a functionality on top of the function without modifying it.. You can also add multiples decorators on top of an existing decorator.

    Example:

```
import time


def timer(func):
        def wrapper(*args, **kwargs)
                start = time.time()
                result  = func()
                total = time.time() - start
                print("Time:", total)
                return result

        return wrapper

@timer
def test():
        for n  in range(20000):
                pass

test()
```