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 (OOP) differ from Process Oriented Programming (POP)?

Object Oriented Programming and Process Oriented Programming are both programming paradigms that are different in several ways. Firstly, in OOP, the program is separated into small parts known as objects and it follows a bottom up approach whilst in POP it is divided into small parts called functions and follows a top down approach. In POP there is no access specifier meanwhile in OOP there are different access specifiers such as protected, private, public, etc.

Furthermore, in Object Oriented Programming it is easy to add more functions and data, with it also providing data hiding features which makes it secure whilst in Process Oriented Programming these features do not exist, making it less secure. It is also more difficult in POP to add new data and functions than in OOP. One of the downsides of Object Oriented

Programming is that overloading can occur whilst this is not an issue with Process Oriented Programming. Another way in which they differ from

each other is that OOP is based on the real world and uses classes and objects to create models whilst POP follows a methodized approach that divides a task into variables and routines through a method that is performed in a systematic way which allows a computer to understand the task. Examples of OOP are Ruby, Scala, Python, C++, Java, etc. whilst examples of POP are C., Pascal, Basic, etc.

2. What's polymorphism in OOP?

The term polymorphism is one of the most important features of Object Oriented Programming and refers to the ability of an object, function, or variable to take on more than one form, this is useful to developers as it allows them to program in general rather than specifically. An example of polymorphism in real life is when an individual has more than one role in their life, such as they are a sister, a university graduate, a student programmer, etc. all at the same time. The most common way of polymorphism that takes place within OOP is when a child class object refers to its parent class. Two of the most popular types of polymorphism are runtime and compile time polymorphism.

Runtime polymorphism, also called Dynamic Method Dispatch, is a process which solves a function call to the overridden method is resolved at runtime through method overriding. Method overriding happens when a derived class has a definition for one of the member function of the base class, causing the base function to be overridden.

Compile-time polymorphism, commonly known as static polymorphism, occurs through function or operator overloading. Whilst Java supports polymorphism, it does not support operator overloading. In this case, method overloading happens when numerous functions hold the same name but don't have the same parameters, which leads them to be considered overloaded. This can also happen within functions when the number of arguments or the type of arguments within a function changes.

3. What's inheritance in OOP?

Inheritance in Object Oriented Programming is the ability of a class to inherit or derive properties from another class, with a child class inheriting traits from its parent class. This is beneficial as it allows code to be reused, saving the programmer time from rewriting the same code numerous times. Additionally, it permits the addition of more features to a class without changing it. It is also transitive, meaning that if for example class 2 inherits from class 1 then all the subclasses from class 2 will also inherit from class 1 automatically. There are different types of inheritance:

Single inheritance is when a child class inherits from a single parent class, indicating why it is called single inheritance as shown in the following example:

```
class Animal(object):

    def __init__(self, name, species):
        self.name = name
        self.species = species

    def display(self):
        print(self.name)
        print(self.species)




class Dog(Animal):
    def __init__(self, name, species, breed, cuteness):
        self.breed = breed
        self.cuteness = cuteness

        Animal.__init__(self, name, species)


a = Dog('Cyril', 'Canis lupus familiaris', 'corgi', 'the_goodest_of_bois')

a.display()
```

Multiple inheritance on the other hand, is where a child class inherits from more than one parent class. This is supported by Python but isn't supported by Java. The different parent classes are specified in a comma separated list within brackets.

Multilevel inheritance is where there is a three or more level inheritance which involves a parent, a child and a grandchild within the inheritance. Hybrid inheritance is another type of inheritance which involves combining more than one form of inheritance whilst Hierarchical inheritance is when within an inheritance there are multiple derived classes which are created from a single base.

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?**

First, I would do it so that you can input the name of the candidate that you wish to vote for. Following this, the name would automatically be added to an array of names which would be the list of voted that I would apply the iterator method of counter:

```
from collections import Counter
```

to convert the list into a dictionary with the candidate names being the key and the frequency of the occurrence of the names within the list as the value (the counts).

```
vote_count = Counter(votes)
```

I would then count the votes per person and store them in a dictionary to then find the three largest number of votes in terms of count. Once this has happened then I would search for the people that are stored within the list that have these amounts and sort the list so that it prints the three names that have the top votes.

5. **What's the software development cycle?**

The software development cycle, commonly known as the Software Development Life Cycle (SDLC) is the name of the complicated process of developing a piece of software from start to finish, with it being considered a cycle since even when it is finished it still needs to be maintained or updated. This cycle includes multiple stages that developers follow in order to obtain a quality end product to satisfy the requirements of the project. The stages of this process are the following:

- Planning and requirement analysis: This is where a requirement analysis is gathered from the customers/sales department by the developers. The data collected from the requirement analysis then allows the developers to design the basic project.
- Defining requirements: Once the analysis has been performed alongside the planning, then the requirements of the target

software are determined through using the Software Requirement Specification (SRS), which is a document that states all the things that will need to be created and defined throughout the duration of the project cycle, after getting the approval of the stakeholders and customers

- Designing architecture: Through using the Software Requirement Specification as a reference, the software designers then go ahead and design the best architecture for their specific software, multiple designs of product architecture are found within the Design Document Specification which is examined by both the stakeholders and the market analysts and then after the examination the most practical design is chosen to be used for the development of the software.
- Developing product: This is where the main body of development of the software begins. In this stage the developers use specific programming code which is chosen from the design in the DDS. It is common for programming tools such as interpreters, compilers, and debuggers to be used at this stage of development with programming languages such as Python, C++ and Java being used as specified through the software regulations.
- Product testing and development: This stage occurs once the product development has finished and involves testing the software to try and find any errors to fix them before delivering the finished product and to ensure that the software executes smoothly. Having said this, some form of testing happens at every stage of the SDLC, meaning that at this stage most of the errors will have been found and fixed already.

The second part of this stage is the Documentation, training, and support:

This is an essential part of the Software Development Life Cycle and involves writing up a very detailed well written document that includes necessary information about the software maintenance alongside its functions and processes and how to use the product. This documentation should also include the software architecture documentation alongside the user and technical documentation. The training involves trying to enhance the employee's work abilities through developing their skills through learning and understanding.

- Deployment and maintenance of product: Once the testing has been finalized then the end product is released in multiple phases according to the strategy of the organization, this is followed by it undergoing testing in a real-world environment in order to ensure it performs correctly (an example of this could be beta testing). Once this is proven, then the organization will publish the full product. Finally, it will depend on constructive feedback from users and product supervision to improve it and help customer.

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

The agile model is a delivery process in which each incremental part is that is delivered is developed through an iteration following each timebox, it is not suitable for small projects as it is more expensive in comparison to other projects. The waterfall model is a structured model that follows a pathway through requirements gathering, SRS document preparation, analysis, coding and testing in a planned manner. These are followed in a specific order as seen within the software development life cycle. This model is not well suited to large projects despite it being easy to use and understand.

A difference between these two models is that in the agile model, the primary way of measuring progress is through functionalities and how they were developed and delivered meanwhile in the waterfall model, it is usually measure through the number of artefacts such as requirement specifications and analysis, design documents, tests, code reviews and other artefacts which have been completed and reviewed.

Unfortunately for the waterfall model, if a project is cancelled whilst in development then no product will be delivered and there will be only some documents left whilst in the agile model even if it is cancelled whilst it is still being created then the customer may still be able to be delivered some useful code that might even be in live operation. Agile models also allow changes to the requirements of the product even once the development has begun whilst the waterfall model does not allow this. Concerning customer interaction, the agile model is very interactive and deploys an incremental version following each interaction to the consumer. This does not happen with the waterfall method where the product is only delivered to the customer once the development has finished.

There tends to be more team members within a waterfall model which allows more to be done simultaneously but the interaction between team members isn’t as coordinated as the agile model where there are less than ten team members which allows them to coordinate their tasks and interact with other team members on a regular basis.

A positive part of following the waterfall method is that due to its vigorous policy of having proper documentation it allows for a clear idea to be formed on what is necessary for the product to be developed and delivered whilst the agile model’s lack of need for proper documentation may lead to confusion and misinterpretation throughout the project.

7. What is a reduced function used for?

The reduced function, known as reduce() in Python and defined in the "functools" module", is used to apply a specific function passed in its arguments to the entire list elements which are mentioned in the sequence that was past along. It works through picking the first two elements of the sequencing and giving the result. Following this, it applies the function again to the previous result and the number following the second element, storing the result once more. The function continues to do this until there are no elements left and the final result is returned and printed onto the console. This function can also be put together with the operator function to do a similar task with lambda functions and to make the code easier to read. This function works with three parameters in Python3 and 2 parameters, putting the third parameter before the second parameter if it is present. An example of reduce() is the following:

```
import functools


lis = [947, 2, 7, 9, 1]


print("Sum of list elements:: ", end="")
print(functools.reduce( lambda a, b: a + b, lis))
```

8. How does merge sort work

Merge Sort, similarly to QuickSort, is a divide and conquer algorithm. This algorithm divides the array that was input into two halves, then proceeds to call itself for the two halves it input before merging the two sorted halves. This process automatically assumes that arr[m+1..r] and arr[l..m] are sorted and combines both of the sorted sub-arrays into a singular one. The Merge Sort algorithm is seen as useful for sorting lists that are linked in O(nLogn)time. The issues that may be encountered with Merge Sort are that it can be slower in comparison to other types of sorting algorithms when it comes to completing smaller tasks. Additionally, it requires

additional memory space for the temporary array. It also needs to go through the whole process despite the cases in which the array is sorted.

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?**

   Generator functions return iterators that you can also loop over like a list, however, unlike lists they don't keep their contents in its memory. A reason to use generators is to read large files such as CSV files or to work with data streams. The generator function allows you to count the number of rows in a CSV file through the csv_reader() generator function, this allows you to open a file and loop through each line, yielding each row without returning it.

```
def csv_reader(file_name):
    for row in open(file_name, "r"):
        yield row
```

   Another reason to use generator functions is to generate an infinite sequence since the memory the computer is finite, this is done through initializing the variable num and thus starting an infinite loop before yielding num so that you can acquire the initial state, This follows the actions of range() but on an infinite sequence. Following the yield, increment num by 1 and then the program will continue to execute until you stop it.

   Generator functions are also used for detecting palindromes as it will locate all the sequences of numbers or letters that are considered to be palindromes meaning that they are written the same both forward and backwards such as "kayak".

   Apart from the yield method, generator objects can also use the .send(), .throw() and .close() methods. It can also be used for Fibonacci numbers. You can also pipeline generators meaning that

multiple generators can be used to pipeline a collection of operations. An example of this is the following where my aim was to find the addition of cubes of numbers within the Fibonacci sequence:

```
def fibonacci_numbers(nums):
    x, y = 0, 1
    for _ in range(nums):
        x, y = y, x+y
        yield x

def cubed(nums):
    for num in nums:
        yield num**3

print(sum(cubed(fibonacci_numbers(10))))
```

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

A decorator in software engineering term is a design pattern tool which is used for wrapping functions or classes in code, allowing existing functions to have more functions added on or classes without disrupting the initial structure, since both functions are callable then what it returns is a callable as well. It basically modifies the behaviour of a class or function. Here is an example of a single decorator:

```
def function1(function):
    def wrapper():
        print("hello")
        function()
        print("I don't like describing decorators")

    return wrapper


def function2():
    print("I want to sleep")


function2 = function1(function2)
function2()
```

Another example would be adding functionality to a function such as creating a new user as shown below:

```
def create_new_user(username, password, first_name, last_name, email_address,
phone_number):
    new_user = {
        "username": username,
        "password": password,
        "first_name": first_name,
        "last_name": last_name,
        "email_address": email_address,
        "phone_number": phone_number
    }

    headers = {'content-type': 'application/json'}
    response = requests.post('http://127.0.0.1:5000/api/user', headers=headers
, data=json.dumps(new_user))
    print(response)
```

Decorators may also have arguments within them:

```
def my_decorator_function(func):

    def this_is_a_wrapper_function(*args, **kwargs):

        func(*args, **kwargs)

    return this_is_a_wrapper_function


@my_decorator_function
def my_function(my_arg):
    pass

print(my_function.__name__)
print(my_function.__doc__)
```

The use of decorators also ensures that you code doesn't repeat itself. Decorators are useful for things such as login required functionalities or simply for adding functionalities to functions in general without changing the origin code.

However, the use of decorators can often lead to confusion as they are supposed to be kept in a different file to the main code and it can be difficult to understand what it does to the code just through looking at it without finding the decorators within the code itself. Another negative aspect of decorators is that once they are applied

to a function then whenever the function is called on, the decorator is automatically applied and if you want to remove the decorator then you would have to write the whole function again which can lead to errors and confusing code as well as it being time consuming for the developer.