THEORY QUESTIONS ASSIGNMENT
Software Stream
Lorelle Brownlee

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.

1. How does Object Oriented Programming differ from Process Oriented Programming?
Process-oriented programming utilised a top-down approach to running a programme

Benefits of object oriented programming include:
- Easy to maintain code,
- Easy to reuse for multiple purposes
- Easy to expand functions of parts of code

Disadvantages of object oriented programming:
- Code can become increasingly complex

Benefits of process oriented programming:
- Structure of code is highly logical
- Code works best when highly simplified, so drives simplification

Disadvantages of process oriented programming:
- Can be difficult to maintain
- Difficult to expand functionality when desired

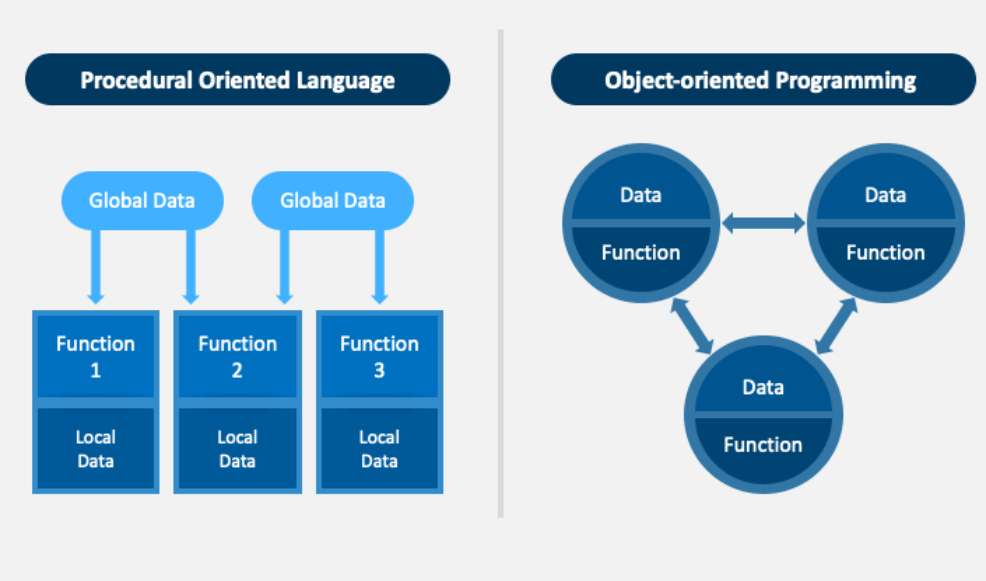


Image taken from sketchbubble.com

2. What's polymorphism in OOP?
Polymorphism is a concept in OOP that describes the ability to use an entity (a method, object or operator) to perform different actions on different types depending on the situation.
One example of polymorphism in OOP is the use of the addition operator. If using '+' on integer data types, the operator is used to perform arithmetic, as seen below.

```
a =  3
b = 8
print(a + b)

#output: 11
```

However, if used on string data types, the operator performs concatenation.

Polymorphism can also describe functions that are compatible with multiple data types. One example is the in-built Python function len().

Below are some examples of the len() function being run on different data types (string, list, and dictionary respectively).

```
print(len("Charmander"))
#output: 10

print(len(["Pidgeot", "Bulbasaur", "Clefairy", "Gengar"]))
#output: 4

print(len({"Pokemon name": "Clefairy", "Region": "Kanto", "Type":
Fairy}))

#output: 3
```

It can also exist in classes.

3. What's inheritance in OOP?

Inheritance is a concept in OOP where a class can inherit all methods and properties from another class.
In inheritance, there is a parent class (aka “super class” or “base class”) and a child class (aka “subclass” or “derived class”).

Some benefits of inheritance include:

- Reusability of pieces of code
- Avoids duplicate data and redundant data
- Reduces space and time complexity
- Helps improve modular structure of code

Some disadvantages of inheritance include:

- Can slow down the execution of code
- Problems can arise from parent and child classes being tightly coupled.

```
class pokemon:
  def __init__(self, pokemon_name, gender):
    self.pokemonName = pokemon_name
    self.pokemonGender = gender

  def printPokedetails(self):
    print(self.pokemonName, self.pokemonGender)

#Use the Person class to create an object, and then execute the
printname method:



class evolved(pokemon):
     pass

x = evolved("Clefairy", "M")
x.printPokedetails()


#output: Clefairy M
```

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?

At a high level, I would first prompt all voters to vote for who they think is the funniest person in the office. They would enter one nominee's name in response to the prompt.
Each voter would be allowed to vote once.
Every individual vote would have the input item (the funny person's name) appended to a list.

I would use the Counter method from collections, to create a dictionary of keys and values. The keys would be nominees and the values would be the sum of all the votes that were cast for them.
I would then create a new list, comprising just the keys of the dictionary. On this list, I would then use the sorted() function. I will also specify that the sorting should be in reverse order.

Once this is sorted, I can print the first 3 items in the new list, which will correspond to the top 3 funniest individuals as voted for by their colleagues.

Example code:

```
# Function to find funniest office team member
from collections import Counter

input =['john','johnny','jackie','johnny',
            'john','jackie','jamie','jamie',
            'john','johnny','jamie','johnny',
            'john', 'john']
# initializing list
total_votes = Counter(input)

# Sort List by dictionary values
results = sorted(total_votes, key= total_votes.get, reverse= True)

# printing result in order
print(results[0:3])
```

5. What's the software development cycle?

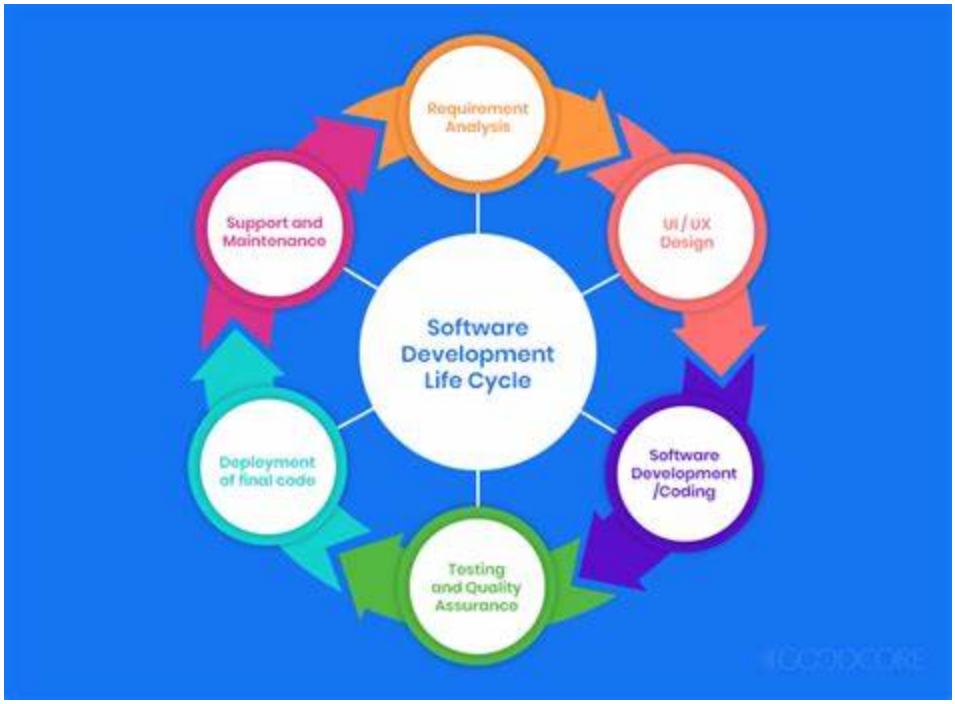
This is the development of software for specific purposes. It is a 6-stage process that requires careful planning and execution.

The plan will outline specific tasks to be undertaken by developers at specific stages. The planning structure enables the team to meet the expectations of stakeholders and adhere to the allocated budget.

The 6 stages are:

1. Planning and requirement analysis
- Inputs from customers, the sales team, stakeholders and other relevant parties are taken on board to establish product requirements. This forms the basis of the basic product design.
2. Defining requirements
- All requirements are specified in this stage. The requirements then receive approval from relevant parties. A Software Requirement Specification (SRS) document is compiled outlining those specifications.
3. Designing architecture
- A Design Document Specification (DDS) is created using the specifications outlined in the SRS. Within the DDS, multiple product architecture designs are outlined. Market analysts and stakeholders assess the designs in the DDS, and select the most appropriate design to take forward.
4. Developing the product
- Developers begin basic work on the project using the programming language specified in the selected design from the DDS.
5. Product testing and integration
- Once the basic product has been developed, more thorough testing of the software can take place. Small amounts of testing takes place throughout all stages of the software development life cycle, but more thorough testing takes place at this stage. All identified issues are tracked, amended and retested.
6. Deployment and maintenance of the product
- After thorough testing and amendments, the final product is released in phases in alignment with the strategy of the organisation. Testing continues in real-world conditions. If performance is satisfactory, the whole product can be released. Feedback is obtained from users and stakeholders as the product is used, and improvements are continually made in line with this feedback.

Documentation, training and support is also an essential part of the process. This is a critical part of the SDLC, as thorough and clear documentation allows for smooth maintenance and improvement to the product. Some critical forms of documentation include software architecture documentation, user documentation and technical documentation.



Taken from www.goodcore.co.uk

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

Agile and Waterfall are both project management methodologies utilised in software development projects. Waterfall is somewhat older than Agile, and Agile is increasingly the more popular methodology of the two within the software development domain. There are a number of differences between the two methodologies:

	Agile	**Waterfall**
Approach	Incremental, iterative	Linear, sequential
Work breakdown	Divides a project into smaller ‘sprints’, which have specific team roles and a well-defined structure	Divides a project into smaller phases
	Works for many small project	Designed for one project
Mindset	Focuses on end user/ stakeholder satisfaction	Focuses on product delivery
Requirements setting	Performed daily	Performed once at the start of the project

Requirement changes	Allows for changes at any time	Ideally avoids scope changes after project has already started
Testing	Performed alongside development	Comes only after 'build' phase
	Test teams can contribute to requirement changes	Test teams do not get involved in requirement changes
Project management	No dedicated project manager - entire team 'self manages' the project	Requires a project manager as an essential role in every phase

7. What is a reduced function used for?

The reduce() function executes a technique called reduction. This is a mathematical technique that reduces any iterable to a single cumulative value.

It does this by:

1. Applying a function to the first two items in an iterable. This generates a partial result.
2. Repeating this step, but this time applying the function to two different items: the partial result from the previous step, and the third item in the iterable. This generates a new partial result.
3. The process is repeated until there are no more iterables, and a single cumulative value results.

The reduce() function can be beneficial as it is written in C, and can internally run faster than an explicit for loop in python undertaking a similar task.

To use the reduce() function, it must be imported from functools.

An example of the use of reduce() is below:

```
from functools import reduce

po = [0, 1, 2, 3, 4]

>>> reduce(my_add, numbers)
0 + 1 = 1
1 + 2 = 3
3 + 3 = 6
6 + 4 = 10
10
```

8. How does merge sort work

Merge sort is a recursive ‘divide and conquer’ algorithm. It can be used on an input array to divide it into two halves, call itself on each of the two halves, then merge the two sorted halves. It does this by

1. Finding the middle point in the array, to use this as a marker for where to split the array in two.
2. Call mergeSort for the first half
3. Call mergeSort for the second half
4. Merge the two halves by calling merge().

The algorithm is recursive, so it will continue to call itself in the way described above until it has divided into such small parts that each of the two halves of the arrays in each call are size 1. Once the array size reaches 1, the merge process begins.

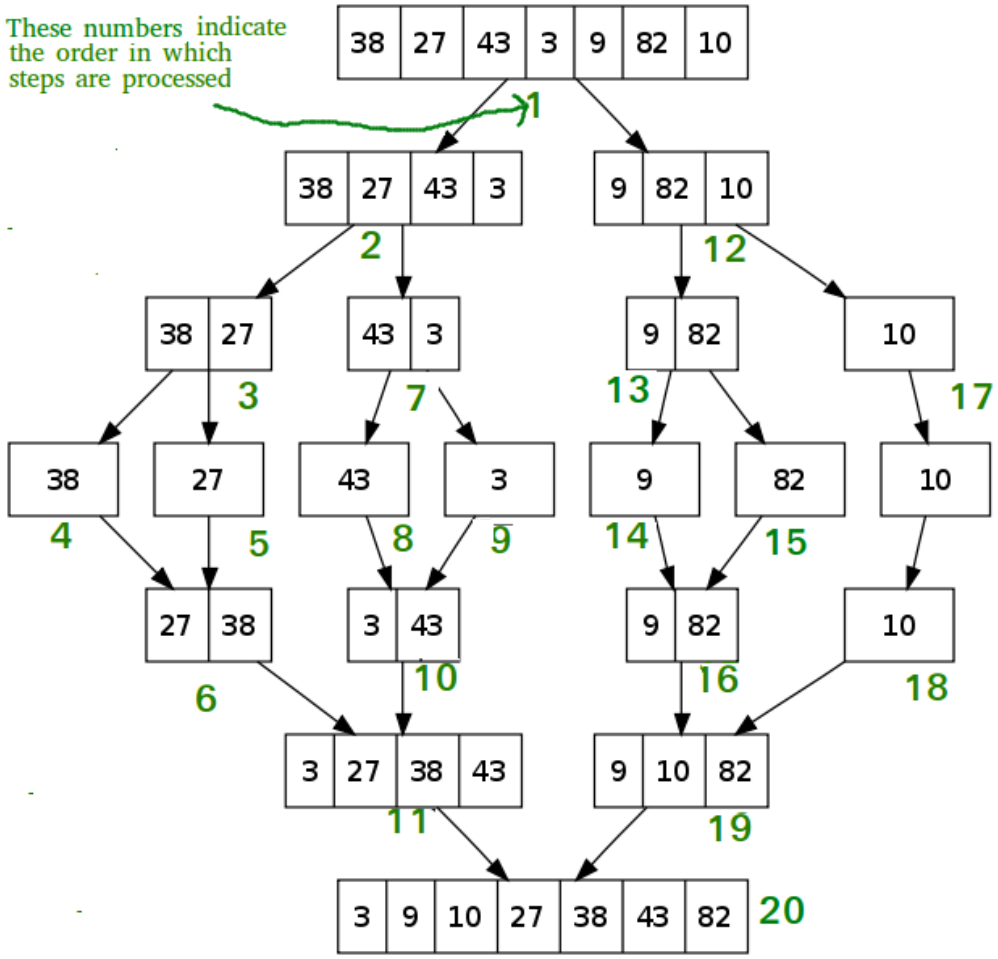


Image taken from Wikipedia.

Merge sort is useful for sorting linked lists.

Disadvantages of merge sort:

- Needs additional memory space of 0(n) size to store the new temporary array in
- If array is sorted already, the algorithm will still complete the whole process
- If the task is small, some other algorithms may be faster.

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?

A generator function is similar to a standard function, but it generates a value and does so using the 'yield' keyword rather than 'return'. If a function's body contains the 'yield' keyword, the function is automatically a generator function.
The generator function will return an iterable object.

One example for the use of generator functions is the generation of Fibonacci numbers. Fibonacci numbers are commonly used for xxx. They are sequences of numbers where each number in the sequence equals the sum of the two preceding numbers in the sequence. Fibonacci numbers represent numerous phenomena that occur in nature and hence they can have many uses. It even sometimes forms a part of the Agile project management methodology, to estimate Product Backlog.

```
# A simple Fibonacci sequence generator

def fibonacci(limit):

      # Initialize first two Fibonacci Numbers
      a, b = 0, 1

      # One by one yield next Fibonacci Number
      while a < limit:
            yield a
            a, b = b, a + b

print("\nUsing for in loop")
for i in fib(40):
      print(i)
```

Generator functions were Introduced with PEP 255, and the kind of iterator they return is called a ‘lazy iterator’. Lazy iterators are objects that you can loop over like a list. However, lazy iterators don’t store their contents in memory.

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

Functions and methods are called ‘callable; objects as they can be called. A decorator is a callable that returns a callable. It adds some functionality to a function, and then returns it.

@jit

JIT is short for Just In Time compilation. Normally when we run Python code, compilation takes place first. This is where the Python code converts into C. . This takes time as data types are allocated memory and stored as unassigned but named objects. @jit allows for most of this to be done at execution. This is somewhat similar to parallel computing, where the Python interpreter works on two things at once to save time.

@jit is in the Numba package, and it makes running more intensive software a lot easier without having to compile into C.

```
from numba import jit

import random
```

```
@jit(nopython=True)
def monte_carlo_pi(nsamples):
   acc = 0
   for i in range(nsamples):
       x = random.random()
       y = random.random()
       if (x ** 2 + y ** 2) < 1.0:
           acc += 1
   return 4.0 * acc / nsamples
```

@do_twice

This decorator can run a function twice with a single call. This can be useful for measuring performance of two different iterations of a function, to optimise the performance of the function. An example can be seen below.

```
from decorators import do_twice
@do_twice
def timerfunc():
   %timeit factorial(15)
```

@count_calls

This decorator can be used to indicate how many times a function is used in a piece of software. It can be helpful for making code more concise and reducing code duplication. It can also be helpful for identifying functions that might benefit from optimisation. Like do_twice, this certainly can come in handy for debugging. When added to a given function, the output will say how many times the function has been run each time it runs. This decorator is in the decorators module from the Python standard library.

```
from decorators import count_calls

@count_calls

def function_example():

   print("Hello World!")

function_example()

function_example()

function_example()
```