THEORY QUESTIONS ASSIGNMENT

Software Stream

Maximum score: 100

Maximum score: 100

80/100

amazing!

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**

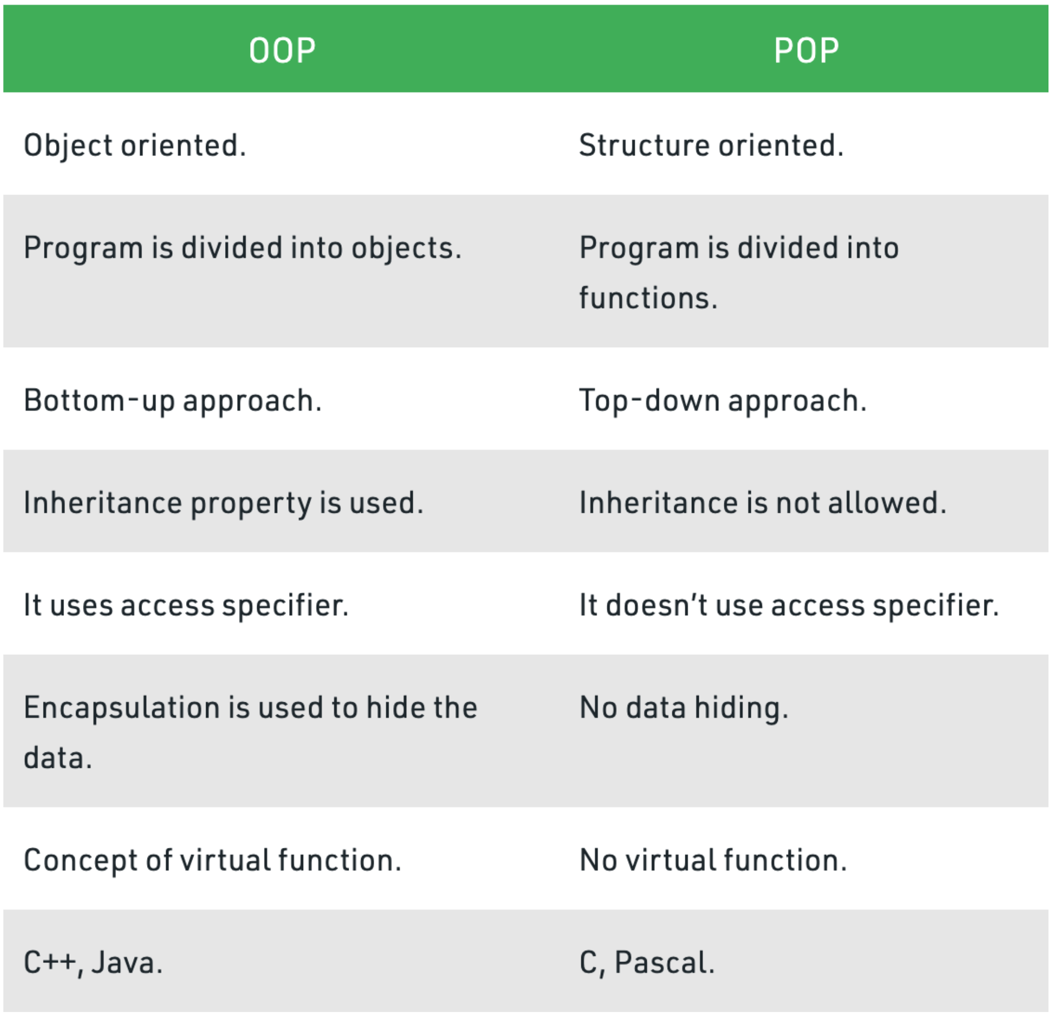
1. How does Object Oriented Programming (OOP) differ from Process Oriented Programming (POP)?

OOP:
Related with the real life objects and their properties. OOP Concepts:

- ❖ Class and Objects
- ❖ Data abstraction ✓
- ❖ Encapsulation
- ❖ Polymorphism   how do they work? particularly the classes and objects part
- ❖ Inheritance

POP:
Related with the conventional style. This approach is also known as the top-down approach. In this approach, a program is divided into functions that perform specific tasks. ✓ This approach is mainly used for medium-sized applications. Data is global, and all the functions can access global data. The basic drawback of the procedural programming approach is that data is not secured because data is global and can be accessed by any function. ✓ Program control flow is achieved through function calls and go to statements.

Difference between OOP and POP: 8/10



OOP	POP
Object oriented.	Structure oriented.
Program is divided into objects.	Program is divided into functions.
Bottom-up approach.	Top-down approach.
Inheritance property is used.	Inheritance is not allowed.
It uses access specifier.	It doesn't use access specifier.
Encapsulation is used to hide the data.	No data hiding.
Concept of virtual function.	No virtual function.
C++, Java.	C, Pascal.

✓ ✓ ✓ ✓ ✓ ✓

why would someone use OOP over POP or vice versa? if feasible can you go more in depth

2. What's polymorphism in OOP?

Polymorphism translates into "many forms". In Python it allows common functions to be used for different types. For example, len() is a function that counts the length of something. It can be used for different types like a string where the number of characters are counted or a list type where the number of items in the list. ✓ ✓

```
print(len("girls"))
Output: 5
```

can you discuss adv/disadv (even if there aren't potential any for either side) for polymorphism?

```
print(len([26, 36, 76]))
Output: 3
```

Polymorphism is particularly powerful as you can also call methods within class types. Doing so allows you to assign different outcomes for the method based on ✓

the class. This is shown quite well using a for loop:

```
class India():
    def capital(self):                                                      ✓
        print("New Delhi is the capital of India.")

    def language(self):
        print("Hindi is the most widely spoken language of India.")

    def type(self):                                                         ✓
        print("India is a developing country.")

class USA():
    def capital(self):
        print("Washington, D.C. is the capital of USA.")

    def language(self):                                                     ✓
        print("English is the primary language of USA.")

    def type(self):
        print("USA is a developed country.")
                                                         9/10
obj_ind = India()
obj_usa = USA()
for country in (obj_ind, obj_usa):                       ✓
    country.capital()
    country.language()
    country.type()
```

Output:

```
New Delhi is the capital of India.
Hindi is the most widely spoken language of India.      ✓
India is a developing country.
Washington, D.C. is the capital of USA.
English is the primary language of USA.
USA is a developed country.
```

when it relates to inheritance, wouldn't it be overriding?

Polymorphism also works very well with the concept of inheritance within Python and is known as Method Overloading. The next answer explains this concept.

3. What's inheritance in OOP?

As polymorphism allows you to define methods within a child class where the parent class has methods of the same name. Inheritance is when the child class inherits the methods from the parent class. It is possible to modify a method in a child class once it has been inherited. Inheritance reduces complexity and the problem of writing the same code multiple times and increases reusability. This is useful when the parent class method doesn’t quite fit the child class. Doing so, allows us to re-implement and modify the method within the child class. This is known as Method Overloading:

you may be confusing overloading with overriding unfo
Overloading is when funcs have same name but diff number
of parameters (so what's used is decided by parameter passed)

Overriding is when, using inheritance, one method is supercedes
/ takes more priority than the parent classes identical method

e.g. this child method has a new approach
but same name

this is polymorphism via overriding!

```
class Bird:
  def intro(self):
    print("There are many types of birds.")

  def flight(self):
    print("Most of the birds can fly but some cannot.")

class sparrow(Bird):
  def flight(self):
    print("Sparrows can fly.")

class ostrich(Bird):
  def flight(self):
    print("Ostriches cannot fly.")

obj_bird = Bird()
obj_spr = sparrow()
obj_ost = ostrich()

obj_bird.intro()
obj_bird.flight()

obj_spr.intro()
obj_spr.flight()

obj_ost.intro()
obj_ost.flight()
```

8/10

Output:

```
There are many types of birds.
Most of the birds can fly but some cannot.
There are many types of birds.
Sparrows can fly.
There are many types of birds.
Ostriches cannot fly.
```

Method Overloading means a class containing multiple methods with the same name but may have different arguments. Python doesn't support this, but inheritance is one of the ways in which Method Overloading can still be achieved.

you can do overloading normally in python?

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?

0/10

5. What's the software development cycle?

There are 6 main parts to the Agile Software Development Cycle:

- Planning
- Analysis
- Design
- Implementation
- Testing & Integration
- Maintenance

✓

what is the adv/disadv of such a structured process?

9/10

✓ During the first two parts requirements for the project are gathered where project managers and stakeholders take the lead. After the requirements are gathered they are analysed for validity and the possibility of incorporating the requirements in the system is also studied.

✓ In the following Design phase, the system and software design is prepared from the specifications completed in the first steps. Here, hardware and system requirements are specified which helps define overall system architecture. Testers also come up with the test strategy so decide what and how to test.

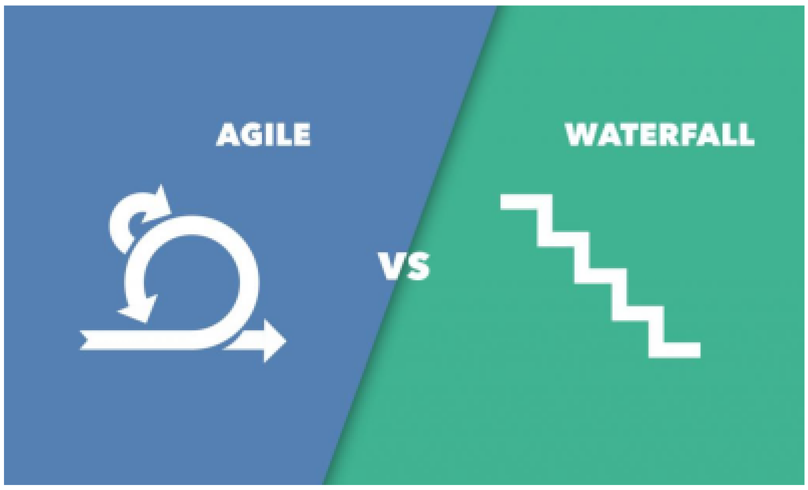
✓ The Implementation part involves the core coding work and mainly concerns the developer team. It is usually the longest phase of the software development cycle.

✓ Once the developers are happy with the main code, the testing team begins checking whether the code meets the requirements. All types of functional testing are done such as unit testing, integration testing, system testing, acceptance testing as well as non-functional testing. It can then be integrated into the delivery product/deployed to the customer to be used. The customers will then do beta testing.

✓ Lastly, maintenance is once the final product is developed. Any final bugs in the system are caught and fixed by the developer team before final deployment, after which the product should be continuously maintained.

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

✓ Agile is a modern project management methodology and is usually depicted using a loop. In contrast, Waterfall is the traditional project management methodology, which is usually depicted using a stairway. I will outline them using the respective images below:



if feasible can you go into more depth? why would someone use agile over waterfall or vice versa? whats lost / gained with either approach? current approach below is descriptive, if possible be critical of each side as well

✓

8/10

e.g. if agile is flexible and preferred by tech, then why does waterfall exist / be used even now?

Agile is known for its incremental approach as it focuses on people, results and collaboration. It is highly regarded for its flexibility with high adaptability to change. In this way a whole project is broken down into small increments, which are completed in interactions or short time frames. Each iteration involves a development cycle (explained in Q5). With a working product delivered at the end of each iteration, a new or updated product is released at the end of several iterations. Agile is the current preferred methodology for many tech companies due to its flexibility and quick production of deliverables. ✓

Waterfall originates from manufacturing and construction-type industry projects. The base core mentality of this method is that tasks need to be done in succession one by one. In software development this applies to specialised tasks that may need to be reviewed before moving on to the next phase. Waterfall is known for being linear and sequential, flowing downwards like a staircase or waterfall. ✓

7. What is a reduced function used for?

✓ The reduce() function in Python “folds” or “reduces” a list of items into a single cumulative value - a mathematical technique referred to as “folding” or “reduction”. It operates on any iterable, not just lists. It is performed via these steps:

✓
- **Apply** a function (or callable) to the first two items in an iterable and generate a partial result.
- **Use** that partial result, together with the third item in the iterable, to generate another partial result.
- ✓ **Repeat** the process until the iterable is exhausted and then return a single cumulative value.

✓ The idea behind Python’s reduce() is to take an existing function, apply it cumulatively to all the items in an iterable, and generate a single final value. In general, Python’s reduce() is handy for processing iterables without writing explicit for loops as its internal loop can be faster.

✓ Python’s reduce() was originally a built-in function (and still is in Python 2.x), but it was moved to functools.reduce() in Python 3.0. This decision was based on some possible performance and readability issues.

✓ Another reason for moving reduce() to functools was the introduction of built-in functions like sum(), any(), all(), max(), min(), and len(), which provide more efficient, readable, and Pythonic ways of tackling common use cases for reduce().

The reduce() function is used in multiple ways, which are listed below with a couple of examples:

- Summing numeric values
  - ✓ When myadd() adds two numbers (a and b) and returns the result, reduce can be used to calculate the sum in a Python iterable:

    -
      ```
      >>> from functools import reduce
      >>> numbers = [1, 2, 3, 4]
      >>> reduce(my_add, numbers)
      10
      ```
- Multiplying Numeric values
- Finding the minimum and maximum values
  - Can be done with a for loop:
    - ✓
      ```
      >>> numbers = [3, 5, 2, 4, 7, 1]
      >>> # Minimum
      >>> min_value, *rest = numbers
      >>> for num in rest:
      ...     if num < min_value:
      ...         min_value = num
      …
      >>> min_value
      1

      >>> # Maximum
      >>> max_value, *rest = numbers
      >>> for num in rest:
      ...     if num > max_value:
      ...         max_value = num
      …
      >>> max_value
      7
      ```
  - Can be done with less lines of code via a reduce() function when the solution uses two user-defined functions, e.g. the first takes two arguments (a and b) and return their minimum, and the second has a similar process but returns the maximum value:
    - ✓
      ```
      >>> from functools import reduce
      >>> # Minimum
      >>> def my_min_func(a, b):
      ...     return a if a < b else b
      …

      >>> # Maximum
      >>> def my_max_func(a, b):
      ```

can you mention the 3rd param of reducer, go into more detail?

10/10

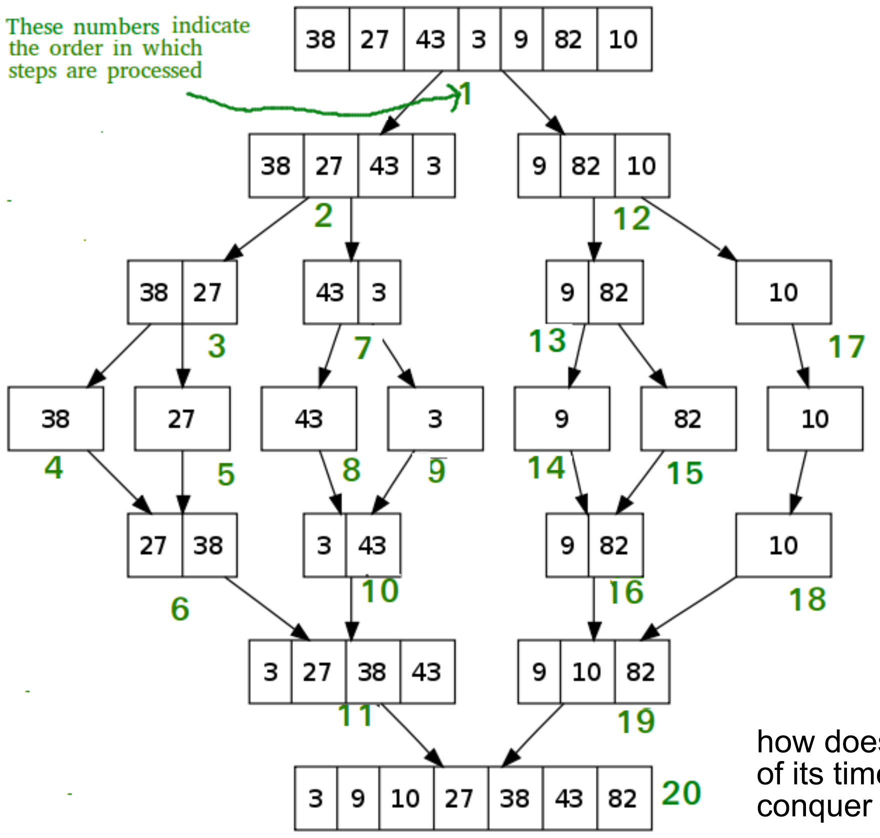
```
...     return a if a > b else b
…

>>> numbers = [3, 5, 2, 4, 7, 1]

>>> reduce(my_min_func, numbers)
1

>>> reduce(my_max_func, numbers)
7
```

✓

- ❖ Checking if all values are true
- ❖ Checking if any value is true

8. How does merge sort work

✓ A Merge Sort is a Divide and Conquer algorithm. It divides the input array into two halves, calls itself for the two halves, and then merges the two sorted halves. The image below shows how a Merge Sort divides an array recursively into two halves ‘til the size merges back into one. It can also be used to merge two lists.



✓

9/10

how does it accomplish the log N part of its time perf (e.g. discuss how divide-conquer strategy plays, why use it)

can you discuss the adv of merge sort + more info on its performance (altho discussed in disadv, mean more along its current time & space complexity ordinarily e.g. N log N time)

There are a few drawbacks of Merge Sort:

✓

- ❖ Slower compared to the other sort algorithms for smaller tasks.

✓

- ❖ Merge sort algorithm requires an additional memory space of 0(n) for the temporary array.
- ❖ It goes through the whole process even if 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?

✓ Unlike for loops, generators save the state of the function so the next time a function is called, execution continues from where it left off with the same variable values it had before yielding.

✓ It can be very helpful for multiple use cases such as generating an infinite sequence or detecting palindromes. Here I run through the use case of reading large files and working with data like CSV files. Below is an example of how you could try to find the number of rows within a CSV file:

```
csv_gen = csv_reader("some_csv.txt")
row_count = 0
for row in csv_gen:
    row_count += 1
print(f"Row count is {row_count}")
```

✓ Looking at this example, you might expect csv_gen to be a list. To populate this list, csv_reader() opens a file and loads its contents into csv_gen. Then, the program iterates over the list and increments row_count for each row.

✓ This is a reasonable explanation, but would this design still work if the file is very large? What if the file is larger than the memory you have available? To answer this question, let's assume that csv_reader() just opens the file and reads it into an array:

```
def csv_reader(file_name):
    file = open(file_name)
    result = file.read().split("\n")
    return result
```

✓ This function opens a given file and uses file.read() along with .split() to add each line as a separate element to a list. If you were to use this version of csv_reader() in the row counting code block you saw further up, then you'd get the following output:

```
Traceback (most recent call last):
  File "ex1_naive.py", line 22, in <module>
    main()
  File "ex1_naive.py", line 13, in main
    csv_gen = csv_reader("file.txt")
  File "ex1_naive.py", line 6, in csv_reader
    result = file.read().split("\n")
MemoryError
```

✓ In this case, open() returns a generator object that you can lazily iterate through line by line. However, file.read().split() loads everything into memory at once, causing the MemoryError.

So, how can you handle these huge data files? Take a look at a new definition of csv_reader():

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

10/10

✓ In this version, you open the file, iterate through it, and yield a row. This code should produce the following output, with no memory errors:

```
Row count is 64186394
```

✓ Here, you've essentially turned csv_reader() into a generator function. This version opens a file, loops through each line, and yields each row, instead of returning it.

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

✓ Decorators "decorate" or "wrap" another function and let you execute code before and after the wrapped function runs. Decorators allow you to define reusable building blocks that can change or extend the behavior of other functions. And, they let you do that without permanently modifying the wrapped function itself. The function's behavior changes only when it's decorated.

✓ The @ syntax is just a shorthand for calling the decorator onan input function. Multiple decorators on a single function are applied bottom to top (decorator stacking).

✓ As a debugging best practice, use the functools.wraps helper in your own decorators to carry over metadata from the undec-orated callable to the decorated one.

✓ Just like any other tool in the software development toolbox, decorators are not a cure-all and they should not be overused. It's important to balance the need to "get stuff done" with the goal of not making your code messy and difficult to read, ultimately harming the design. It is highly advised not to use decorators in conjunction with object-oriented inheritance and it can lead to a design that is nearly unmaintainable.

9/10

can you give examples, or discuss how it may be used currently in programs e.g. flask?