**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?**  
   *Process Oriented Programming is a command-driven language where the user has to specify “what to do” and also “how to do” step-by-step procedures. These instructions are executed in the sequential “top-down” order and written to solve specific problems.   
   Object Oriented Programming (OOP) is where the user only has to specify “what to do” and not “how to do”. It is based on concepts of objects, that contain data in the form of attributes and code in the form of methods. OOP follows a “bottom-up” approach.*
2. **What's polymorphism in OOP?**  
   *Polymorphism is one of the core concepts in OOP where it describes that different data types can be used within the same interface. It refers to the ability of a variable, function or object to take on multiple forms hence allowing programmers to program in general rather than program in specific.*
3. **What's inheritance in OOP?***It is a mechanism where one class inherits the attributes and methods of another class. The class whose properties and methods are inherited is known as the Parent class and the class that inherits the properties from the parent class is the Child class.With inheritance, fields and methods of existing classes can be re-used.*
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 quote 1 funny comment or joke that was said by each person as a statement. Then I would request the user to chose/vote which of the statements listed that they found the most funniest and scale it from 1 (most funniest), 2 (moderate funny), 3 (least funny). This introduces randomisation on the people.*
5. **What's the software development cycle?**  
   *The Software Development Life Cycle (SDLC) is a framework commonly used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates. It’s typically divided into six to eight steps: Planning, Requirements, Design, Build, Document, Test, Deploy, Maintain.*
6. **What's the difference between agile and waterfall?**  
   *Agile is an iterative methodology that incorporates a cyclic and collaborative process. Underneath the umbrella of Agile are a range of frameworks and product delivery methods, such as Scrum, Lean, Six Sigma, and Kanban. Agile frameworks and methods do not generally manage the project lifecycle, rather provide a flexible and iterative solution, which allow you to adapt as the situation demands.   
   Waterfall is sequential and can also be collaborative, but tasks are generally handled in a more linear process. It focuses on up-front planning with requirements fully defined before a project commences. Like its name suggests, work cascades, much like a waterfall, through different project phases. Each phase needs to be completed before the next one can begin.*
7. **What is a reduced function used for?**  
   *Python's reduce() is a function that implements a mathematical technique called folding or reduction. reduce() is useful when you need to apply a function to an iterable and reduce it to a single cumulative value.  
   Example:*>>> from functools import reduce

>>> **def** do\_sum(x1, x2):   
 **return** x1 + x2

>>> reduce(do\_sum, [1, 2, 3, 4])

10

1. **How does merge sort work?**  
   *Works like a divide-and-conquer sorting algorithm. Merge sort works by splitting the input list into two halves, repeating the process on those halves, and finally merging the two sorted halves together.*
2. **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 are special kind of function that behaves like an iterator. They allow programmers to make an iterator (an iterable object such as strings, lists, dictionaries) in a fast, easy, and clean way.*  
     
   *Generators also provide a very neat way of producing data which is huge or even infinite.   
     
   Some use case examples include:   
   1) Reading large files - like .csv or .txt files; below code can be used to count rows in a given large 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}")  
     
   2) Generating an infinite sequence - since computer memory is finite, the use of generator will help generate an infinite sequence. This is one useful method for detecting palindromes.   
     
   The program will continue to execute until you stop it manually.  
     
   Below code increases a number by 1:  
   def infinite\_sequence():  
    num = 0  
    while True:  
    yield num  
    num += 1  
     
   The crucial advantage of generators consists in automatically creating the methods* ***\_\_iter\_\_()*** *and* ***next()****.  
   Example:***def** city\_generator():

**yield**("Hamburg")

**yield**("Konstanz")

**yield**("Berlin")

**yield**("Zurich")

**yield**("Schaffhausen")

**yield**("Stuttgart")   
  
city = city\_generator()  
**print**(**next**(city))  
>>> Hamburg  
**print**(**next**(city))  
>>> Konstanz  
*… and so on…  
until the last one…*  
---------------------------------------------------------------------------

**StopIteration** Traceback (most recent call last)

<ipython-input-22-02870f953c83> in <module>

----> 1print(next(city))

**StopIteration**:

1. **Decorators - A page for useful (or potentially abusive?) decorator ideas. What is the return type of the decorator?**  
   *In Python, a decorator is a design pattern that allows a user to add new functionality to an existing object without modifying its structure. Decorators are usually called (in programming) before the definition of a function you want to decorate.*   
     
   *Example:*  
   @my\_decorator

def hello():

print('hello')  
  
*This function hello above is known as a function object. The @my\_decorator is actually a function that has the ability to use the hello object, and return a different object to the interpreter. The object that the decorator returns, is what becomes known as hello. Essentially, it's the same thing as if you were going to write your own normal function, such as: hello = decorate(hello). Decorate is passed to the function -- which it can use however it wants -- then returns another object. The decorator has the ability to swallow the function, or return something that is not a function, if it wanted.  
  
Using decorators can ensure that your code is DRY (Don’t Repeat Yourself).   
  
A few use cases of decorators are: Authorization in Python frameworks (i.e. Flask and Django), Logging, Measuring execution time, Synchronization.*