**THEORY QUESTIONS ASSIGNMENT**

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

**Object Oriented Programming (OOP)** is a programming model that organized software design around data and objects, rather than functions and logic.

Object Oriented Programming Langaugaes include: Python, Java, C++, C#, Ruby etc.

OOP can be defined by programs that are made from objects, which are able to hold data that have unique attributes and behaviours.

Using OOP, we are able to manipulate and modify data.

Objects have attributes ( for example first\_name, last\_name ) and methods are actions that can be applied to the objects.

class Student:  
 def \_\_init\_\_(self, first\_name, last\_name):  
 self.first\_name = first\_name  
 self.last\_name = last\_name  
  
student1 = Student("Adriana", "Wegrzynowska")  
print(student1.first\_name)  
print(student1.last\_name)

**Process Oriented Programming (POP**) relies on functions that make up the software and can be executed at any point. It is harder to modify and change data in process-oriented programming.

The computer reads the code as a series of instructions and executes it in order.

1. **What is polymorphism in OOP?**

Polymorphism in OOP is the ability to use one function in different ways.

We can use this concept in creating class methods. We can have methods in different classes to have the same name, but using Polymorphism will allow us to use the same method in different classes, and we will have different final outputs.

There are two concepts: overloading and overriding.

Overloading is a concept when we create many different methods with the same name but the arguments and parameters will be different.

Overriding is a concept that is used to redefine specific methods and attributes of a parent class to fit the needs of a child class.

1. **What is inheritance in OOP?**

Inheritance is the concept that one class acquires the attributes and methods of another class. The first class will be the parent, and the inherited class will be the child.

Inheritance is represented using UML. UML, also known as Unified Modelling Language, is method used to show class hierarchies. That means that the parent class will be located at the top, just before the child class. That helps to read and understand the code of other people.

1. **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?**

from collections import Counter  
  
colleagues = ['Dwight', 'Michael', 'Jim', 'Pam', 'Jan', 'Erin', 'Kevin']  
votes = []  
  
while len(votes) < len(colleagues):  
 vote = input("Who is the funniest person in the office?: ")  
 if vote in colleagues:  
 votes.append(vote)  
 else:  
 print("this person is not in our office. Check your spelling!")  
  
total\_votes = Counter(votes)  
  
results = {k: v for k, v in sorted(total\_votes.items(), key=lambda item: item[1], reverse=True)}

top\_three = {k: results[k] for k in list(results)[:3]}  
  
print(top\_three)

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

Software Development Cycle ( also known as Software Development Life Cycle SDLC) is a process used to plan, create and maintain software.

1. **Planning**

This is the initial stage of the SDLC where all important aspects of the project and production management are covered.

1. **Analysing**

At this stage, all details that are reuired for the project are gathered as well as the first ideas for prototypes are determined.

1. **Design**

Developers will outline the details for the overall project, along the specificc aspects such as user interfaces, system interfaces, network requirement and databases.

1. **Implementation**

In this part, developers actually write the code and build the whole project according to the design that have been made earlier.

1. **Test and integrate**

In this phase, the application is being tested and developers are making sure that there are not any bugs and everything works correctly.

1. **Deployment**

The overall design will come together. The information system will be integrated into its environment and eventually installed

1. **Maintenance**

Users can discover bugs that were not found in test phase, so developers are still working on the project to correct all bugs.

1. **What is the difference between agile and waterfall?**

In Agile frameworks such as Scrum or Kanban, the team is encouraged to work on different phases of the SDLC at the same time. That speeds up the whole process and the clients can give more input.

The waterfall is a more traditional approach and is focused on completing one task before starting the next one. It is more linear and takes much more time to finish the project.

However, each phase is clearly defined and the process is systematically reviewed.

1. **What is reduced function used for?**

The reduced function is a type of function used in lists, tuples, dictionaries and other inerrable types of data in general.

Reduce() has been removed in Python 3 and replaced by the **functools module.**

* Reduce function is used for calculating the sum of numeric values.
* Reduce function is used for finding the minimum and maximum value in an inerrable.
* Reduce function is used to check if all values are true in an iterable.

1. **How does the merge sort work?**

Merge sort is a sorting algorithm based on ‘divide and conquer technique.

Merge sort’s principle is to divide the list into two halves, then it will   
call the sub-parts of these two halves to further divide again into two halves.

It continues to repeat until there is only one element in each sub-part.

def mergeSort(array):  
 if len(array) > 1:  
  
 # r is the point where the array is divided into two subarrays  
 r = len(array)//2  
 L = array[:r]  
 M = array[r:]  
  
 # Sort the two halves  
 mergeSort(L)  
 mergeSort(M)  
  
 i = j = k = 0  
  
 # Until we reach either end of either L or M, pick larger among  
 # elements L and M and place them in the correct position at A[p..r]  
 while i < len(L) and j < len(M):  
 if L[i] < M[j]:  
 array[k] = L[i]  
 i += 1  
 else:  
 array[k] = M[j]  
 j += 1  
 k += 1  
  
 # When we run out of elements in either L or M,  
 # pick up the remaining elements and put in A[p..r]  
 while i < len(L):  
 array[k] = L[i]  
 i += 1  
 k += 1  
  
 while j < len(M):  
 array[k] = M[j]  
 j += 1  
 k += 1  
  
  
# Print the array  
def printList(array):  
 for i in range(len(array)):  
 print(array[i], end=" ")  
 print()  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 array = [86, 9, 19, 2, 1, 73]  
  
 mergeSort(array)  
  
 print("Sorted array is: ")  
 printList(array)

1. **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?**

Generators are iterators but are considered to be a more elegant and effective way of coding. It is used to save a lot of space as well. Using a generator will allow us to iterate over a returned object in a loop. The use case of the generator is that we are able to loop through a large amount of data that is used before or after a function does not need to be stored. They are particularly useful when using huge data sets which will take up a large amount of memory during the processing or cleaning of the data.

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

A decorator is a function that takes another function as an argument and returns a modified version of it.

A good example of a decorator includes logic functionalities, for example requesting users to log in or granting access to a website. Also, a @route decorator is a very common example.

@route(“/home”)

Def home():

**function logic**