**WEEK 1**

Answer the following questions individually. Once you have completed this task, you will collaborate with a group of 3-5 students to create a presentation on "Object-oriented programming, four OOP principles, and the history of Python".

1. What is Computer programming?

Computer programming is a sphere in science where programmers write a source code that, in the case of keeping the right syntax, will be turned into a machine language. This machine language will give Computers an instruction to behave.

<https://www.freecodecamp.org/news/what-is-programming/>

2. What are the fundamental concepts of computer programming?

There are multiple fundamental concepts in programming. These include:

* Variables and Data types, which are basically variables that can store different types of data such as text, number or boolean. These can be adjusted or kept static.
* Control structures, which are different operations that can regulate the program after execution. The examples for those are For, While, and Break.
* OOP (question 7)
* Algorithms and Data-structures. Algorithms are pieces of code that create a set of actions for computers to do, and Data-structures are formats for storing and changing data efficiently.

<https://www.alooba.com/skills/concepts/programming/programming-concepts/#:~:text=Programming%20Concepts%2C%20also%20known%20as,efficient%2C%20readable%2C%20and%20maintainable%20code>

3. What are the different types of programming languages?

Programming languages can be divided by many different categories. The most basic category is the Level of a language. Low level programming languages (assembly) are closer to machine language, and high level languages are closer to human language (python). Programming languages also divide into different types:

* Procedural languages ©, where every action is executed step by step.
* Object Oriented Languages, where the main concept is objects and classes, which encapsulate properties and behaviour in them. These are Java, Python and others.
* Functional Languages, which are based on mathematical functions and can be efficient in parallel programming and analysis.
* Scripting Languages (JS), which are easy programming languages. These are often used in web development and server-based programming.
* Domain-Specific languages. These are programming languages that are more deeply adjusted for specific areas, such as web-development (HTML), mathematical equations (MATLAB) and database management (SQL).

<https://www.prepbytes.com/blog/general-programming/types-of-programming-language/>

4. What are the characteristics of high-level and low-level programming languages?

High level programming languages are closer to Human language, which means that they are more readable and abstract, which means that many actions are covered within low amounts of operations. They are also easy to port from one platform to another, because they are platform-independent.

Low level programming languages are much closer to Machine language. They directly work with hardware, which increases their speed and efficiency. They are also often made for specific platforms/hardware.

5. What is machine language?

Machine language is the primary language to work with computers. It consists of instructions and binary numbers which are 0 and 1, which represent other digits, letters and other.

<https://www.sciencedirect.com/topics/engineering/machine-language#:~:text=Machine%20language%20is%20the%20language,little%20bit%20easier%20to%20read>

6. What is assembly language?

Assembly is a low level programming language. It works mostly directly with the hardware. Assembly can only include one instruction in a line of code. Programs written in Assembly are then compiled by Assembler, which is an instrument for converting low level code into high level one.

<https://medium.com/sololearn/what-is-assembly-and-how-is-it-used-7a1944431952>

7. What is object-oriented programming?

Object oriented programming is a type of programming languages/programming where everything is organised for maximum efficiency. These languages often consist of Classes and Objects.

<https://www.coursera.org/articles/object-oriented-programming-languages>

8. What are the different concepts in object-oriented programming and explain them?

* Abstraction: Code is made simpler, so people can focus on the final destination of the program, and not the processes that happen in between.
* Encapsulation: Objects in OOP serve as storages of variables and also can produce different actions.
* Inheritance: Classes can be elongated to create subclasses, for example: person -> corpo employee -> HR employee. This allows to use code more efficiently and to organise it, so it will be easier to navigate.
* Polymorphism: Objects in OOP can take any variables (if permitted), which allows to have millions of different variants and outcomes.

9. What is the difference between procedural and object-oriented programming?

While Object Oriented programming is based on creating objects and classes, Procedural programming is based on sequential actions. Because of this, Procedural programming does not give an option to encapsulation, inheritance and polymorphism.

10.Why is it important to understand syntax and semantics?

Lack of syntax understandment will lead to often errors and to the disability to write some pieces of code. Lack of Semantics might lead to incorrect outcomes of code or error, when a person, for example, mixes up some objects in code.

11.What is the difference between compiled and interpreted programming languages?

Compiled programming languages translate the code into machine language before executing it, whilst Interpreted languages use an interpreter, which transforms the code in real time during the execution. Compiled languages can be faster during the execution and more efficient in terms of memory and CPU usage, but they take more time at the start, with the translation process. Compiled languages also can be platform dependent, while interpreted languages might not have the same problem.

<https://www.freecodecamp.org/news/compiled-versus-interpreted-languages/>

**WEEK 2**

1. What is computational thinking?

Computational Thinking is a type of thinking made to solve problems or to understand different things. Computational thinking involves breaking down a problem into smaller pieces that will be easier to manage. It consists of 4 main principles.

2. The four cornerstones of computational thinking?

Decomposition: separating one problem into multiple smaller problems step by step.

Abstraction: simplifying the problem and getting rid of anything that is not relevant.

Pattern Recognition: searching ways to solve a problem based on previous solved problems that you have seen or done.

Algorithm: creating a step-by-step solution for a problem.

3. How to apply computational thinking to problem solving?

Computational thinking is highly useful for problem solving. To apply it, start with decomposing the task into smaller steps, with keeping the abstraction in unnecessary information. While decomposing a problem do not forget to find associations if you have any. At the end create a step-by-step solution.

**WEEK 3**

Part 1:

1. You are organising a charity event, presenting your steps in a hierarchical structure using the top-down and bottom-up approaches.

For the bottom up approach we will need to do this:

* Take money
* Divide spendings into Food, Decorations and Entertainment
* Buy all needed items
* Start the Charity event

For the top down approach we will need to do this:

* Decompose the event into smaller parts
* Find the root items needed

1. In your words, explain the benefits and drawbacks of the top-down and bottom-up approach.

In the bottom up approach, solutions to the problems come from the solutions to sub-problems, while in the top-down approach we find the solution naturally using recursion. The names speak for themself.

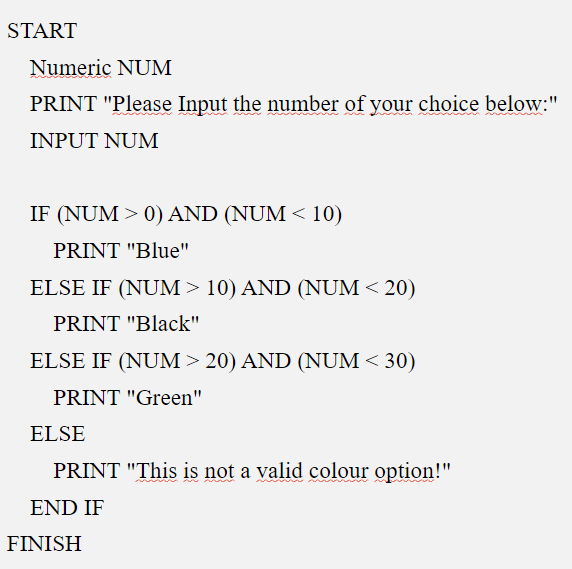
Part 2:

1. What is pseudocode?

Pseudocode is a piece of step-by-step instructions written in a programming manner, where everything is highly abstracted. It is meant to be understood by any programmer and be implementable into most programming languages.

1. Why is Pseudocode important?

Pseudocode is a crucial step for programmers that are working on big projects. It helps for them to plan out the code and have a better understanding of it before writing a code.



**WEEK 4**

1. What is a Flowchart?

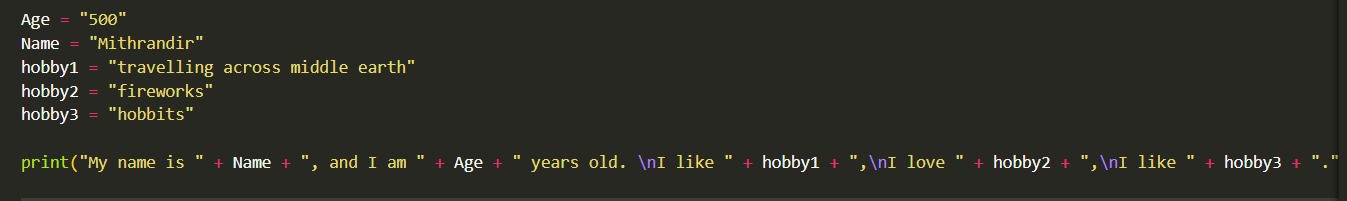
A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

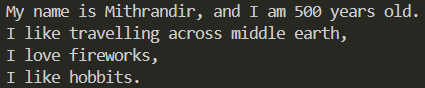
1. What are the benefits of flowchart?

* Making a complex process more easily to understand and management ​
* Helping reader understand the LOGIC of the code​
* Finding the problem directly and maintain them or making some optimise​
* It also can be a tool for a team members understand their target

**WEEK 5**

1. Write 5 facts about you in python (and also question 4)



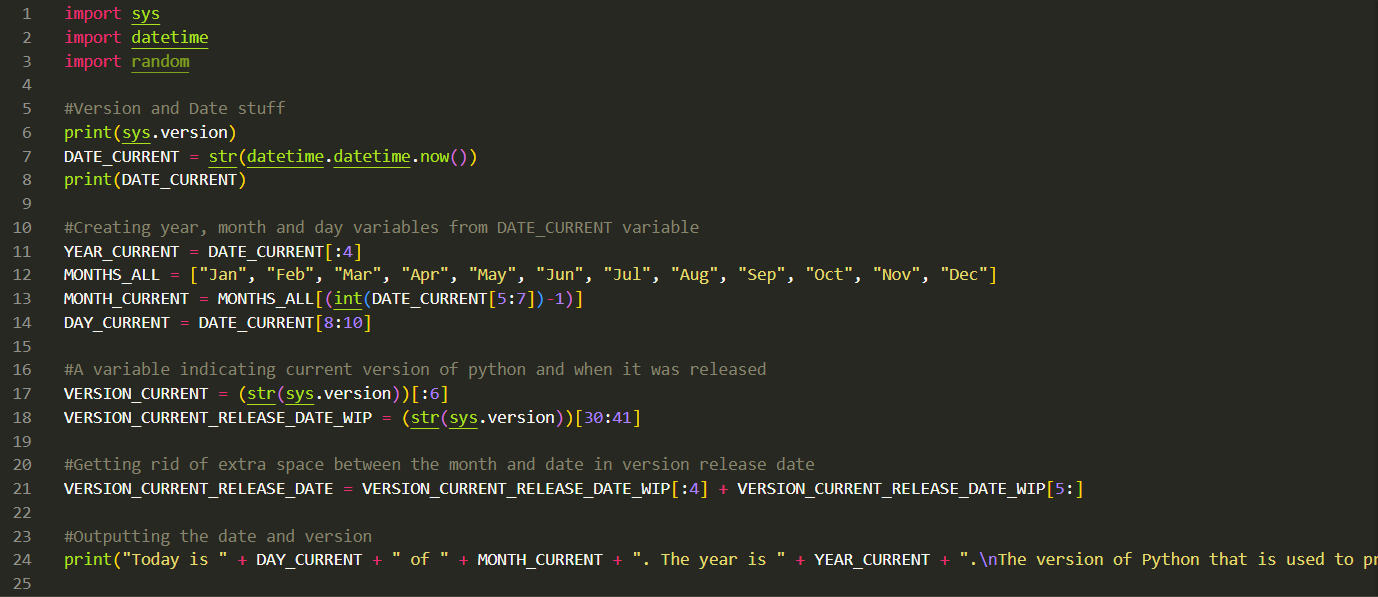


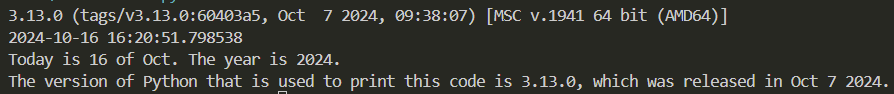
5:







6. 



This code is not applicable if the version update will include at least one new letter in this part:

