



TRAINEE SOFTWARE MODULE (2026)

Ubuntu 22.04 Dual Boot:(Dec 26)

To start with this module, we want you to dual-boot **Ubuntu 22.04** first.

Follow [this](#) doc to dual boot your system.

Now that you have Ubuntu (working preferably), no more hunting through menus or clicking through windows, you can manage files, and install software without leaving the command line.

Go through the following tutorial to get used to basic Linux commands:

<https://www.freecodecamp.org/news/the-linux-commands-handbook/>

Checkpoint:

After this, you should be comfortable using the following commands:

`cd , ls , pwd , sudo , mkdir , rm , mv , cp , cat , echo , chmod`

Github (Dec 27 – Dec 28):

If you already have the knowledge of how *git* works, good!
Else, no issues, we have got you covered.

Follow this [tutorial](#) to know how to use git from your terminal.

You might also want to check out this cheat sheet:

https://github.com/mikeizbicki/ucr-cs100/blob/2015winter/textbook/ch_eatsheets/git-cheatsheet.md

Checkpoint:

Try out this fun git game that walks you through using git commands.
Doing it up to level 6 is recommended, above that is up to your enthusiasm.

<https://github.com/git-game/git-game>

Python & Required Libraries (Dec 28 - Dec 29)

Python is the programming language we majorly use.

It is suggested to use IDEs like VSCode, you can follow this video for installing:

▶ How to Install and Use Visual Studio Code on Ubuntu 20.04 LTS Li...

Those who are new to python can refer to -

<https://www.w3schools.com/python/>

Those familiar with python can refer to -

1. <https://www.w3schools.com/python/numpy/default.asp>: Numpy
2. <https://www.w3schools.com/python/pandas/default.asp>: Pandas
3. https://www.w3schools.com/python/matplotlib_intro.asp:
Matplotlib

Checkpoint:

Python will be used extensively so continue to practice if you are unfamiliar with it.

Solve the below Problem Statement and add a screenshot of the output in your documentation:

Problem Statement:

*You are given a CSV file (**cones.csv**) containing unordered 2D positions of blue and yellow cones that represent the left and right boundaries of a closed race track. Using **only NumPy, pandas, and Matplotlib**, reconstruct the track centerline by estimating the overall track center, sorting the cones angularly around this center, computing midpoints between corresponding blue-yellow cone pairs, and forming a continuous closed-loop trajectory with a consistent direction of travel. Visualize the cones and the generated centerline, animate a particle moving along the trajectory, and export the animation as an MP4 file to validate the result.*

Use this csv file: [cones.csv](#)

C++ and Features(Dec 30 - Dec 31)

Python has been the main programming language used so far. We are now shifting to C++ because it offers better performance, faster execution, and more control over system resources.

Some of you might have already got a hang of C++ in your CS101, if not you can refer to [this](#) to brush up your existing knowledge :)

<https://www.geeksforgeeks.org/cpp/c-plus-plus/>

Get familiar with the below topics:

1. Basics of CPP - brush through the basics of CPP quickly, include major topics: Data Types, Control Flow statements, Loops, Functions and Arrays
2. Core concepts - Pointers and References, Namespaces
3. Recursion: Refer [this](#) to revise about recursion
4. OOP (Object Oriented Programming) - Classes and Objects, Inheritance, Polymorphism, Encapsulation

Checkpoint:

Solve the below problem statement and add a screencast of the output in your documentation:

Problem Statement:

Design and implement a terminal-based Snake game in C++ for Linux using object-oriented programming and the ncurses library for rendering and keyboard input. The game should feature real-time control of the snake using arrow keys or WASD, random food spawning on free cells, snake growth and score increment when food is eaten, and game termination on collision with walls or the snake's own body. The design must include clear classes such as Game (overall control and loop), Snake (body and movement), Food (position and respawn), and optionally Board (dimensions and drawing), with proper encapsulation via private data members and public methods like update(), draw(), and handleInput(). The program should compile on Linux with g++ and link against the ncurses library (for example, g++

**.cpp -lncurses -o snake) and present a simple game-over screen showing the final score and an option to restart or quit.*

ROS (Jan1 - Jan5)

What is ROS? ROS (Robot Operating System) is a framework used in many robotic applications to build and interconnect code. It provides a set of libraries and tools that allow developers to create software for robots, making it easier to build, test, and deploy new robotic systems.

Use [this](#) resource for installing ROS Humble.

Set your locale, sources and install ros2 packages (install ros-desktop, not base). Set up your environment and try the example given.

Once you have ROS on your system, it's time to do some hands-on stuff. Go on to [tutorials](#) and complete the following ones to gain a basic understanding of it.

CLI tools:

- Configuring environment
- Turtlesim, ros2 and rqt
- Understanding nodes and topics
- Launching nodes

Client Libraries:

- Using colcon
- Creating a workspace and a package
- Writing a publisher and subscriber in Python

Checkpoint:

Handshake Assignment:

- Make 2 nodes: VCU and AI. VCU represents the vehicle control unit, and AI is the computer that processes our autonomous software.
- These nodes will use 2 topics to communicate: Name these topics as – “VCU2AI” and “AI2VCU”.
- Aim: VCU needs to check continuously if the computer (AI) is working or not. Devise a method to ensure this. You need to write blocks of codes in VCU and AI nodes to implement this method.

