



GROUP 13

SOFTWARE DEVELOPMENT TOOLS

WEEK 4: INTEGRATED DEVELOPMENT ENVIRONMENT
TOOL OF FOCUS: VISUAL STUDIO CODE



INTRODUCTION

VISUAL STUDIO CODE:

It is a code editor created for development tasks like writing, editing, task running, debugging code and version control.

FEATURES


- Supports multiple languages.
- Multiple extensions.
- Integrated terminal.
- Built-in Git Version Control.
- Debugging.

SETTING UP VISUAL STUDIO CODE

The following steps were followed to install Visual Studio Code;

1. Download:

- Go to the visual studio code website using your web browser and search for the set-up for the corresponding OS.



The screenshot shows the Visual Studio Code documentation website in a web browser. The address bar shows the URL `code.visualstudio.com/docs/setup/windows`. The page title is "Visual Studio Code on Windows". The left sidebar contains a navigation menu with links to Overview, SETUP, Overview, Linux, macOS, Windows (selected), Raspberry Pi, Network, Additional Components, Enterprise, Uninstall, GET STARTED, USER GUIDE, and SOURCE CONTROL. The main content area is titled "Visual Studio Code on Windows" and "Installation". It lists three steps: 1. Download the Visual Studio Code installer for Windows. 2. Once it is downloaded, run the installer (VSCodeUserSetup-(version).exe). This will only take a minute. 3. By default, VS Code is installed under `C:\Users\{Username}\AppData\Local\Programs\Microsoft VS Code\`. Below the steps, it says "Alternatively, you can also download a Zip archive, extract it and run Code from there." and a tip: "Tip: Setup will add Visual Studio Code to your %PATH%, so from the console you can type 'code .' to open VS Code on that folder. You will need to restart your console after the installation for the change". The right sidebar contains a section "IN THIS ARTICLE" with links to Installation, User setup versus system setup, Updates, Windows Subsystem for Linux, Next steps, and Common questions. At the bottom of the right sidebar are links to Subscribe, Ask questions, Follow @code, Request features, Report issues, and Watch videos.

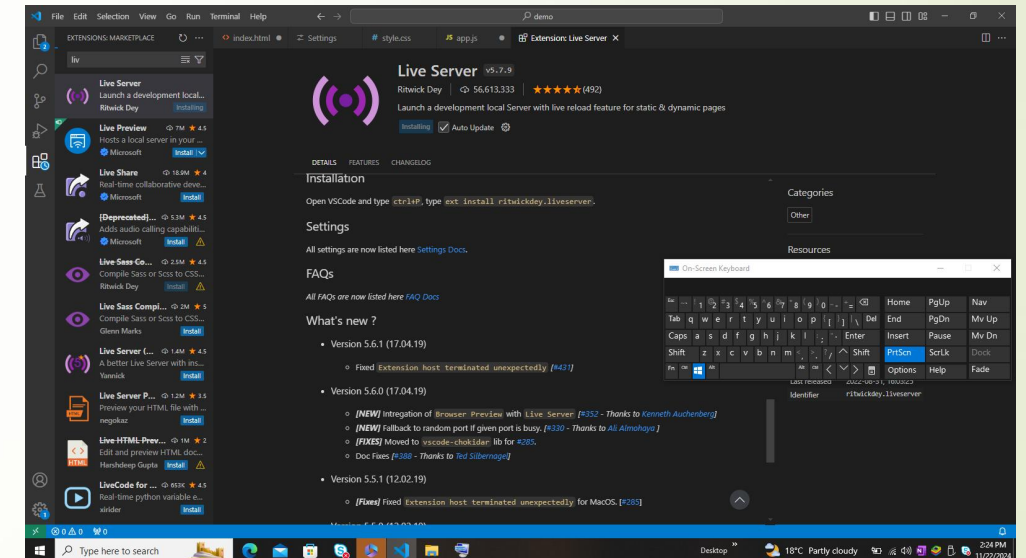
continuation

1. INSTALLATION:

- The installer was run and prompts were followed.
- VS Code was then launched after installation.

2) EXTENSIONS:

- They are software components which add new functionality in a VS CODE
- They provide language support, code formatting, debugging tools and more. They can be easily installed from the VS Code market space.
- Some extensions we installed were Python, Live Server, which act as a host and many more.



SETTING UP WORKSPACES

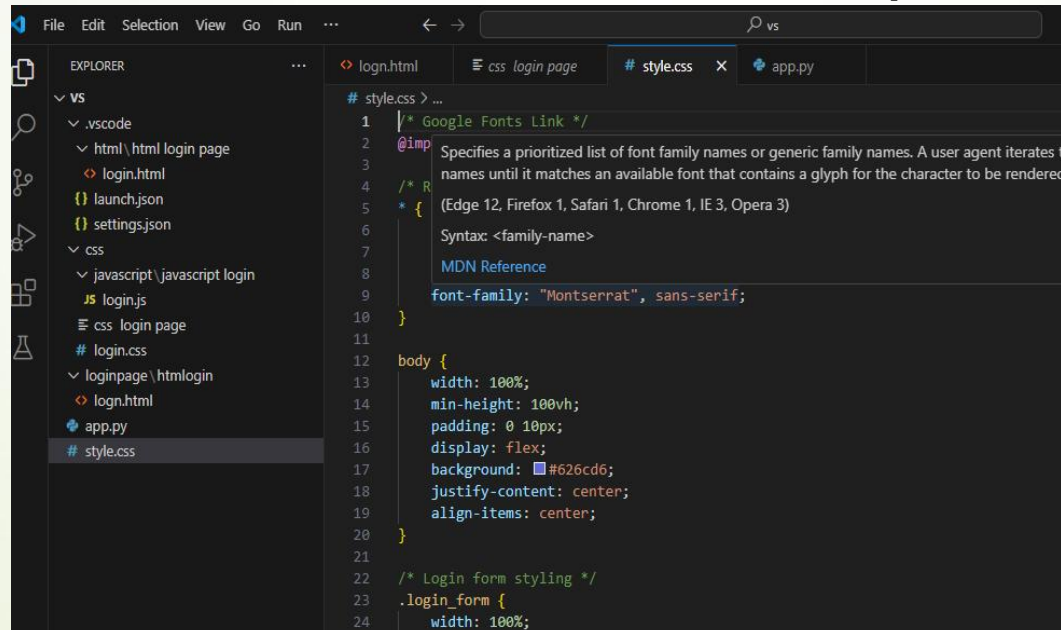
1. CREATING A WORKSPACE:

- **OPEN A FOLDER** : File>Open folder to create a new workspace
- **ADD MULTIPLE FOLDERS FOR THE VARIUOS TASKS** : File>Add folder to workspace.

2. SAVING THE WORKSPACE

After adding the workspace ,

- go to **File>Save Workspace As...**
- Choose location and save the **.code-workspace** file



The screenshot shows the Visual Studio Code (VS Code) interface. On the left, the Explorer sidebar displays a file tree for a workspace named 'vs'. The tree includes folders like '.vscode', 'html\html login page', 'css', 'javascript\javascript login', and 'loginpage\htmllogin'. Files listed include 'login.html', 'launch.json', 'settings.json', 'login.js', 'login.css', 'login.html', and 'app.py'. The main editor area shows the 'style.css' file, which contains CSS code for a login form. A tooltip is visible over the '@import' rule, providing a description and a link to the MDN Reference. The code in the editor includes a comment for Google Fonts Link, an @import rule for Montserrat font, and a body selector with flexbox styling.

```
# style.css > ...
1  /* Google Fonts Link */
2  @import url('https://fonts.googleapis.com/css?family=Montserrat');
3
4  /* Reset */
5  * {
6    margin: 0;
7    padding: 0;
8    box-sizing: border-box;
9  }
10
11  body {
12    width: 100%;
13    min-height: 100vh;
14    padding: 0 10px;
15    display: flex;
16    background-color: #626cd6;
17    justify-content: center;
18    align-items: center;
19  }
20
21
22  /* Login form styling */
23  .login_form {
24    width: 100%;
```



INSTALLING EXTENSIONS

Extensions are enhancements that improve functionality, and allow users to customize their development environment to better suit their needs.

SETTING UP EXTENSIONS

1. Open extension panel(Ctrl+shift+X)which can also be found on the left side bar.
2. Search for desired extension(Python, Javascript).
3. Select extension and click “install”.
4. Restart VS Code if required
5. Verify installation in Extension Panel.

SIMPLE PROGRAM USING PYTHON (BMI CALCULATOR)

- **INTRODUCTION:**

- **A BODY MASS INDEX CALCULATOR** is a measurement of a person's leanness or corpulence (A Person state of having much fat or not) based on their height and weight, and is intended to quantify tissue mass. It is widely used as a general indicator of whether a person has a healthy body weight for their height. Specifically, the value obtained from the calculation of BMI is used to categorize whether a person is underweight, normal weight, overweight, or obese depending on what range the value falls between. These ranges of BMI vary based on factors such as region and age, and are sometimes further divided into subcategories such as severely underweight or very severely obese. Being overweight or underweight can have significant health effects.

- Its calculated by Using formular

- **BMI = WEIGHT IN KG / HEIGHT SQUARE m**

Below are conditions for a person weight result from BMI

```
if bmi < 18.5:
    return "Underweight"
elif 18.5 <= bmi < 24.9:
    return "Normal weight"
elif 25 <= bmi < 29.9:
    return "Overweight"
else:
    return "Obesity"
```


Code screenshots + Output

The image displays a Visual Studio Code editor window with a Python script named `bmi.py` and its execution output in the terminal.

Code Editor:

```
1 def calculate_bmi(weight, height):
2     """Calculate BMI using weight and height."""
3     bmi = weight / (height ** 2)
4     return bmi
5
6 def get_bmi_category(bmi):
7     """Determine the BMI category."""
8     if bmi < 18.5:
9         return "Underweight"
10    elif 18.5 <= bmi < 24.9:
11        return "Normal weight"
12    elif 25 <= bmi < 29.9:
13        return "Overweight"
14    else:
15        return "Obesity"
16
17 def main():
18     # Get user input
19     weight = float(input("Enter your weight in kilograms: "))
20     height = float(input("Enter your height in meters: "))
21
22     # Calculate BMI
23     bmi = calculate_bmi(weight, height)
24
25     # Get BMI category
26     category = get_bmi_category(bmi)
27
28     # Display the result
29     print(f"Your BMI is: {bmi:.2f}")
30     print(f"BMI Category: {category}")
31
32 if __name__ == "__main__":
33     main()
```

Terminal Output:

```
PS C:\G13> & C:/Users/admin/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/G13/bmi.py
Enter your weight in kilograms: 64
Enter your height in meters: 1.62
Your BMI is: 24.39
BMI Category: Normal weight
PS C:\G13>
```

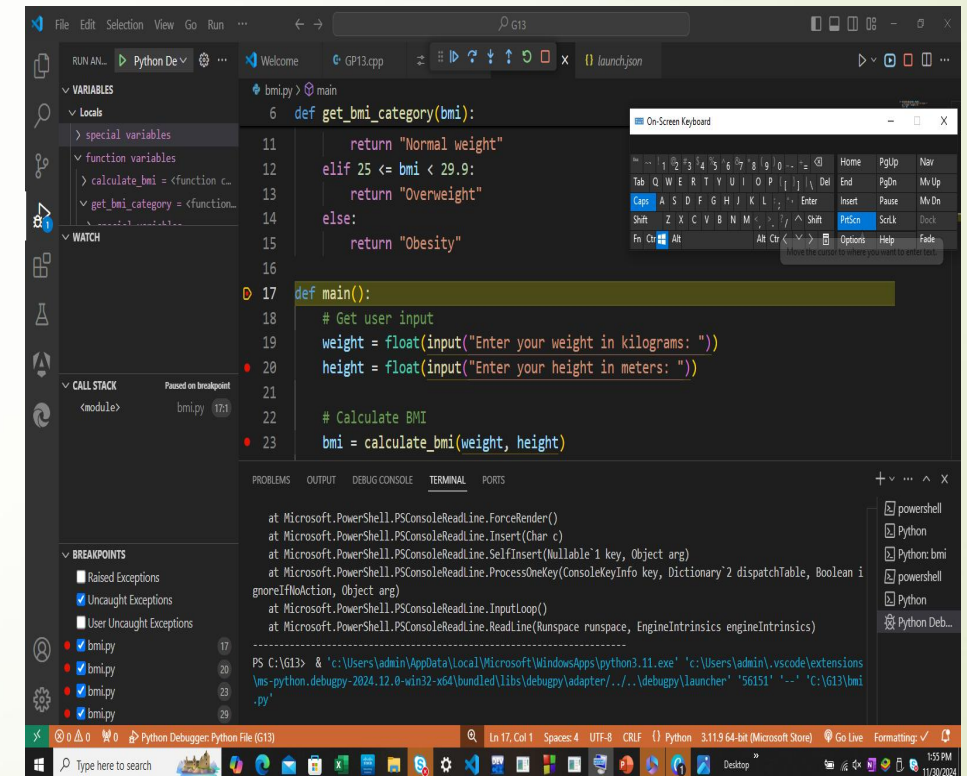
The terminal shows the script being executed from the command prompt, with the user providing input for weight (64 kg) and height (1.62 m). The output displays the calculated BMI (24.39) and the corresponding category (Normal weight).

DEBUGGING FEATURES

DEBUGGING STEPS

- DEFINITION OF DEBUGGING: IT is a process of identify and fixing errors in a software
- STEPS OF DEBUGGING OUR CODE
- We open debug view by ctrl+shift+D
- We selected the python file from drop down menu
- We set break points on line 17,20,23 and 29 and press f5 to debug
- We inspect the variables and use toolbar f10 and f11 in stepping over and into
- We check debug console for any output error

DEBUGGING IMAGE



SPRINT BACKLOG TASK IMPLEMENT

- We implemented a login page on our sprint backlog

