

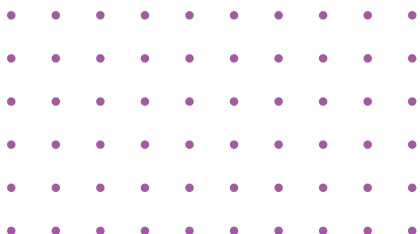
# WEEK 2 - PART 1

## PROGRAMMING ENVIRONMENTS -

## COMPONENTS, TOOLS, AND PLATFORMS



LINNAEUS  
UNIVERSITY  
SUMMER 2024



# TABLE OF CONTENTS

3

what is an environment?

4

key components - programming environments

operating systems ..... 4

programming languages ..... 4

runtime environments ..... 5

node.js ..... 6

CPython ..... 6

virtual machines - vm's ..... 6

7

development tools & environments

development tools ..... 7

development environments ..... 8

IDE - Integrated Development Environment ..... 8

Vscode ..... 10

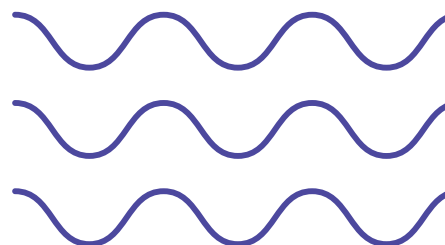
extensions ..... 11

packages ..... 12

npm ..... 12

pypi ..... 13

online IDE's ..... 14



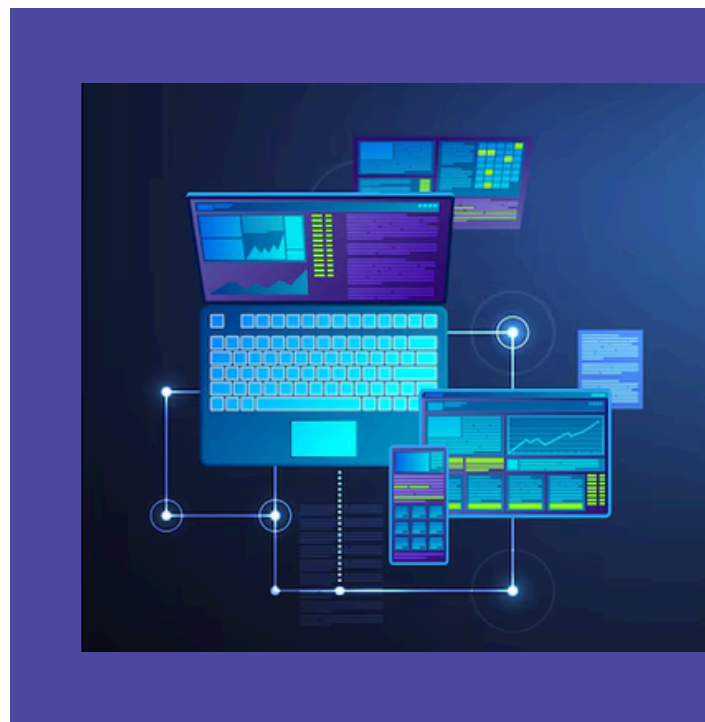
# WHAT IS AN ENVIRONMENT?

In the context of programming, the term "environment" often appears and can be challenging to grasp because it **can apply to various aspects and areas within programming**. For instance, if you are using a Mac computer, you are working in a MacOS environment. Similarly, if you are working in an Integrated Development Environment (IDE), that too is considered an environment.

Broadly, a **programming environment refers to the combination of the underlying hardware, software, and settings** in which a program is developed and executed. In addition to its more abstract meanings related to system configuration and context, "environment" can also refer to specific applications or software.

**The term "environment" encompasses everything necessary for the code to run correctly** and produce the desired results. Each software environment serves a distinct purpose.

Additionally, **different environments come with varying requirements, capabilities, and performance characteristics**, making it essential to choose the appropriate environment for a specific programming task.



# KEY COMPONENTS - PROGRAMMING ENVIRONMENTS

## OPERATING SYSTEMS

The **operating system serves as an environment** in computing by managing computer hardware and offering underlying essential services for other software programs. It oversees critical functions, thereby **providing a stable platform** where applications can operate and interact with hardware efficiently. Operating systems are responsible for managing the **entire computing environment**, including system resources, user interfaces, and application execution. They provide a broad and **general-purpose** environment that supports all types of applications and services on a computer - including other environments.

## PROGRAMMING LANGUAGES

**Programming languages are typically not directly executable by a computer's hardware.** Instead, they require a compiler or an interpreter to translate the code into machine-readable instructions.



A compiler translates the entire program into machine code all at once, while an interpreter executes the code line by line, translating it as it runs.

Each programming language has its own syntax and rules, which determine how code is written and executed. **To start coding in a specific programming language, you'll need to download and install the appropriate environment for that language.** Additionally, you may need to download libraries and frameworks specific to that language.

## RUNTIME ENVIRONMENTS

**A runtime environment provides the necessary infrastructure, tools, and services for executing a specific application or running code in a specific programming language.**

Runtime environments are often specific to a programming language or platform. They abstract the complexities of the underlying hardware and offer a standardized platform for developers to write and run their applications. **Runtime environments typically include components** such as libraries, interpreters, and virtual machines that are required to run programs written in a particular language.

**A runtime environment abstracts low-level details** (more advanced features that operate in the background for software to run properly) and provides a consistent execution environment for programs written in a particular language. It manages technical tasks necessary for the program's operation, **allowing developers to focus on writing code rather than system-level operations.**

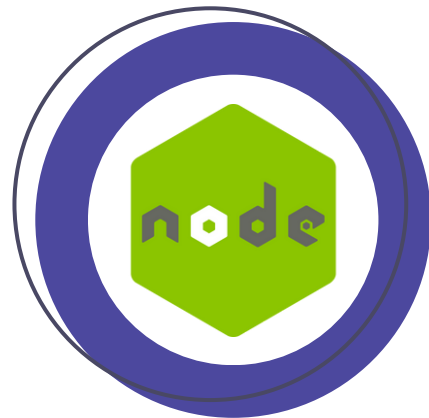
It handles all the essential behind-the-scenes tasks required for the program to function properly, which may not be visible but are crucial for its operation.



Runtime environments also provide a **consistent execution environment**, meaning they offer a reliable and predictable setting for your code to run. Regardless of whether your program is executed on someone else's computer or on a server anywhere in the world, the runtime environment ensures that it **behaves consistently across all these systems and configurations.**

For example, if you're working with **JavaScript**, you'll need to set up the appropriate development environment for the language.

This typically involves **installing Node.js**, which provides the JavaScript runtime environment necessary for executing JavaScript code outside of a web browser.



Similarly, **Python** requires you to install a Python interpreter to execute Python scripts, with **CPython being the most common** and widely used runtime environment.



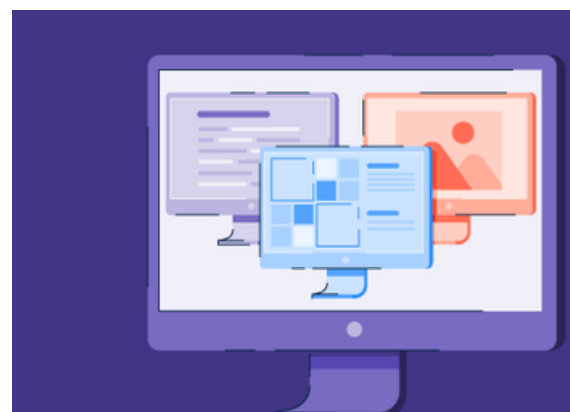
These components collectively provide the necessary tools and infrastructure to write, test, and execute your code effectively on your computer.

## VIRTUAL MACHINES - VM'S

**A virtual machine is a software emulation of a physical computer system.**

It allows you to **run multiple operating systems or environments within a single physical machine.**

Each virtual machine operates independently and includes its own operating system along with all the code necessary for its operation.



This means that the computer you're working on has its own operating system, and **the virtual machine has an additional operating system running within it.**

While it tends to take up more memory and repeat code, it's very useful for testing and isolating different environments. While it tends to use more memory and can duplicate code, it is very **useful for testing and isolating different environments.**

By utilizing virtual machines, users can **quickly create and discard multiple isolated environments**, reducing the risk of software conflicts and enabling safe experimentation with new configurations.

---

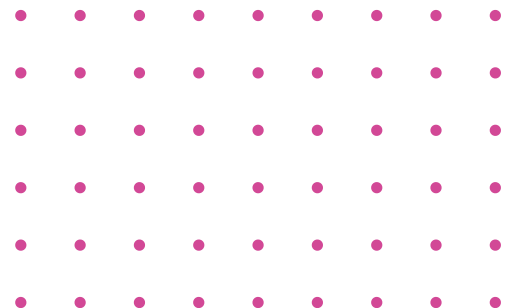
# DEVELOPMENT TOOLS & ENVIRONMENTS

## DEVELOPMENT TOOLS



**Development tools are software applications and utilities designed to aid** in the creation, testing, debugging, and management of software applications. While development tools are essential components of development and runtime environments, **they are not all considered environments themselves.**

Development tools are used within these environments to facilitate various stages of the software development lifecycle.



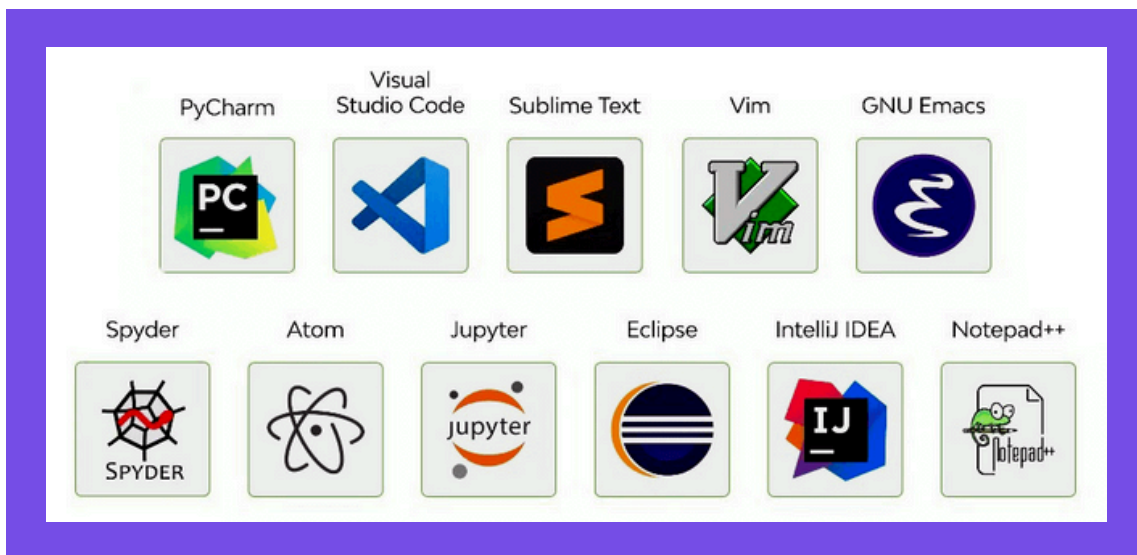
# DEVELOPMENT ENVIRONMENTS

A development environment is a comprehensive setup that includes all the tools, libraries, and configurations required for software development.

It is a general term, encompassing, among other things, integrated development environments (IDEs), code editors, debuggers, compilers, and other essential tools. This environment provides everything needed to write, test, and debug code, offering a complete and cohesive platform for developers to create software efficiently.

## IDE - INTEGRATED DEVELOPMENT ENVIRONMENT

An Integrated Development Environment (IDE) is software that works as a centralized platform providing a comprehensive set of tools and features for software development. Think of an IDE as a word processor for code, with far more functionality than just writing code.



IDE's provide a **code editor** with features such as **syntax highlighting**, which helps with writing in a specific programming language; **auto-completion**, which speeds up coding by suggesting possible code completions; **code cleanup**, which helps in organizing and removing unnecessary code; and **code formatting**, which ensures consistent and readable code structure.

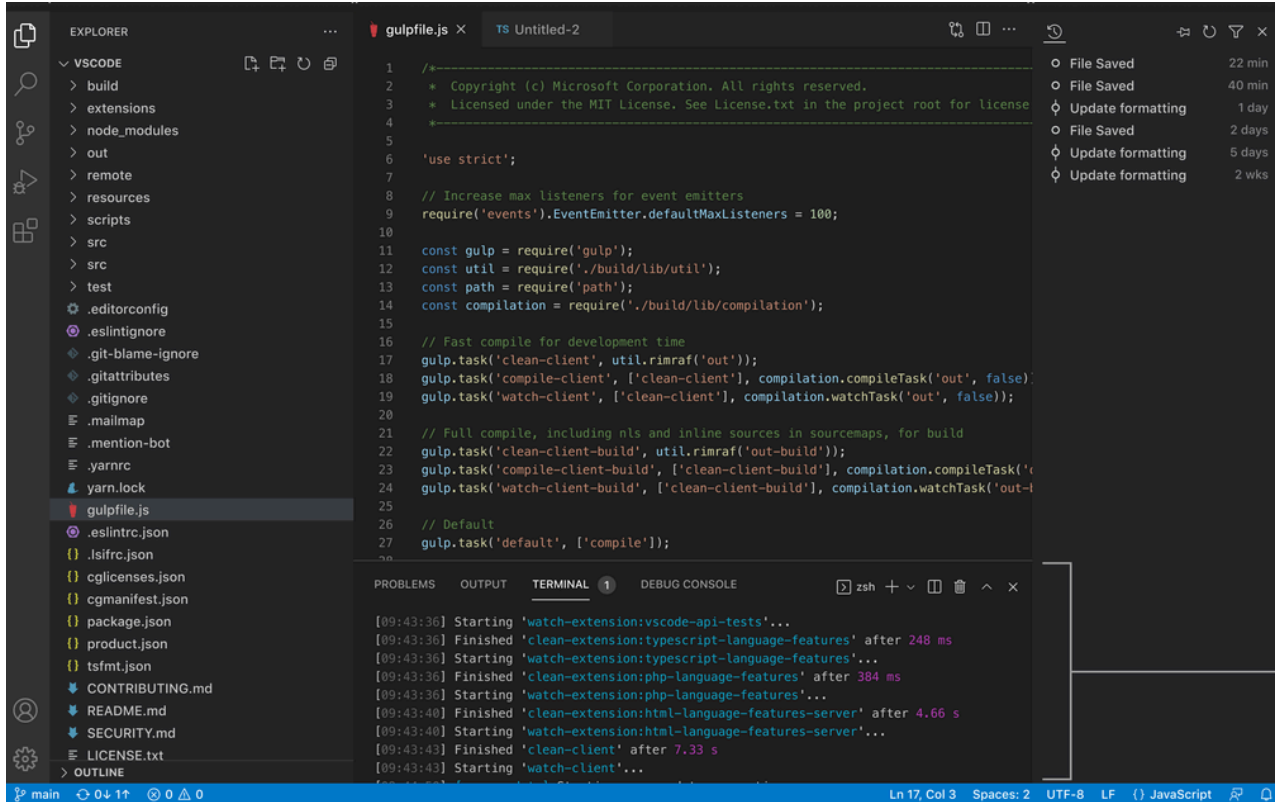


IDEs facilitate project organization by allowing developers to **create and manage projects with multiple files and folders**. They provide tools for creating new files and folders and for navigating between different parts of the project.

**They often visually represent the actual folder structure**, offering a visual method for developers to navigate and organize their files and folders within the IDE.

IDEs also include tools for automating application builds and deployments. **Deployment refers to delivering code to a platform** where users can access the software. IDEs can help compile code, package applications, and deploy them to specific environments or platforms.


IDEs often include built-in tools to compile source code into executable formats. This is essential for languages that require compilation, such as Java. **Compilation translates human-readable code into machine code that can be run on computers.**



The screenshot displays the Visual Studio Code interface. On the left, the Explorer sidebar shows a project structure with folders like 'build', 'extensions', 'node\_modules', 'out', 'remote', 'resources', 'scripts', 'src', and 'test', along with various configuration files. The main editor area shows the 'gulpfile.js' file with the following code:

```
1 /*
2  * Copyright (c) Microsoft Corporation. All rights reserved.
3  * Licensed under the MIT License. See License.txt in the project root for license
4  */
5
6 'use strict';
7
8 // Increase max listeners for event emitters
9 require('events').EventEmitter.defaultMaxListeners = 100;
10
11 const gulp = require('gulp');
12 const util = require('./build/lib/util');
13 const path = require('path');
14 const compilation = require('./build/lib/compilation');
15
16 // Fast compile for development time
17 gulp.task('clean-client', util.rimraf('out'));
18 gulp.task('compile-client', ['clean-client'], compilation.compileTask('out', false));
19 gulp.task('watch-client', ['clean-client'], compilation.watchTask('out', false));
20
21 // Full compile, including nls and inline sources in sourcemaps, for build
22 gulp.task('clean-client-build', util.rimraf('out-build'));
23 gulp.task('compile-client-build', ['clean-client-build'], compilation.compileTask('out-build', false));
24 gulp.task('watch-client-build', ['clean-client-build'], compilation.watchTask('out-build', false));
25
26 // Default
27 gulp.task('default', ['compile']);
```

At the bottom, the Terminal window shows the output of a gulp command, including timestamps and task names like 'watch-extension:vscode-api-tests...', 'clean-extension:typescript-language-features', 'clean-extension:php-language-features', 'clean-extension:html-language-features-server', and 'clean-client'.



**IDEs also help in packaging the compiled code and related resources (like configuration files and libraries)** into a deployable format. This might involve creating executable files, application archives (such as .zip or .tar files), or installers.

The structure of IDEs—**a single program with a unified interface** combining different tools and features into one easy-to-use screen – helps facilitate programming in one central place.

**There are many different IDEs**, some general purpose, and some more specialized for a specific language. There are even online IDEs, where you can code without setting up runtime environments or downloading the necessary development tools and environments!



## VSCODE



**VSCode is a very popular IDE**, widely used due to its extensive features and customizability, enhancing functionality and tailoring the development environment to individual needs.

VSCode also includes a built-in terminal that functions like any other terminal.

While it appears integrated into VSCode's unified interface, you are still actually working with the terminal on your computer.

Most tutorials on YouTube use VSCode, and it is **widely regarded as a staple in the programming community**.

In Visual Studio Code (VS Code), you can choose which terminal to use directly within the integrated interface. For example, you can open PowerShell inside VS Code; while it appears seamlessly integrated into VS Code's unified interface, you are still operating the computer's terminal.

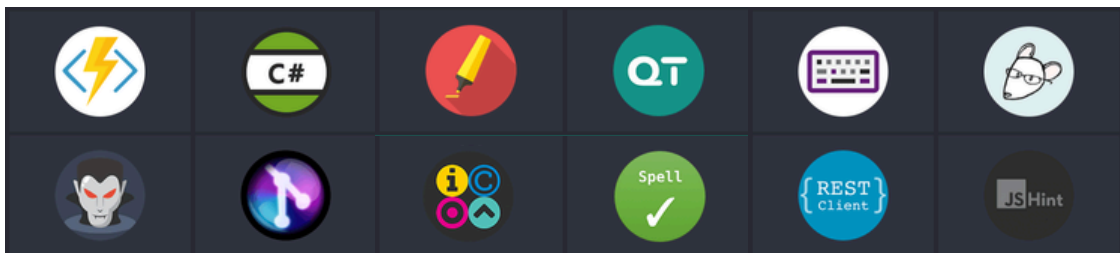
# EXTENSIONS

**VS Code, as it is, comes as a blank slate with only basic functionalities. To customize the user experience, users can download extensions. These small add-ons enhance or modify the application's functionality, expanding the editor's capabilities beyond its out-of-the-box features.**

Extensions integrate with VS Code, enabling the addition of new features, tools, themes, and more to customize and **tailor the development environment** to specific needs. Extensions can be simple, like **changing the look of the interface with themes and icons**, or complex, like adding tools that boost productivity.

In a code editor like Visual Studio Code, **extensions can add support for new programming languages**, provide debugging tools, or improve the user interface. For example, you might add a dark mode theme (vanity extension) or a Python language support tool (useful extension).

One example is Dracula, which changes the UI coloring of the program. **Extensions help users adapt the software to their specific needs and tastes**, making development easier and more efficient.



These extensions are typically distributed through the Visual Studio Code Marketplace, where users can search, browse, and install a wide variety of extensions. **Within the VSCode interface, you can access the Marketplace** and search for specific extensions, read reviews, and install them with just a click, seamlessly integrating new features into your editor.

These extensions are **not included by default** to prevent issues such as Python users having extensions that are only relevant for JavaScript. This approach helps VS Code **run faster and smoother**, while **still giving users the flexibility to customize their experience.**

# PACKAGES

**A package in software development is a bundled collection of code, assets, and metadata that provides specific functionality**

**or addresses particular issues faced by developers, that can be easily distributed and installed.**

Packages can be libraries or modules that developers include in their projects to extend functionality, such as adding new features, utilities, or tools. Think of packages as similar to extensions: they enhance your programming experience by adding functionality and solving problems without altering the core features of an environment.

By leveraging these packages, developers can improve their workflow, save time, and benefit from the expertise of the broader programming community.

When exploring packages, check the weekly download statistics for each package to assess its popularity and reliability. Remember, these are third-party packages, so a large and active community often signifies that a package is well-regarded and widely used.

## NPM



**npm, which stands for Node Package Manager, is a package manager designed specifically for the JavaScript programming language and the Node.js runtime environment.**

It enables you to download JavaScript packages that assist with various programming tasks. These npm packages are collections of code, assets, and metadata that **provide specific functionalities or address particular development needs**, much like extensions. npm helps extend the capabilities of Node.js while keeping the core environment simple and efficient.



**npm automates the processes of installing, updating, configuring, and removing software packages**, streamlining these tasks on your system.

Developers can explore, install, and leverage a wide range of npm packages to enhance their JavaScript projects, improve their workflow, and tap into the expertise of the broader JavaScript community. When assessing npm packages, **checking weekly download statistics and community feedback** can help determine their popularity and reliability. npm thus extends the capabilities of Node.js while maintaining a simple and efficient core environment.

## PYPI

**Python provides a dynamic array of packages and modules through PyPI, the Python Package Index.**

**PyPI serves as a package manager tailored for the Python programming language, allowing you to download packages that support various programming tasks.**



Like npm packages, PyPI packages consist of code, assets, and metadata designed to deliver specific functionalities or resolve particular development challenges, similar to extensions.

PyPI simplifies the management of software packages by automating their installation, updating, configuration, removal. Developers can explore and install a diverse range of PyPI packages to enhance their Python projects, streamline their workflow, and benefit from the broader Python community's knowledge. Again, be careful to evaluate packages by considering their download statistics and community feedback.

# ONLINE IDE'S

Online IDEs are cloud-based platforms that offer a **comprehensive development environment directly accessible from a web browser.**

These platforms simplify the coding and development process by **eliminating the need to install and configure runtime environments on your local machine.**

With online IDEs, you can create a **user account and start coding immediately** without having to install any software or set up complex runtime environments. They provide a **fully-featured code editor with support for syntax highlighting, autocompletion, and code formatting across various programming languages.** Commonly supported languages include Python, JavaScript, Java, C++, Ruby, and more.

A significant advantage of online IDEs is their integrated console, which allows you to execute code and **view output directly within the browser.** This feature streamlines development by providing immediate feedback without the need for local installations or configurations.

Additionally, many online IDEs offer collaborative features that enable multiple users to write, run, and share code simultaneously. This makes them ideal for both individual projects and team-based development.

Overall, online IDEs provide a **convenient, web-based solution for coding, execution,** and collaboration, removing the need for local setup and simplifying the development process.

