pip, Package Management & colorama

AiCE Software Engineering Foundations — Python Programming (Advanced)

Python's standard library is powerful, but the global Python community has built hundreds of thousands of additional tools that extend Python far beyond what ships by default. This lesson opens pip — the tool that gives you access to all of them — and colorama, the first external library you'll actually use.

1. What Is pip?

pip stands for Package Installer for Python. It is a command-line tool that connects to the Python Package Index (PyPI) — a public repository at pypi.org containing over 500,000 Python packages — and downloads and installs them on your machine with a single command.

Think of PyPI as an enormous online market — like Jumia or Takealot, specifically for Python code. Developers from around the world publish their libraries there. You browse, find what you need, and pip delivers it directly to your Python environment.

pip comes pre-installed with Python 3. Confirm it is available on your system:

```
pip3 --version
```

Output will look something like:

```
pip 24.0 from /usr/lib/python3/dist-packages/pip (python 3.12)
```

> **Always Use pip3**
> *On Ubuntu, always use pip3 (the Python 3 version) rather than plain pip, for the same reason you use python3 rather than python — to ensure you are working with the correct version.*

2. Installing a Package

The basic pip install command is:

```
pip3 install package_name
```

For this unit and the culminating capstone project, you will install two libraries: colorama and tabulate. Install both now:

```
pip3 install colorama tabulate
```

You can install multiple packages in a single command by listing them separated by spaces. The output looks something like:

```
Collecting colorama
Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Collecting tabulate
Downloading tabulate-0.9.0-py3-none-any.whl (35 kB)
Installing collected packages: colorama, tabulate
Successfully installed colorama-0.4.6 tabulate-0.9.0
```

Once installed, a library is available to every Python program on your system. You import it exactly like a standard library module — using the import statement at the top of your file.

> **pip3 vs. pip**
> *On some systems, pip and pip3 point to the same thing. On others, pip points to Python 2's installer and pip3 to Python 3's. Since we always use Python 3, always use pip3 to be safe. If you get a 'command not found' error, try: python3 -m pip install package_name, which uses Python 3's pip directly.*

3. Managing Installed Packages

pip provides several commands for managing the packages on your system:

Command	What It Does
pip3 install package	Install a package
pip3 install package==1.2.3	Install a specific version
pip3 install --upgrade package	Upgrade an installed package to the latest version
pip3 uninstall package	Remove an installed package
pip3 list	Show all installed packages and their versions
pip3 show package	Show details about a specific installed package
pip3 freeze	List installed packages in requirements format
pip3 install -r requirements.txt	Install all packages listed in a requirements file

The requirements.txt file deserves special mention. In professional Python projects, every external dependency is listed in a file called requirements.txt — one package per line. When a new developer joins the project or you deploy to a server, a single command installs everything:

```
# requirements.txt
colorama==0.4.6
tabulate==0.9.0
requests==2.31.0
```

```
pip3 install -r requirements.txt
```

Standard Professional Practice

Creating and maintaining a requirements.txt file is standard professional practice. Every project you build from this point forward should have one.

Building on Shared Work

"The Python ecosystem is one of the largest and most collaborative in software history. Every library you install represents thousands of hours of work donated by

> *developers who built something useful and shared it with the world. Use these tools. Contribute back when you can." — Adesina Olagunju, Founder, AiCE*

4. colorama — Adding Colour to Terminal Output

colorama is a library that lets you print colored text in the terminal. By default, all terminal output is the same color — usually white or green on a dark background. colorama lets you highlight important information, distinguish between success and error messages, and make your programs more readable and professional.

How colorama Works

colorama provides ANSI escape codes — special sequences of characters that terminals interpret as color and style instructions. It wraps these codes in easy-to-use constants so you don't need to remember the codes yourself.

Fore — controls the text (foreground) color: Fore.RED, Fore.GREEN, Fore.YELLOW, Fore.BLUE, Fore.MAGENTA, Fore.CYAN, Fore.WHITE

Style — controls text style and resetting: Style.BRIGHT (bold), Style.DIM, Style.RESET_ALL (returns to default color)

On Windows, colorama requires calling colorama.init() before use. On Linux and macOS (including Ubuntu), ANSI codes work natively — but calling init() anyway makes your code portable across all operating systems.

Your First colorama Program

```
import colorama
from colorama import Fore, Style

# Initialise colorama (important for cross-platform
compatibility)
colorama.init()

# Basic colour output
print(Fore.GREEN + "This text is green." + Style.RESET_ALL)
print(Fore.RED + "This text is red." + Style.RESET_ALL)
print(Fore.YELLOW + "This text is yellow." + Style.RESET_ALL)
```

```
# Bold / bright text
print(Style.BRIGHT + Fore.GREEN + "Bold green text." +
Style.RESET_ALL)

# Using f-strings with colorama
name = "Amara Diallo"
score = 88
print(f"{Style.BRIGHT}{Fore.CYAN}Student
Report{Style.RESET_ALL}")
print(f"Name : {Fore.WHITE}{name}{Style.RESET_ALL}")
print(f"Score: {Fore.GREEN}{score}{Style.RESET_ALL}")
```

Always Reset After Colouring

Every time you apply a colour or style with colorama, add Style.RESET_ALL at the end. Think of it as turning the colour off after turning it on. If you forget, your entire program output remains coloured — almost never what you want.

Building a Colour Helper Function

Rather than writing Fore.GREEN + text + Style.RESET_ALL every time, build a reusable helper function — this is the pattern used in the Student Management System capstone:

```
import colorama
from colorama import Fore, Style
colorama.init()

def colour(text, colour_code, bold=False):
    """Return text wrapped in the given colour code."""
    style = Style.BRIGHT if bold else ""
    return f"{style}{colour_code}{text}{Style.RESET_ALL}"

# Convenience wrappers
def green(text, bold=False): return colour(text, Fore.GREEN,
bold)
def red(text, bold=False): return colour(text, Fore.RED, bold)
def yellow(text, bold=False): return colour(text, Fore.YELLOW,
bold)
```

```
# Using the helpers
print(green("Success: Student enrolled successfully.",
bold=True))
print(red("Error: Student ID already exists."))
print(yellow("Warning: Score is below passing mark."))
```

These helper functions encapsulate the colorama logic cleanly — exactly the pattern used throughout the Student Management System later in this program.

Color Conventions in Terminal Applications

Color	Convention	Example Use
Green	Success, positive results, passed	Student enrolled, payment processed, PASS
Red	Errors, failures, warnings	Record not found, invalid input, FAIL
Yellow	Warnings, caution, neutral info	Score near passing mark, low stock warning
Cyan	Information, headers, prompts	Menu headers, loading messages
White	Normal data, standard output	Student names, scores, general content
Bold	Emphasis, important values	Final totals, grade letters, headings

Key Takeaways

pip3 install package_name connects to PyPI and installs any of over 500,000 published packages, and pip3 freeze together with a requirements.txt file is how professional projects document and reproduce their exact dependencies. colorama wraps ANSI escape codes in Fore. and Style. constants for cross-platform terminal color, and Style.RESET_ALL after every colored string is a habit worth building immediately. The next lesson introduces tabulate for formatted tables, virtual environments for isolating project dependencies, and closes with a capstone that combines colorama and tabulate into one polished student report.