Demonstration

Rohan Alexander

8 September, 2025

Introduction

`Python` is a general-purpose programming language created by Guido van Rossum. `Python` version 0.9.0 was released in February 1991, and the current version, 3.13, was released in October 2024. It was named `Python` after *Monty Python's Flying Circus*.

`Python` is a popular language in machine learning, but it was designed, and is more commonly used, for more general software applications. This means that we will especially rely on packages when we use Python for data science.

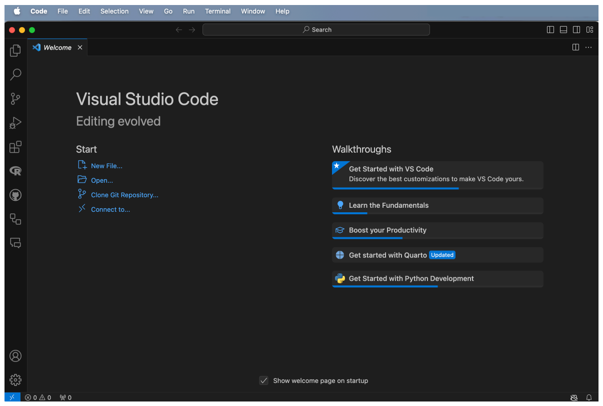
Python, VS Code, and uv

There are other options, but the community more broadly has settled on VS Code (you can pick other options if you have a good reason to do that). You can download VS Code for free here and then install it.

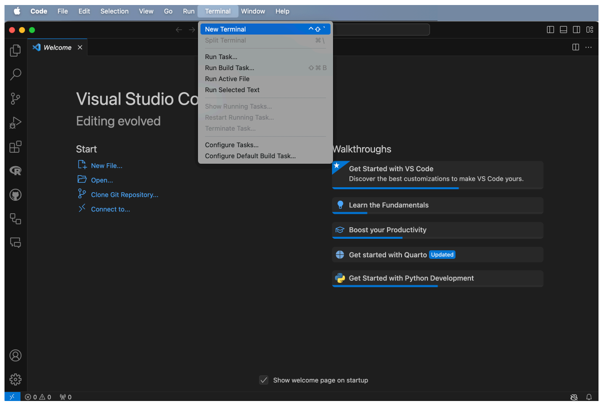
Open VS Code (Figure 1a), and open a new Terminal: Terminal -> New Terminal (Figure 1b). We can then install `uv`, which is a Python package manager, by putting `curl -LsSf https://astral.sh/uv/install.sh | sh` into the Terminal and pressing "return/enter" afterwards (Figure 1c). Finally, to install Python we can use `uv` by putting `uv python install` into that Terminal and pressing "return/enter" afterwards (Figure 1d).

Project set-up

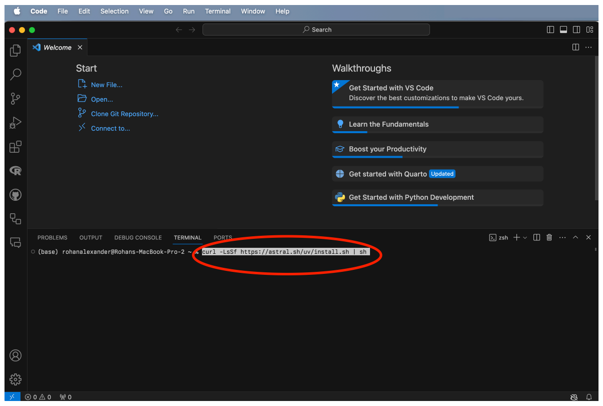
We are going to get started with an example that downloads some data from Open Data Toronto. To start, we need to create a project, which will allow all our code to be self-contained.



(a) Opening VS Code

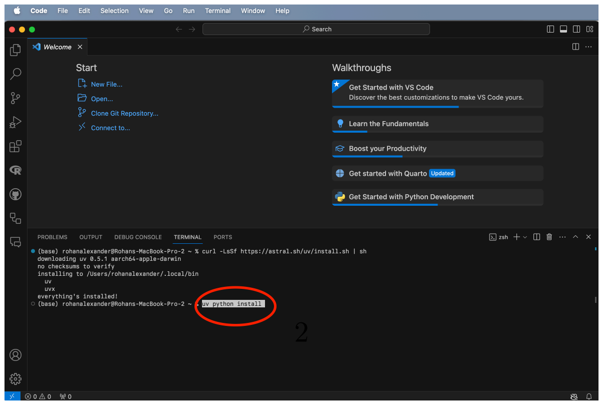


(b) Opening a Terminal in VS Code



```
(base) rohanalexander@Rohans-MacBook-Pro-2 ~ % curl -LsSf https://astral.sh/uv/install.sh | sh
```

(c) Install uv



```
(base) rohanalexander@Rohans-MacBook-Pro-2 ~ % curl -LsSf https://astral.sh/uv/install.sh | sh
downloading uv 0.5.1 aarch64-apple-darwin
no checksums to verify
installing to /Users/rohanalexander/.local/bin
  uv
  uvx
everything's installed!
(base) rohanalexander@Rohans-MacBook-Pro-2 ~ % uv python install
```

(d) Install Python

Figure 1: Opening VS Code and a new terminal and then installing uv and Python

Open VS Code and open a new Terminal: "Terminal" -> "New Terminal". Then use Unix shell commands to navigate to where you want to create your folder. For instance, use `ls` to list all the folders in the current directory, then move to one using `cd` and then the name of the folder. If you need to go back one level then use `..`.

Once you are happy with where you are going to create this new folder, we can use `uv init` in the Terminal to do this, pressing "return/enter" afterwards (`cd` then moves to the new folder "shelter_usage").

```
#| eval: false
#| echo: true

uv init week_1
cd week_1
```

By default, there will be a script in the example folder. We want to use `uv run` to run that script, which will then create an project environment for us.

```
#| eval: false
#| echo: true

uv run hello.py
```

A project environment is specific to that project. We will use the package `numpy` to simulate data. We need to add this package to our environment with `uv add`.

```
#| eval: false
#| echo: true

uv add numpy
```

We can then modify `hello.py` to use `numpy` to simulate from the Normal distribution.

```python
import numpy as np

def main():
    np.random.seed(853)

    mu, sigma = 0, 1
    sample_sizes = [10, 100, 1000, 10000]
    differences = []
    for size in sample_sizes:
        sample = np.random.normal(mu, sigma, size)
        sample_mean = np.mean(sample)
        diff = abs(mu - sample_mean)
        differences.append(diff)
        print(f"Sample size: {size}")
        print(f"  Difference between sample and population mean: {round(diff, 3)}")

if __name__ == "__main__":
    main()
```

After we have modified and saved `hello.py` we can run it with `uv run` in exactly the same way as before.

At this point we should close VS Code. We want to re-open it to make sure that our project environment is working as it needs to. In VS Code, a project is a self-contained folder. You can open a folder with "File" -> "Open Folder…" and then select the relevant folder, in this case "week_1". You should then be able to re-run `uv run hello.py` and it should work.

Simulate

We would like to more thoroughly simulate the dataset that we are interested in. We will use `polars` to provide a dataframe to store our simulated results, so we should add this to our environment with `uv add`.

```
#| eval: false
#| echo: true

uv add polars
```

Create a new Python file called `00-simulate_data.py`.

```
#### Preamble ####
# Purpose: Simulates a dataset of daily shelter usage
# Author: Rohan Alexander
# Date: 8 September 2025
# Contact: rohan.alexander@utoronto.ca
# License: MIT
# Pre-requisites:
# - Add `polars`: uv add polars
# - Add `numpy`: uv add numpy
```

```python
# - Add `datetime`: uv add datetime


#### Workspace setup ####
import polars as pl
import numpy as np
from datetime import date

rng = np.random.default_rng(seed=853)


#### Simulate data ####
# Simulate 10 shelters and some set capacity
shelters_df = pl.DataFrame(
    {
        "Shelters": [f"Shelter {i}" for i in range(1, 11)],
        "Capacity": rng.integers(low=10, high=100, size=10),
    }
)

# Create data frame of dates
dates = pl.date_range(
    start=date(2024, 1, 1), end=date(2024, 12, 31), interval="1d", eager=True
).alias("Dates")

# Convert dates into a data frame
dates_df = pl.DataFrame(dates)

# Combine dates and shelters
data = dates_df.join(shelters_df, how="cross")

# Add usage as a Poisson draw
poisson_draw = rng.poisson(lam=data["Capacity"])
usage = np.minimum(poisson_draw, data["Capacity"])

data = data.with_columns([pl.Series("Usage", usage)])

data.write_parquet("simulated_data.parquet")
```

We can then import our simulated dataset.

```
import polars as pl

df = pl.read_parquet("simulated_data.parquet")

print(df.head(5))
```

Acquire

Use this source: https://open.toronto.ca/dataset/daily-shelter-overnight-service-occupancy-capacity/

```
import polars as pl

# URL of the CSV file
url = "https://ckan0.cf.opendata.inter.prod-toronto.ca/dataset/21c83b32-d5a8-4106-a54f-010db

# Read the CSV file into a Polars DataFrame
df = pl.read_csv(url)

# Save the raw data
df.write_parquet("shelter_usage.parquet")
```

We are likely only interested in a few columns and only rows where there are data.

```
import polars as pl

df = pl.read_parquet("shelter_usage.parquet")

# Select specific columns
selected_columns = ["OCCUPANCY_DATE", "SHELTER_ID", "OCCUPIED_BEDS", "CAPACITY_ACTUAL_BED"]

selected_df = df.select(selected_columns)

# Filter to only rows that have data
filtered_df = selected_df.filter(df["OCCUPIED_BEDS"].is_not_null())

print(filtered_df.head())

renamed_df = filtered_df.rename({"OCCUPANCY_DATE": "date",
                                  "SHELTER_ID": "Shelters",
                                 "CAPACITY_ACTUAL_BED": "Capacity",
                                 "OCCUPIED_BEDS": "Usage"
                                 })

print(renamed_df.head())

renamed_df.write_parquet("cleaned_shelter_usage.parquet")
```

Explore

Manipulate the data into a summary table.

```
import polars as pl

df = pl.read_parquet("cleaned_shelter_usage.parquet")

# Convert the date column to datetime and rename it for clarity
df = df.with_columns(pl.col("date").str.strptime(pl.Date, "%Y-%m-%d").alias("date"))

# Group by "Dates" and calculate total "Capacity" and "Usage"
aggregated_df = (
    df.group_by("date")
    .agg([
        pl.col("Capacity").sum().alias("Total_Capacity"),
        pl.col("Usage").sum().alias("Total_Usage")
    ])
    .sort("date")  # Sort the results by date
)

# Display the aggregated DataFrame
print(aggregated_df)
```