DATA SCIENCE FOR ECONOMISTS

ECON 220 LAB

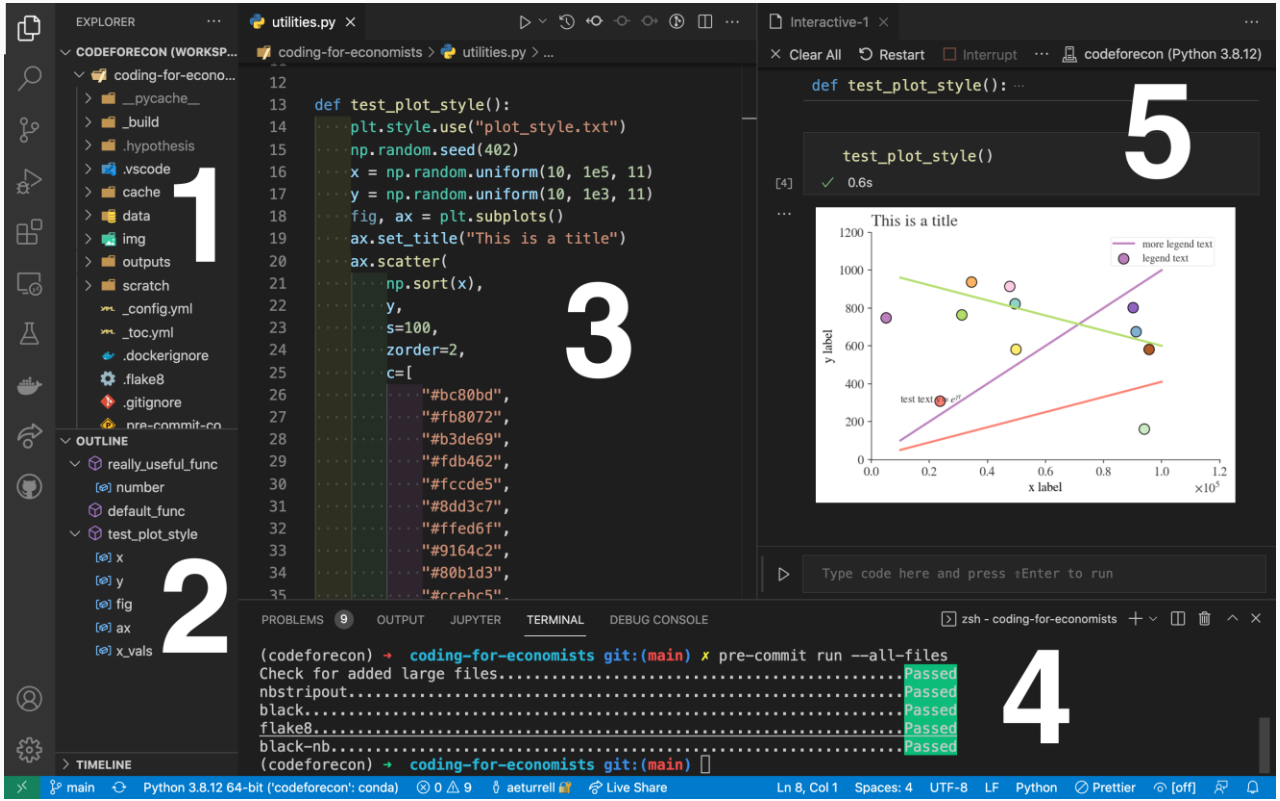
Jafet Baca-Obando

Week 2, Introduction to Python (I) – 09/05/2025

Outline

- Setup
- Loading data using Pandas, NumPy, and Matplotlib
- Exploration and descriptive statistics
- Basic visualization and data handling

Typical layout of Visual Studio Code



The economist's Python toolkit

- Pandas
  - Data manipulation and analysis (loading, cleaning, exporting, etc.)
- NumPy
  - Numerical computation (average, sum, median, etc.)
- Matplotlib
  - Visualization

Installation

- Python packages don’t come built-in
  - We need to install them (just once)
  - And then import them into our scripts

```python
pip install pandas
pip install numpy
pip install matplotlib
```

- Notes
  - Download **Jupyter Notebook** in the VS code extensions
  - VS Code might require you to install Ipykernel. Allow it to do so

Loading the packages into the workspace

- Import the necessary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

✓ 2.3s Python

Loading a dataset

- Use Pandas to load a CSV file directly from a URL.
- The data is loaded into a **DataFrame**, the central data structure in Pandas.
- Like an Excel spreadsheet

```python
# Specify the path to the data file
path = "https://raw.githubusercontent.com/jbacaob/ECON-220-Lab/refs/heads/main/Lectures/Week%202/starwars.csv"

# Load the dataset
data = pd.read_csv(path)
```

✓ 0.3s Python

First look: .head() and .tail()

- How do you get a quick overview of the data without printing the entire DataFrame?
  - **data.head(n):** Shows the **n** first rows
  - **data.tail(n):** Shows the last **n** rows

```
data.head(10)  # Display the first 10 rows of the dataset
```

✓ 0.0s Open 'data' in Data Wrangler Python

Exploring the DataFrame’s structure

- Before analysis, understand the dataset's dimensions and data types
  - **.shape**: Returns a tuple representing the dimensions (rows, columns).
  - **.columns**: Returns the list of column names.
  - **.dtypes**: Returns the data type of each column (object usually means string, float64 is a float, int64 is an integer).

```
data.shape  # Get the dimensions of the dataset (rows, columns)
```
✓ 0.0s Python

```
(87, 9)
```

```
data.columns  # Get the column names of the dataset
```
✓ 0.0s Python

```
Index(['Unnamed: 0', 'name', 'height', 'mass', 'hair_color', 'eye_color',
       'gender', 'homeworld', 'species'],
      dtype='object')
```

Scales of measurement

- **Categorical**
  - Places data into distinct groups or labels.
  - **Nominal**: Categories with no natural order (for example, homeworld)
  - **Ordinal:** Categories with a meaningful order (for example, income groups: low, medium, high)
- **Interval**: Variable with fixed increments (temperature, grades)
- **Ratio/continuous**: Changing increments (height, weight)

Descriptive statistics: numerical data

- For continuous (or ratio) variables like height, we can easily calculate summary statistics.
  - **.mean()**: Calculates the average.
  - **.describe()**: Provides a full summary (count, mean, standard deviation, min, max).

```
data['height'].mean()  # Calculate the mean height of characters
```

✓ 0.0s Python

```
174.35802469135803
```

```
data['height'].describe()  # Get summary statistics for the height column
```

✓ 0.0s Python

	# height
count	81.0
mean	174.35802469135803
std	34.770428758492216

Analyzing categorical data

- We’re often interested in the frequency of each category.
- **.value_counts()**: Returns a series containing counts of unique values.

```
eyecolor_count = data['eye_color'].value_counts()  # Get the frequency of each species in the
dataset
eyecolor_count  # Display the frequency counts
```

✓ 0.0s Open 'eyecolor_count' in Data Wrangler Python

eye_color	# count
brown	21
blue	19
yellow	11
black	10

Basics of data visualization

- A picture is worth a thousand words.
- Tables of numbers are useful, but visualizations are essential for:
  - Quickly identifying patterns, trends, and outliers (extreme values)
  - Communicating findings effectively
- We will use **matplotlib** to create our first plots.

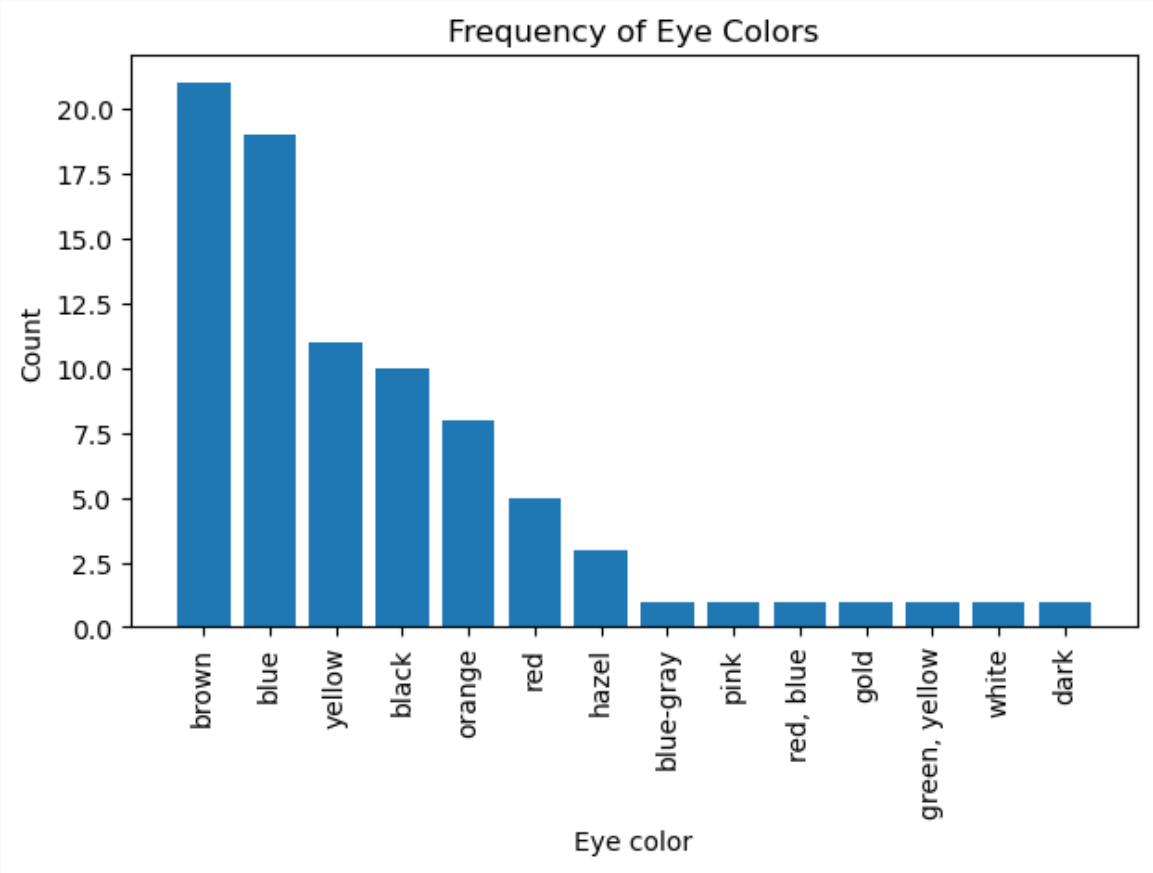
Visualizing frequencies: bar chart

- A bar chart is excellent for visualizing the frequency counts of categorical data.

```
plt.figure(figsize=(7, 4))
plt.bar(eyecolor_count.index, eyecolor_count.values)
plt.title('Frequency of Eye Colors')
plt.xlabel('Eye color')
plt.ylabel('Count')
plt.xticks(rotation=90)
plt.show()
```

✓ 0.1s Python

Visualizing frequencies: bar chart



Handling data: filtering

- We may want to analyze a subset of your data.
- Filter a DataFrame by passing a condition inside square brackets [ ].
  - Let's filter for eye colors that appear for more than 3 characters.

```
freq_eye_colors = eyecolor_count[eyecolor_count > 3]  # Filter eye colors with more than 5
characters
freq_eye_colors  # Display the filtered frequency counts
```

✓ 0.0s Open 'freq_eye_colors' in Data Wrangler Python

eye_color	# count
brown	21
blue	19
yellow	11
black	10
orange	8
red	5

Subsetting rows and columns

- Indexing + column selection to create more complex subsets.
- **Goal:** Find the homeworld of all 'Human' **or** 'Droid' characters.
  - | is the "OR" operator, & is the "AND" operator.

```
condition = (data['species'] == 'Human') | (data['species'] == 'Droid')
columns_to_keep = ['species', 'homeworld']
filtered_data = data[columns_to_keep][condition]  # Filter the dataset for human characters
filtered_data
```

✓ 0.0s Open 'filtered_data' in Data Wrangler Python

	species	homeworld
0	Human	Tatooine
1	Droid	Tatooine
2	Droid	Naboo
3	Human	Tatooine
4	Human	Alderaan

Sorting data

- Sorting your data makes it easier to inspect.
- **.sort_values(by='column_name')**: Sorts the DataFrame by the specified column.

```
filtered_data.sort_values(by='species')  # Sort the filtered data by species
```

✓ 0.0s Python

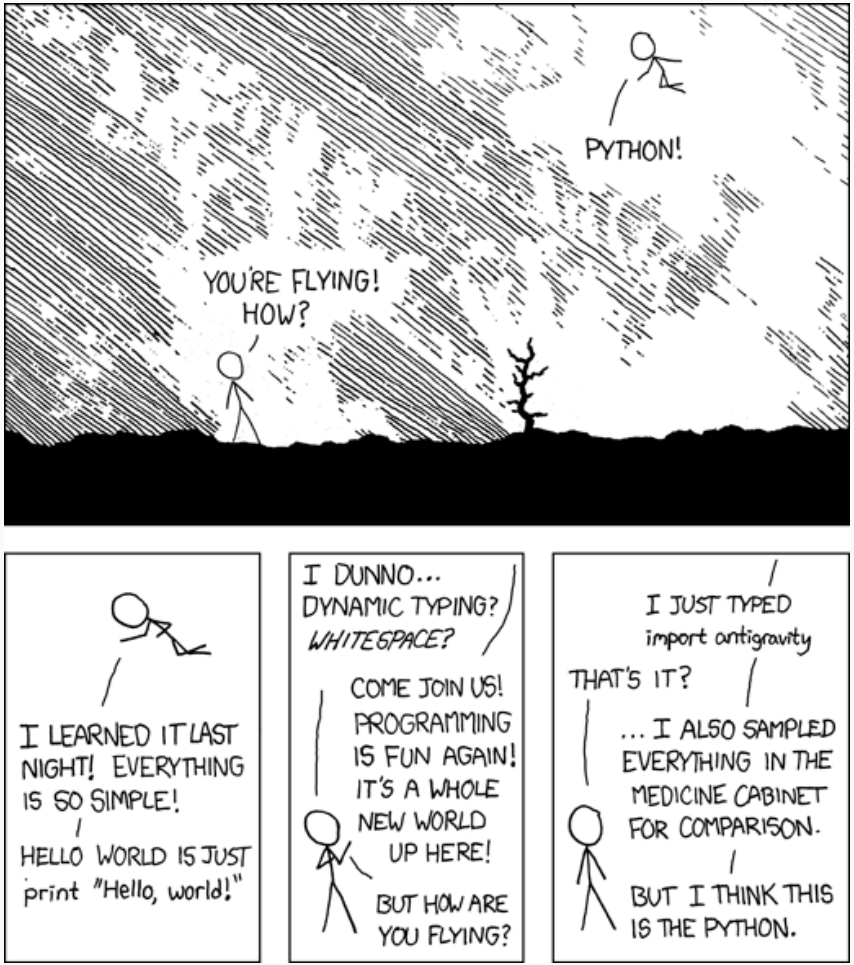
	species	homeworld
1	Droid	Tatooine
2	Droid	Naboo
84	Droid	*Missing value*
21	Droid	*Missing value*
7	Droid	Tatooine
0	Human	Tatooine
40	Human	Tatooine

Recap

- We loaded essential libraries (pandas, matplotlib).
- Imported data into a DataFrame.
- Explored its structure and computed summary statistics.
- Created basic visualizations.
- Subset and sorted the data based on conditions.

To-do list

- DataCamp
  - **Complete “Introduction to Python” course**
  - **Upload certificate on Canvas: 09/06/2025**
  - Complete “Intermediate Python” course
  - Upload certificate on Canvas: 09/13/2025



Python, from xkcd.