Python Camp — Day 2
Web Scraping & APIs

David Pouliot
Southern Oregon University

Today

- Functions — and reusing lists from Day 1
- Dictionaries — looking up data by name: `obj[key]`
- Installing packages with `pip install`
- How the web works: requests and responses
- HTML (web pages) vs JSON (data)

Install first:

```
pip install requests beautifulsoup4
```

How the Web Works

Request → Response

- Your program asks a server for a URL — a **request**.
- The server sends back data — a **response**.
- Two common shapes for that data:
  - HTML — a web page meant for humans. We dig out the parts we want.
  - JSON — structured data meant for programs. Much easier.

The requests library

One function does the asking: `requests.get()`.

```
1 import requests
2 response = requests.get("https://pokeapi.co/api/v2/pokemon/pikachu")
3 print(response.status_code)    # 200 means OK
```

Dictionaries

A dictionary: look up by name, not number

A list finds things by position. A dictionary finds them by name.

```
pokemon = {"name": "Pikachu", "hp": 35, "type": "electric"}

print(pokemon["name"])    # Pikachu
print(pokemon["hp"])      # 35
```

- Written with {curly braces} as "key": value pairs.
- The key (in quotes) is how you look up the value.

Why this matters: `obj[key]` is everywhere

The data you fetch today is dictionaries — so it's all the same syntax.

```
data = response.json()                  # JSON  -> a dictionary
data["stats"][0]["base_stat"]           # nested dicts + lists

book["title"]                           # BeautifulSoup tag
row["Name"]                             # a pandas column (Day 3)
```

Learn `obj[key]` once and you can read JSON, scraped tags, and spreadsheet data all week.

Advanced: key-safe lookups with `.get()`

Asking for a key that isn't there crashes your program:

```
print(pokemon["nickname"])       # KeyError -> crash!

print(pokemon.get("nickname"))               # None, no crash
print(pokemon.get("nickname", "no name"))    # your own default
```

Use `.get()` when a key might be missing — common with real API data that isn't always complete.

Track A: Scraping HTML

books.toscrape.com

BeautifulSoup pulls pieces out of an HTML page.

```
import requests
from bs4 import BeautifulSoup

url = "https://books.toscrape.com"
response = requests.get(url)
soup = BeautifulSoup(response.text, "html.parser")

books = soup.find_all("article", class_="product_pod")
for book in books:
    title = book.h3.a["title"]
    price = book.find("p", class_="price_color").text
    print(f"{title}: {price}")
```

Track B: JSON APIs

PokeAPI — no key needed

`.json()` turns the response into a Python dictionary.

```
1 import requests
2
3 name = input("Enter a Pokemon name: ")
4 response = requests.get(f"https://pokeapi.co/api/v2/pokemon/{name.lower()}"
5 data = response.json()
6
7 print(f"HP:      {data['stats'][0]['base_stat']}")
8 print(f"Attack: {data['stats'][1]['base_stat']}")
```

Track C — NASA Picture of the Day

Free instant key at api.nasa.gov.

```python
import requests

API_KEY = "your_key_here"
url = f"https://api.nasa.gov/planetary/apod?api_key={API_KEY}"
data = requests.get(url).json()

print(data["title"])
print(data["url"])
```

Stretch goals

- Save your results to a `.txt` file
- Fetch several items (pages, Pokémon, or dates)
- Find the most expensive book / strongest Pokémon
- Let the user type what to look up
- Store results in a list and `sort()` them

Pick **one** track to start. You can always try another after.

Saving your results to a file

Use `with open(...)` — it closes the file for you, even if something goes wrong.

```
with open("results.txt", "w") as f:
    f.write(f"{title}: {price}\n")
    f.write("more lines here...\n")
```

- `"w"` means write (it replaces the file's contents).
- Everything indented under `with` can use the file `f`.
- `\n` starts a new line.

Pick-Your-Own Projects

More project options — pick any

Every one is a free, no-key API or the same scraping pattern.

Starter — one request

- Weather lookup (Open-Meteo)
- Dad-joke fetcher
- Bored? activity suggester

Standard — a loop or a list

- Quotes scraper
- Trivia quiz with score
- Crypto price checker

Boss challenge: combine two APIs into one program · scrape every page to a CSV

Starter code and links for all of these are on the camp website.