First PyPI Example.

Why is this important?

When working on Python projects, it's common to use external libraries to add functionality without having to write everything from scratch. The Python Package Index (PyPI) is a repository of software for the Python programming language, where you can find and install packages created by other developers.

What are we going to do?

We're going to create a virtual environment and install a few packages from PyPI to see how they work.

Steps

1. Create a virtual environment in your project folder.

```
python -m venv venv
```

This command creates a new directory called `venv` that contains the virtual environment.

You can think of this as a separate space on your computer where you can install packages without affecting your global Python installation.

2. Activate the virtual environment.

- On Windows:

```
venv\Scripts\activate
```

- On macOS and Linux:

```
source venv/bin/activate
```

When the virtual environment is activated, your command prompt will change to show the name of the environment (usually `venv`).

This is something you need to do every time you want to work on your project, normally we call it `venv` but you can name it whatever you want.

Note: If you folks have been working with `npm` or `node.js` this is similar to `node_modules` but it's a bit more isolated.

3. Let's install the `freegames` package from PyPI. This package allows us to fetch information about free games available on various platforms.

```
pip install freegames
```

Now to learn how to use this package we can check out its documentation on PyPI

- On the page you find the description on how to use the package.
- On the left side of the page you can find links to the "Source" which is the code repository which you can find here and "Homepage" which is the documentation page which you can find here.
  - most packages will have these links and we'll be using this.

Generally documentation pages will have examples on how to use the package.

- In Python there's normally going to be documentation that either use `sphinx` or `mkdocs` to generate the documentation pages.

4. Let's run the `snake` game from the documentation

```
python -m freegames.snake
```

Neat right?

5. How does this work? Let's go take a look at the code for the `freegames` package.

- Go to the following link to see the source code on GitHub: freegames source code
- you can observe here how it's using a built in library called `turtle` to create the game graphics.
- you can also observe that

6. Now that have this package how do we save it so that we can use it later? We can use a command called `pip freeze` to save the packages we have installed in our virtual environment to a file called `requirements.txt`

- Note pip freeze will save all the packages you have installed in your virtual environment.

```
pip freeze > requirements.txt
```

This command creates a file called `requirements.txt` that lists all the packages and their versions installed in the virtual environment.

- you can see here that `freegames` is listed along with its version here what it looks like:

```
freegames==2.5.3
```

7. Let's install another package called `requests` which is a popular package for making HTTP requests in Python.

```
pip install requests
```

Let's also save this to our `requirements.txt` file again with the command

```
pip freeze > requirements.txt
```

Now you can see that `requests` is also listed in the `requirements.txt` file along with its version:

```
certifi==2025.10.5
charset-normalizer==3.4.4
freegames==2.5.3
idna==3.11
requests==2.32.5
urllib3==2.5.0
```

You can see that `requests` has some dependencies like `certifi`, `charset-normalizer`, `idna`, and `urllib3` which were also installed automatically.

8. Let's get the top headlines from the HackerNewsAPI using the `requests` package.

- Let's read the `requests` documentation: requests documentation
- The hacker news API documentation can be found here: Hacker News API this just gets the top stories from hackernews
- Note that `requests` is one of the most popular packages in Python and has great documentation, this allows us to get REST API data easily (a bit like `fetch` in JavaScript).
- Create a new Python file called `hackernews.py` and add the following code:

```python
import requests

def get_top_ten_headlines():
    headlined_ids = requests.get("https://hacker-news.firebaseio.com/v0/topstories.json").json()[:10]
    return headlined_ids

def get_headline_by_id(headline_id):
    headline = requests.get(f"https://hacker-news.firebaseio.com/v0/item/{headline_id}.json").json()
    return headline

def main():
    top_ten_ids = get_top_ten_headlines()
    for index, id in enumerate(top_ten_ids):
        headline = get_headline_by_id(id)
        print(f"Headline {index + 1}.")
        print(f"Title: {headline['title']}")
        print(f"URL: {headline.get('url', 'No URL available')}")
        print("-" * 40)

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

- Run the code with the command:

```
python hackernews.py
```

You should see the top ten headlines from Hacker News printed in your terminal.

9. Once you're done working on this project you can deactivate the virtual environment with the command:

```
deactivate
```

This command will return your command prompt to its normal state (without the `(venv)` prefix) to your terminal.

Summary

In this example, we created a virtual environment, installed packages from PyPI, and used those packages in a Python script. We also learned how to manage our project's dependencies using a `requirements.txt` file. This is a fundamental skill for Python developers, as it allows for better project organization and dependency management.