

### Modules and Packages





## Using PyPi With pip install





- PyPI is a repository for open-source thirdparty Python packages.
- It's similar to RubyGems in the Ruby world, PHP's Packagist, CPAN for Perl, and NPM for Node.js.



- So far we've really only used libraries that come internally with Python.
- There are many other libraries available that people have open-sourced and shared on PyPi.
- We can use **pip install** at the command line to install these packages.





- By installing Python from python.org or through the Anaconda distribution you also installed pip
- pip is a simple way to download packages at your command line directly from the PyPi repository



- There are packages already created for almost any use case you can think of!
- A quick google search will usually help you discover a link to the PyPi page for the package, or for the package documentation.



- Let's quickly show you how to download and install external packages.
  - Windows Users: Command Prompt
  - MacOS/Linux Users: Terminal



# Writing Your Own Modules and Packages





- Now that we understand how to install external packages, let's explore how to create our own modules and packages.
- Modules are just .py scripts that you call in another .py script.
- Packages are a collection of modules.
- Let's create some examples!





\_\_name\_\_ and "\_main\_"





- An often confusing part of Python is a mysterious line of code:
  - o if \_\_name\_\_ == "\_\_main\_\_":



• Sometimes when you are importing from a module, you would like to know whether a modules function is being used as an import, or if you are using the original .py file of that module.





 Let's explore this some more, but make sure to check out the full explanatory text file that is in this part's folder!