PHYS 4332 – Computational Physics – W2025

Python Tutorial

2.0 – Installing packages using pip

The base installation of Python includes the Python standard library. This is an extensive library of build-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers (e.g., the `os()` module), as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming (e.g., the `datetime()` module).

In addition to the standard library, the Python Package Index (PyPI) is an active collection of more than 100,000 third-party modules, packages, and entire application development frameworks. This is a repository of Python software developed, maintained, and shared by the Python community. The Package Installer for Python (`pip`, part of the standard library) provides a simple way of installing third-party packages from PyPI and other repositories.

- To get the version of pip you currently have installed, from a terminal/command prompt type:

```
>>> pip --version

pip 24.3.1 from C:\Python 3.12\Lib\site-packages\pip (python 3.12)
```

- To upgrade pip to the latest version, type:

```
>>> python -m pip install --upgrade pip

Requirement already satisfied: pip in c:\python 3.12\lib\site-packages (24.3.1)
```

- To get a list of all available commands and options for pip, use `help`:

```
>>> pip help

Usage:
  pip <command> [options]

Commands:
  install                     Install packages.
  download                    Download packages.
  uninstall                   Uninstall packages.
  freeze                      Output installed packages in requirements format.
  list                        List installed packages.
  show                        Show information about installed packages.
  check                       Verify installed packages have compatible dependencies.
  config                      Manage local and global configuration.
  search                      Search PyPI for packages.
  cache                       Inspect and manage pip's wheel cache.
```

```
  index                       Inspect information available from package indexes.
  wheel                       Build wheels from your requirements.
  hash                        Compute hashes of package archives.
  completion                  A helper command used for command completion.
  debug                       Show information useful for debugging.
  help                        Show help for commands.

General Options:
  -h, --help                  Show help.
...
```

- You can also get detailed help for specific commands (e.g. `install`):

```
>>> pip help install

Usage:
  pip install [options] <requirement specifier> [package-index-options] ...
  pip install [options] -r <requirements file> [package-index-options] ...
  pip install [options] [-e] <vcs project url> ...
  pip install [options] [-e] <local project path> ...
  pip install [options] <archive url/path> ...

Description:
  Install packages from:

  - PyPI (and other indexes) using requirement specifiers.
  - VCS project urls.
  - Local project directories.
  - Local or remote source archives.

  pip also supports installing from "requirements files", which provide
  an easy way to specify a whole environment to be installed.
...
```

- To list the packages currently installed, use `list`:

```
>>> pip list
```

- To install a new package, use `install`. By default, this will install the latest version of the package. For example, to install the `numpy` package:

```
>>> pip install numpy

Collecting numpy
  Downloading numpy-2.1.3-cp312-cp312-win_amd64.whl.metadata (59 kB)
Downloading numpy-2.1.3-cp312-cp312-win_amd64.whl (12.6 MB)
---------------------------------------- 12.6/12.6 MB 14.9 MB/s eta 0:00:00
Installing collected packages: numpy
Successfully installed numpy-2.1.3
```

  Here, `pip` found and installed `numpy` version 2.1.3.

- To install a specific version of a package, use `install==`version number. This will automatically uninstall other versions of the same package on your system. For example, to install version 2.1.0 of the `numpy` package:

```
>>> pip install numpy==2.1.0

Collecting numpy==2.1.0
  Downloading numpy-2.1.0-cp312-cp312-win_amd64.whl.metadata (59 kB)
Downloading numpy-2.1.0-cp312-cp312-win_amd64.whl (12.6 MB)
---------------------------------------- 12.6/12.6 MB 19.2 MB/s eta 0:00:00
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 2.1.3
    Uninstalling numpy-2.1.3:
      Successfully uninstalled numpy-2.1.3
Successfully installed numpy-2.1.0
```

- Now let's see if any packages are outdated by using the `--outdated` option of the `list` command:

```
>>> pip list --outdated

Package Version Latest Type
------- ------- ------ -----
numpy   2.1.0   2.1.3  wheel
```

- To upgrade a package to the latest version, use the `-U` option:

```
>>> pip install -U numpy
```

- To uninstall a package, use `uninstall` and type `y` to confirm:

```
>>> pip uninstall numpy
```

- To get a list of installed packages in a requirements format, use `freeze`:

```
>>> pip freeze
```

- To export this list to a file called `Requirements.txt`:

```
>>> pip freeze > Requirements.txt
```

- To automatically install all the packages listed in `Requirements.txt` (with their corresponding versions):

```
>>> pip install -r Requirements.txt
```