CERTIFIED ASSOCIATE IN PYTHON PROGRAMMING

By: Imran

Module 1: Modules in Python
Part 3: Python Package Installer (PIP)

PYTHON PACKAGING ECOSYSTEM AND HOW TO USE IT

PYTHON PACKAGING ECOSYSTEM AND HOW TO USE IT

- The repository (or *repo* for short) we mentioned before is named **PyPI** (it's short for Python Package Index) and it's maintained by a workgroup named the Packaging Working Group, a part of the Python Software Foundation, whose main task is to support Python developers in efficient code dissemination.
- You can find their website here: https://wiki.python.org/psf/PackagingWG.
- The PyPI website (administered by PWG) is located at the address: https://pypi.org/.

As of June 2020, PyPI was a home to 237,515 projects, consisting of 2,877,545 files managed by 427,487 users.

PYTHON PACKAGING ECOSYSTEM AND HOW TO USE IT

- We must point out that PyPI is not the only existing Python repository. On the contrary, there are lots of them, created for projects and led by many larger and smaller Python communities. It's likely that someday you and your colleagues may want to create your own repos.
- Anyway, PyPI is the most important Python repo in the world. If we modify the classic saying a little, we can state that “all Python roads lead to PyPI”, and that’s no exaggeration at all.

PIP

- Installation of PIP doesn't only depend on the OS you use, although this is a very important factor.
- Let's start with MS Windows.

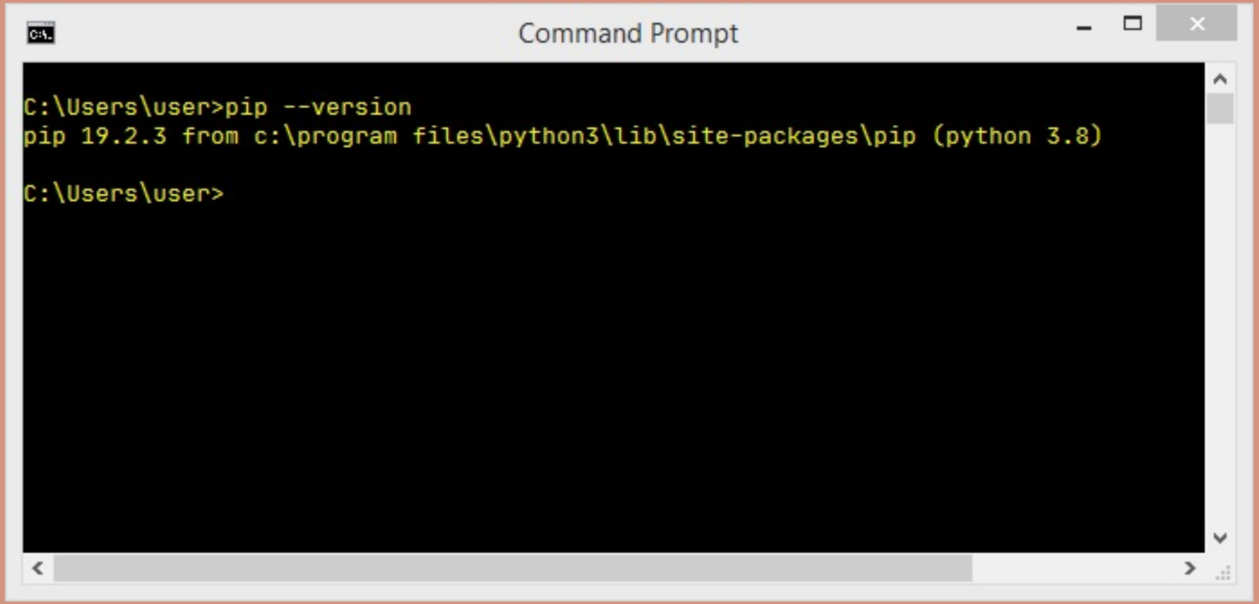
pip on MS Windows

- The MS Windows Python installer already contains *pip*, and so no other steps need to be taken in order to install it. Unfortunately, if the PATH variable is misconfigured, *pip* may unavailable.
- To verify that we haven't misled you, try to do this:
  - open the Windows console (*CMD* or *PowerShell*, whatever you prefer)
  - execute the following command:

```
pip --version
```

pip on MS Windows

- in the most optimistic scenario (and we really want that to happen) you'll see something like this



```
C:\Users\user>pip --version
pip 19.2.3 from c:\program files\python3\lib\site-packages\pip (python 3.8)

C:\Users\user>
```

pip on MS Windows

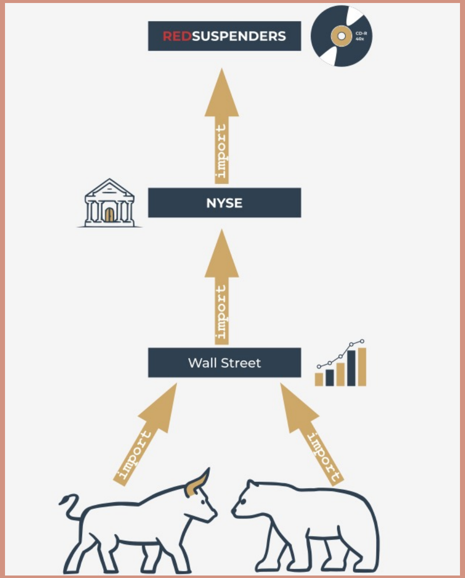
- the absence of this message may mean that the PATH variable either incorrectly points to the location of the Python binaries, or doesn't point to it at all; for example, our PATH variable contains the following substring:

  C:\Program Files\Python3\Scripts\;C:\Program Files\Python3\;

- the easiest way to reconfigure the PATH variable is to **reinstall Python**, instructing the installer to set it for you.

DEPENDENCIES

- Now that we're sure that *pip* is ready at our command, we're going to limit our focus to MS Windows only, as its behavior is (should be) the same in all OSes, but before we start, we need to explain an important issue and tell you about **dependencies**.
- Imagine that you've created a brilliant Python application named *redsuspenders*, able to predict stock exchange rates with 99% accuracy.
- Of course, you've used some existing code to achieve this goal – e.g., your app imports a package named *nyse* containing some crucial functions and classes. Moreover, the *nyse* package imports another package named *wallstreet*, while the *wallstreet* package imports other two essential packages named *bull* and *bear*.



DEPENDENCIES

- As you’ve probably already guessed, the connections between these packages are crucial, and if somebody decides to use your code (but remember, we've already called dibs on it) they will also have to ensure that all required packages are in place.
- To make a long story short, we can say that **dependency is a phenomenon that appears every time you're going to use a piece of software that relies on other software**. Note that dependency may include (and generally does include) more than one level of software development.
- Does this mean that a potential *nyse* package user is obliged to trace all dependencies and manually install all the needed packages? That would be horrible, wouldn't it?

DEPENDENCIES

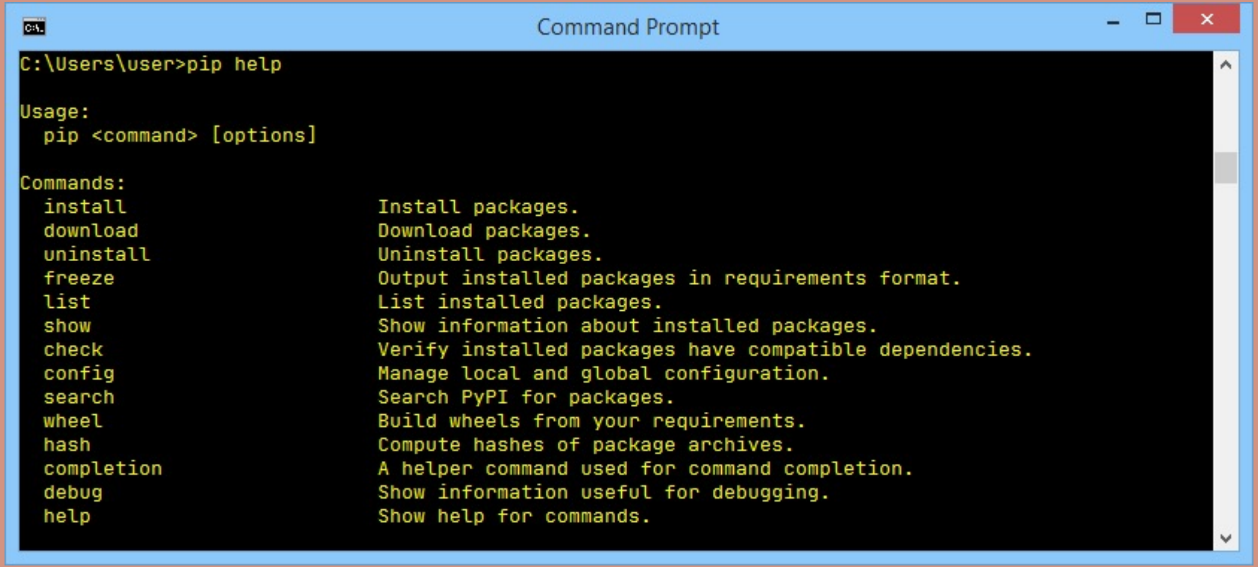
- Yes, it's definitely horrible, so you shouldn't be surprised that the process of arduously fulfilling all the subsequent requirements has its own name, and it's called *dependency hell*.
- How do we deal with that? Is every user doomed to visit hell in order to run the code for the first time?
- Fortunately not - *pip* can do all of this for you. Really. It can discover, identify, and resolve all dependencies. Moreover, it can do it in the cleverest way, avoiding any unnecessary downloads and reinstalls.

How to use pip

- Now we're ready to ask *pip* what it can do for us. Let's do it – issue the following command:

```
pip help
```

and wait for *pip*'s response. This is what it looks like:


Command Prompt

```
C:\Users\user>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.
  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.
```

How to use pip

- Don't forget that you may be obliged to replace *pip* with *pip3* if your environment requires this.
- The list produced by *pip* summarizes all the available operations, and the last of them is `help`, which we've just used already.
- If you want to know more about any of the listed operations, you can use the following form of *pip* invocation:

```
pip help operation
```

- For example, the line:

```
pip help install
```

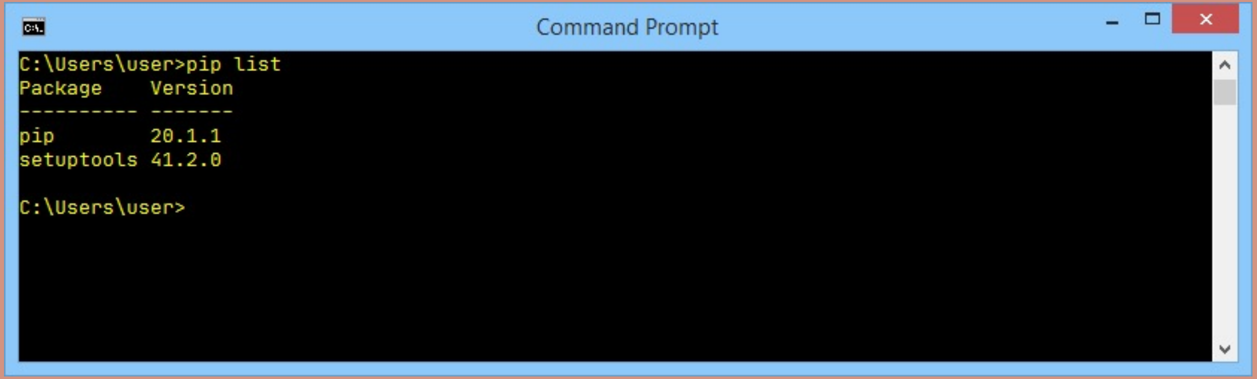
- will show you detailed information about using and parameterizing the `install` command.

How to use pip

- If you want to know what Python packages have been installed so far, you can use the `list` operation – just like this:

```
pip list
```

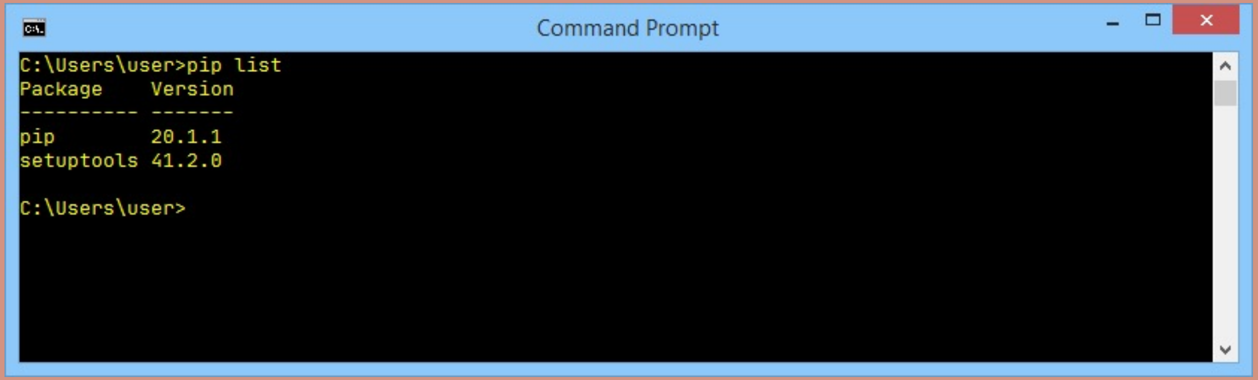
- The output you'll see is rather unpredictable. Don't be surprised if your screen ends up being filled with completely different content. Ours look as follows:



```
C:\Users\user>pip list
Package    Version
---------- -------
pip        20.1.1
setuptools 41.2.0

C:\Users\user>
```

How to use pip



```
C:\Users\user>pip list
Package    Version
---------- -------
pip        20.1.1
setuptools 41.2.0

C:\Users\user>
```

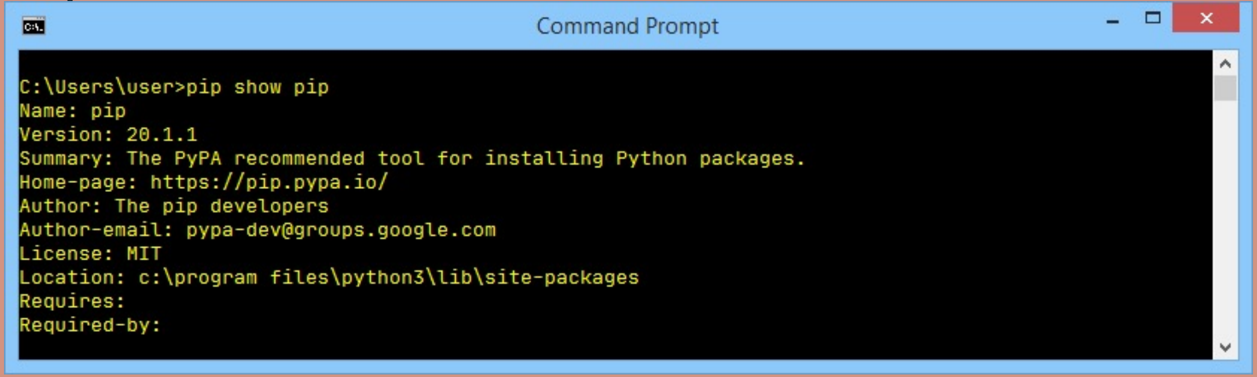
- As you can see, there are two columns in the list, one showing the name of the installed package, and the other showing the version of the package. We can't predict the state of your Python installation.
- The only thing we know for sure is that your list contains the two lines we see on our list: *pip* and *setuptools*. This happens because the OS is convinced that a user wanting pip will very likely need the *setuptools* soon. It's not wrong.

How to use pip

- The *pip* list isn't very informative, and it may happen that it won't satisfy your curiosity. Fortunately, there's a command that can tell you more about any of the installed packages (note the word **installed**). The syntax of the command looks as follows:
- `pip show package_name`
- We're going to use it in a slightly deceptive way – we want to convince *pip* to confess something about itself. This is how we do it:

```
pip show pip
```

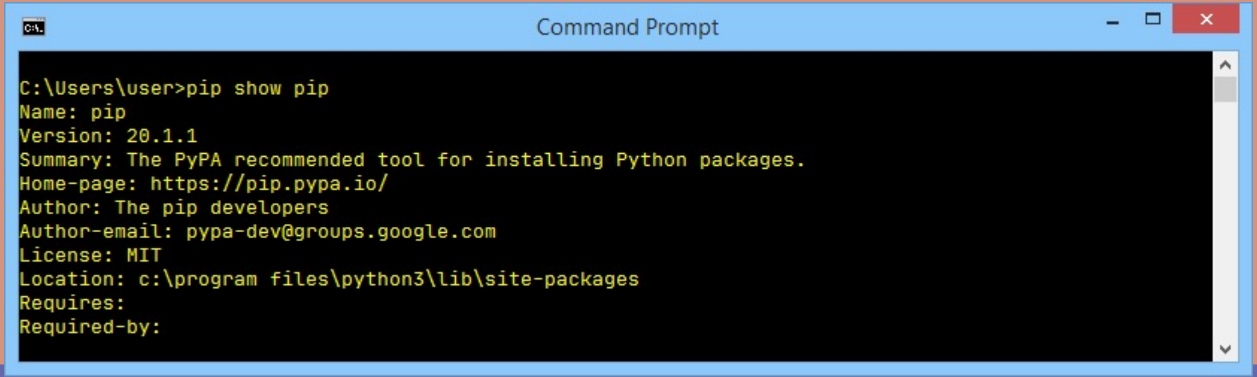
- It looks a bit odd, doesn't it? Despite this, it works fine, and *pip*'s self-presentation looks consistent and current:



```
C:\Users\user>pip show pip
Name: pip
Version: 20.1.1
Summary: The PyPA recommended tool for installing Python packages.
Home-page: https://pip.pypa.io/
Author: The pip developers
Author-email: pypa-dev@groups.google.com
License: MIT
Location: c:\program files\python3\lib\site-packages
Requires:
Required-by:
```

How to use pip

- You may ask where this data comes from? Is *pip* really so perceptive? Not at all – the information appearing on the screen is taken from inside the package being shown. In other words, the package's creator is obliged to equip it with all the needed data (or to express it more precisely – metadata).
- Look at the two lines at the bottom of the output. They show:
  - which packages are needed to successfully utilize the package (Requires:)
  - which packages need the package to be successfully utilized (Required-by:)
- As you can see, both properties are empty. Feel free to try to use the show command in relation to any other installed package.



```
C:\Users\user>pip show pip
Name: pip
Version: 20.1.1
Summary: The PyPA recommended tool for installing Python packages.
Home-page: https://pip.pypa.io/
Author: The pip developers
Author-email: pypa-dev@groups.google.com
License: MIT
Location: c:\program files\python3\lib\site-packages
Requires:
Required-by:
```

How to use pip

- The power of *pip* comes from the fact that it's actually a gateway to the Python software universe. Thanks to that, you can browse and install any of the hundreds of ready-to-use packages gathered in the PyPI repositories. Don't forget that *pip* is not able to store all PyPI content locally (it's unnecessary and it would be uneconomical).
- In effect, *pip* uses the Internet to query PyPI and to download the required data. This means that you have to have a network connection working whenever you're going to ask *pip* for anything that may involve direct interactions with the PyPI infrastructure.
- One of these cases occurs when you want to search through PyPI in order to find a desired package. This kind of search is initiated by the following command:

```
pip search anystring
```

How to use pip

- The anystring provided by you will be searched in:
  - the names of all the packages;
  - the summary strings of all the packages.
- Be aware of the fact that some searches may generate a real avalanche of data, so try to be as specific as possible. For example, an innocent-looking query like this one:

```
pip search pip
```

- produces more than 100 lines of results (try it yourself – don't take our word for it). By the way – the search is case insensitive.
- If you’re not a fan of console reading, you can use the alternative way of browsing PyPI content offered by a search engine, available at https://pypi.org/search.

How to use pip

- Assuming that your search is successful (or you’re determined to install a specific package of an already known name) you can use pip to install the package onto your computer.
- Two possible scenarios may be put into action now:
  - you want to install a new package for you only – it won't be available for any other user (account) existing on your computer; this procedure is the only one available if you can’t elevate your permissions and act as a system administrator;
  - you’ve decided to install a new package system-wide – you have administrative rights and you're not afraid to use them.
- To distinguish between these two actions, pip uses a dedicated option named --user (note the double dash). The presence of this option instructs pip to act locally on behalf of your (non-administrative) user.
- If you don’t add this, pip assumes that you’re as a system administrator and it’ll do nothing to correct you if you’re not.

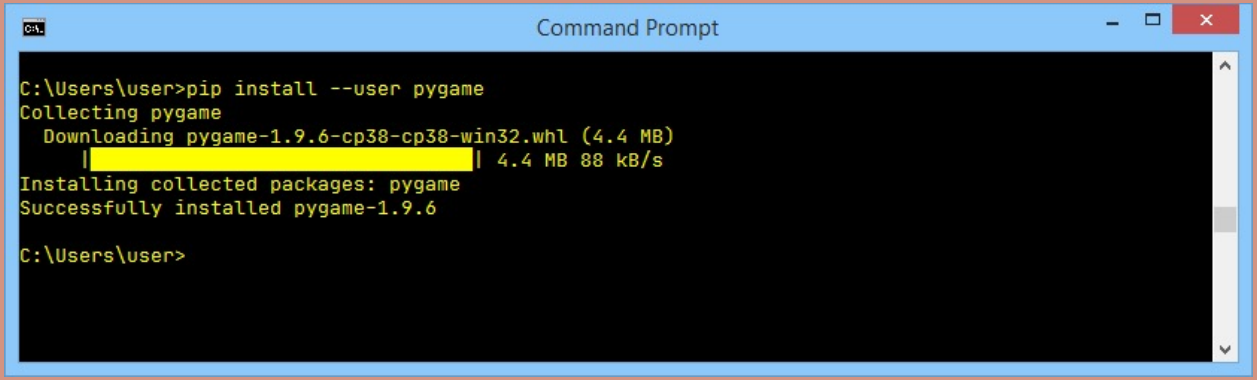
How to use pip

- In our case, we’re going to install a package named pygame – it's an extended and complex library allowing programmers to develop computer games using Python.
- If you’re a system administrator, you can install pygame using the following command:

```
pip install pygame
```

- If you're not an admin, or you don't want to fatten up your OS by installing pygame system-wide, you can install it for you only:

```
pip install --user pygame
```



```
C:\Users\user>pip install --user pygame
Collecting pygame
  Downloading pygame-1.9.6-cp38-cp38-win32.whl (4.4 MB)
     |████████████████████████████████| 4.4 MB 88 kB/s
Installing collected packages: pygame
Successfully installed pygame-1.9.6

C:\Users\user>
```

How to use pip

- The pip install has two important additional abilities:
  - it is able to **update** a locally installed package – e.g., if you want to make sure that you're using the latest version of a particular package, you can run the following command:

```
pip install -U package_name
```

  - where -U means update. Note: this form of the command makes use of the --user option for the same purpose as presented previously;
- it is able to **install a user-selected version** of a package (*pip* installs the **newest** available version by default); to achieve this goal you should use the following syntax:

```
pip install
```

How to use pip

- it is able to **install a user-selected version** of a package (*pip* installs the **newest** available version by default); to achieve this goal you should use the following syntax:

```
pip install package_name==package_version
```

- (note the double equals sign) e.g.,

```
pip install pygame==1.9.2
```

How to use pip

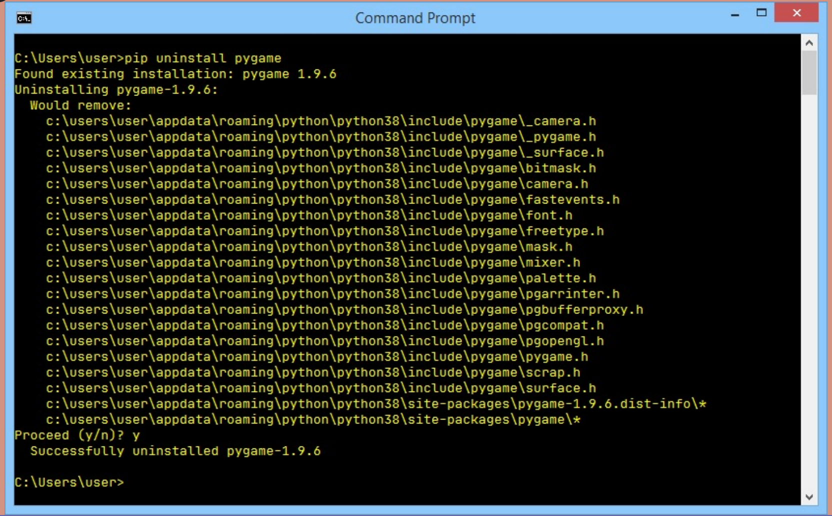
- If any of the currently installed packages are **no longer needed** and you want to get rid of them, *pip* will be useful, too. Its `uninstall` command will execute all the needed steps.
  - The required syntax is clear and simple:

```
pip uninstall package_name
```

  - so if you don't want *pygame* anymore you can execute the following command:

```
pip uninstall pygame
```

- *Pip* will want to know if you're sure about the choice you're making – be prepared to give the right answer.
- The process looks like this:



```
C:\Users\user>pip uninstall pygame
Found existing installation: pygame 1.9.6
Uninstalling pygame-1.9.6:
  Would remove:
    c:\users\user\appdata\roaming\python\python38\include\pygame\_camera.h
    c:\users\user\appdata\roaming\python\python38\include\pygame\_pygame.h
    c:\users\user\appdata\roaming\python\python38\include\pygame\_surface.h
    c:\users\user\appdata\roaming\python\python38\include\pygame\bitmask.h
    c:\users\user\appdata\roaming\python\python38\include\pygame\camera.h
    c:\users\user\appdata\roaming\python\python38\include\pygame\fastevents.h
    c:\users\user\appdata\roaming\python\python38\include\pygame\font.h
    c:\users\user\appdata\roaming\python\python38\include\pygame\freetype.h
    c:\users\user\appdata\roaming\python\python38\include\pygame\mask.h
    c:\users\user\appdata\roaming\python\python38\include\pygame\mixer.h
    c:\users\user\appdata\roaming\python\python38\include\pygame\palette.h
    c:\users\user\appdata\roaming\python\python38\include\pygame\pgarrinter.h
    c:\users\user\appdata\roaming\python\python38\include\pygame\pgbufferproxy.h
    c:\users\user\appdata\roaming\python\python38\include\pygame\pgcompat.h
    c:\users\user\appdata\roaming\python\python38\include\pygame\pgopengl.h
    c:\users\user\appdata\roaming\python\python38\include\pygame\pygame.h
    c:\users\user\appdata\roaming\python\python38\include\pygame\scrap.h
    c:\users\user\appdata\roaming\python\python38\include\pygame\surface.h
    c:\users\user\appdata\roaming\python\python38\site-packages\pygame-1.9.6.dist-info\*
    c:\users\user\appdata\roaming\python\python38\site-packages\pygame\*
Proceed (y/n)? y
  Successfully uninstalled pygame-1.9.6

C:\Users\user>
```

Excercise

Why should I ensure which one of *pip* and *pip3* works for me?

Excercise

Why should I ensure which one of *pip* and *pip3* works for me?

When Python 2 and Python 3 coexist in your OS, it's likely that *pip* identifies the instance of pip working with Python 2 packages only.

Excercise

How can I determine if my *pip* works with either Python 2 or Python 3?

Excercise

How can I determine if my *pip* works with either Python 2 or Python 3?

`pip --version` will tell you that.

Excercise

Unfortunately, I don't have administrative right. What should I do to install a package system-wide?

Excercise

Unfortunately, I don't have administrative right. What should I do to install a package system-wide?

You have to ask your *sysadmin* - don't try to hack your OS!