Getting Started With Python

- Setting the scene
- Running Python script code
- Creating virtual environment

> Demo folder: 01-GettingStarted

1. Setting the Scene

- Hello Python
- What can you do with Python?
- Downloading Python
- Using the Python documentation
- Installing Python packages

Hello Python

- Python is a powerful and expressive programming language
  - Object-oriented
  - Dynamic typing
  - Interpreted
- Available on a wide range of platforms
  - Unix/Linux
  - Windows
  - Mac OS X
  - etc …

What can you do with Python?

- Scripting
- File I/O
- String handling and regular expressions
- Web applications and REST web services
- Data science

Downloading Python

Are we all there?

- Anaconda
- PyCharm CE

Using Python Documentation

- Docs available online at https://docs.python.org

Installing Python Packages

There are many Python packages available

- e.g. NumPy, MatPlotLib, etc.
- See https://pypi.python.org/pypi for details

You can use the pip package manager to install Python packages:

- For example, to install the NumPy package:

```
pip install numpy
```

To find where pip installed a package:

```
pip show numpy
```

Note: These are already installed in Anaconda

2. Running Python Script

- Running Python script interactively
- Creating variables
- Line continuation
- Blocks
- Creating and running Python modules
- Python keywords

Running Python Script Interactively

Run the Python interpreter in interactive mode, and execute Python code.

```
python
```

Then enter some Python code. For example:

```python
print("Hello World")
```

Creating Variables

In Python, you don't need to declare a variable

- Just assign it a value, and Python will create it for you dynamically

Rules for identifiers in Python

- Can contain uppercase or lowercase letters, digits, and underscore
- But can't start with a digit

```python
firstname = "Homer"
lastname = "Simpson"
fullname = firstname + " " + lastname
print(fullname)
```

Line Continuation

If a statement spans multiple lines…

- You can use `\` to continue from one line to the next

```python
firstname = "Homer"
lastname = "Simpson"
fullname = firstname + \
" " + \
lastname
print(fullname)
```

Blocks

Python uses indentation to denote blocks

- Don't use {}
- Use `` `:` `` to indicate the start of an indented block

```python
age = 21
if age >= 18 and age <= 30:
    print("You are eligible for an 18-30s holiday!")
print("That's all folks")
```

Creating and Running Python Modules

You can put Python code into modules

- A module is just a script file containing Python code
- Typically starts with a lowercase letter and ends in `` `.py` ``

greeting.py

```python
print("Hello Python!")
print("This is my module")
```

You can run the module via the Python interpreter

```
python greeting.py
```

Python Keywords

Here is a full list of all the keywords in Python

- False, None, True
- if, elif, else, assert, is
- and, or, not
- for, in, from, while, break, continue, pass
- def, return, global, nonlocal, lambda
- import, from
- class, del
- raise, try, except, as, finally
- with, as
- yield
- await, async

Any questions?

Annex: Creating a Virtual Environment

- Overview
- Installing virtualenv
- Creating a virtual environment for a project
- Activating a virtual environment
- Using virtualenv
- Deactivating a virtual environment

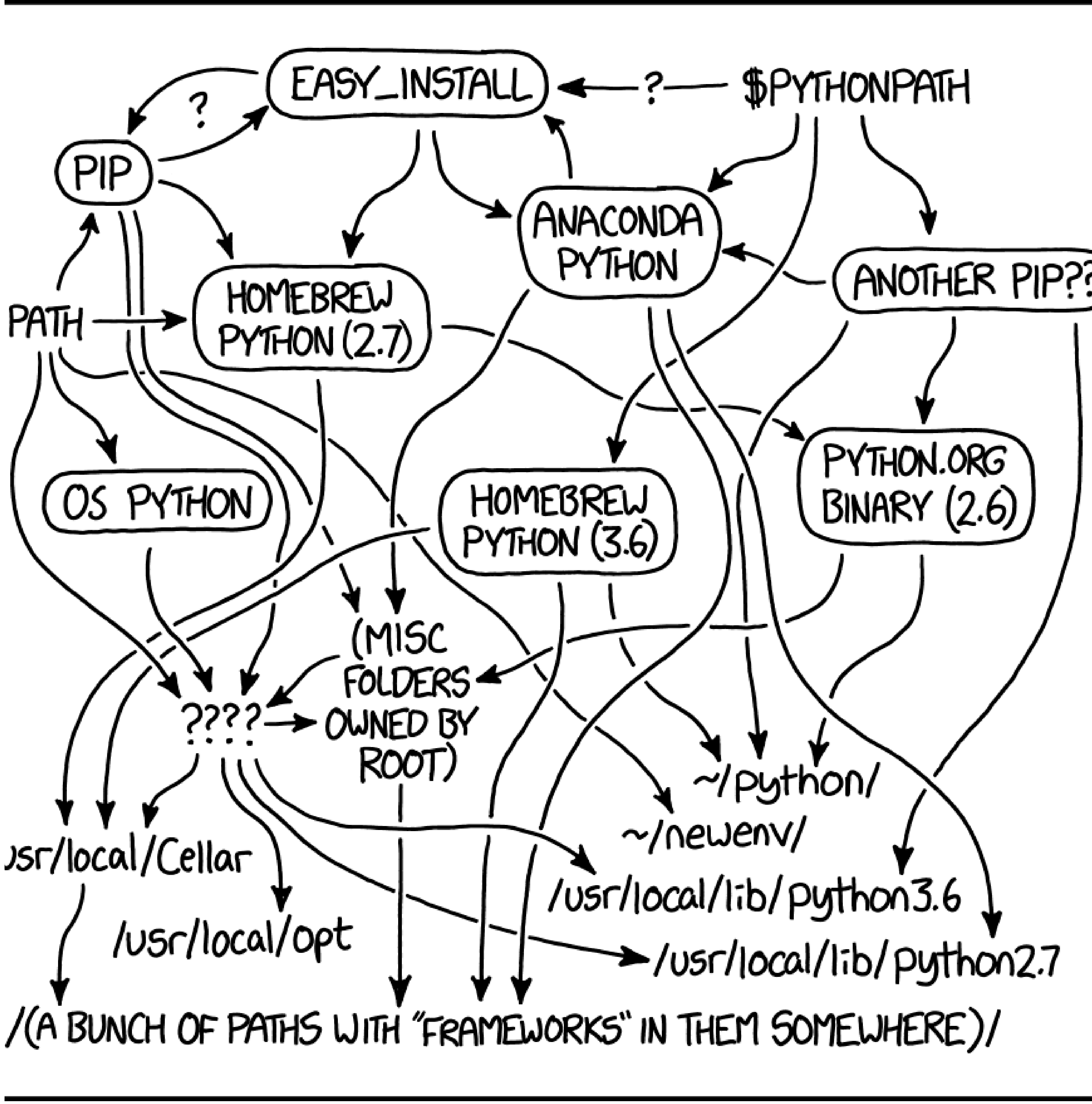
Overview

In your life as a Python developer, you'll likely create many applications that use diverse Python packages

Ideally you would like the applications to be independent of each other

- The Python packages you download for one application shouldn't interfere with the Python packages for other applications

To help you keep Python application environments isolated from each other, you can use the virtualenv tool



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

Installing virtualenv

You install virtualenv via pip as a one-off exercise as follows:

```
pip install virtualenv
```

You can test you installation as follows:

```
virtualenv --version
```

Creating a virtual environment for a project

To create a virtual environment for a particular project:

```
virtualenv MyProject
```

This command creates a folder named MyProject that contains:

- Python executable files
- A copy of the pip library, which you can use to install other packages (locally for this virtual environment)

Activating a virtual environment

To begin using a virtual environment, you must activate it

In Mac/Unix

```
source MyProject/bin/activate
```

In Windows:

```
MyProject\Scripts\activate
```

After you've activated a virtual environment, its name will appear in the command prompt.

Using virtualenv

You can now use pip to installl packages into your virtual environment.

- e.g. to install the 'request' packages:

```
pip install requests
```

This will install the package into Lib/site-packages folder.

You can write a Python script that uses the package

```python
import requests
response = requests.get('https://httpbin.org/ip')
print('Your IP is {0}'.format(response.json()['origin']))
```

And run it as normal

```
python main.py
```

Deactivating a virtual environment

You can deactivate a virtual environment as follows:

```
deactivate
```

This tears down your virtual environment

- You don’t see the packages in that virtual environment any more
- You can reactivate it whenever you need to (see 2 slides previous)