KOLT Python

Third-Party Packages

Ahmet Uysal

Monday 25th November, 2019

KOLT

Agenda

1. Installations

2. Package Management

3. Virtual Environments

4. Example: Telegram Bot

Installing Python

- Go to *python.org/downloads*
- Install the latest version of Python for your operating system
- (Windows only) Make sure to add python to the `environment variables` by checking the corresponding permission on the installation or by hand
- Check the installation by running **`python`**(Windows)/**`python3`**(macOS/Linux) in terminal.

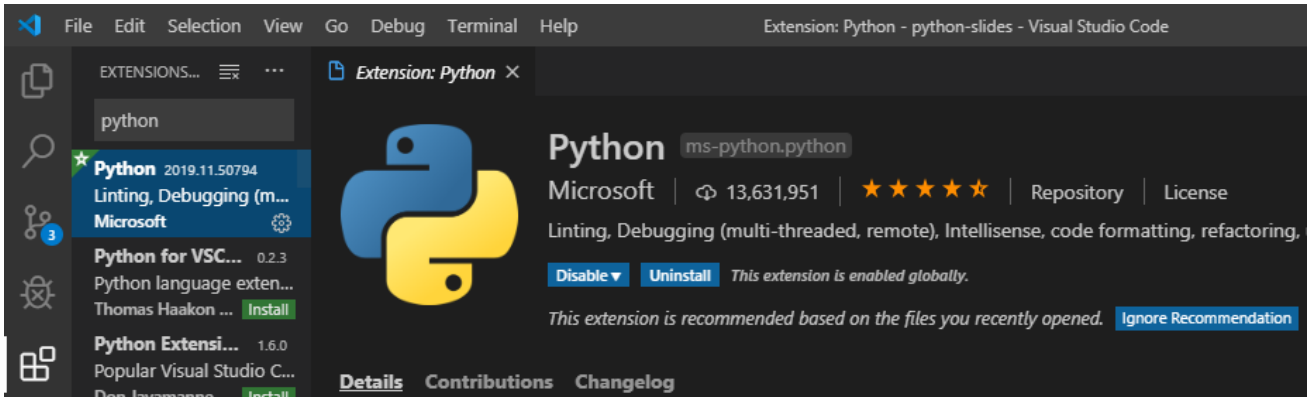
```
C:\Users\auysal>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello, world!')
Hello, world!
```

Installing an Editor/IDE(Integrated Development Environment)

- Having a specialized editor/IDE can help a lot.
- We will use Visual Studio Code but you are free to use any editor/IDE of your choice.
- Get Visual Studio Code from *code.visualstudio.com/download*

Configuring Visual Studio Code for Python

- Install Python extension for VS Code.
- Select the Python(3.8.0) Interpreter in VS Code.
- For more information, visit *VS Code Python Tutorial*.



Python Package Index (PyPI)

Repository of software for the Python programming language.

- 23,000+ Python3 packages.
- If you want a package, PyPI probably has it.

Visit *pypi.org* to explore packages.

pip

- Recommended tool for installing Python packages.
- **`pip`** is already installed with modern Python distributions.
- Try `pip -V` on your command line/terminal(`pip3 -V` for Macs).

```
$ pip -V
pip 19.2.3 from --PATH_TO_PIP-- (python 3.8)
```

Any problems?

Common pip commands

Install a package:

```
$ pip install package_name # latest version
$ pip install package_name==1.0.1 # specific version
$ pip install package_name>=1.0.1 # minimum version
```

Uninstall a package:

```
$ pip uninstall package_name
```

Update a package:

```
$ pip install --upgrade package_name
```

Search PyPI for matches:

```
$ pip search query
```

Virtual Environments

A *virtual environment* is an **isolated** Python environment that contains the Python interpreter, installed **libraries** and scripts.
Why do we need them?
What happens if two different programs use the same library?

- We might want to use different versions of the same library.
- Updating a library for **Program A** can harm another **Program B**. (*Breaking Changes*)
- We want **isolation** between programs.

Creating Virtual Environments

In Python 3.6+, the recommended way to create a virtual environment is using **venv** package, which is included in the standard installation (similar to **pip**).

Creating a virtual environment:

```
$ python -m venv /path/to/new/virtualenv
```

Activating a virtual environment:

cd(Change directory) to virtual environment folder.

In Windows: `$ Scripts\activate`

In Mac/Linux: `$ source bin/activate`

Deactivating a virtual environment:

```
$ deactivate
```

Example: Telegram Bot

- Create a virtual environment.
- Activate the virtual environment you just created.
- `$ pip install python-telegram-bot`

Create an API Token

t.me/botfather