Aim: Make a list of any 6 libraries of python with their functionalities in brief. Also,write the steps required for installing any new library in a new environment."

**1) Theano**

**Installation: pip install Theano**

Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. Theano features:

* tight integration with NumPy – Use numpy.ndarray in Theano-compiled functions.
* transparent use of a GPU – Perform data-intensive computations much faster than on a CPU.
* efficient symbolic differentiation – Theano does your derivatives for functions with one or many inputs.
* speed and stability optimizations – Get the right answer for log(1+x) even when x is really tiny.
* dynamic C code generation – Evaluate expressions faster.
* extensive unit-testing and self-verification – Detect and diagnose many types of errors.

**2) PyTorch**

**Installation: pip install pytorch**

PyTorch is a python based library built to provide flexibility as a deep learning development platform. The workflow of PyTorch is as close as you can get to python’s scientific computing library – numpy.

* Easy to use API – It is as simple as python can be.
* Python support – As mentioned above, PyTorch smoothly integrates with the python data science stack. It is so similar to numpy that you might not even notice the difference.
* Dynamic computation graphs – Instead of predefined graphs with specific functionalities, PyTorch provides a framework for us to build computational graphs as we go, and even change them during runtime. This is valuable for situations where we don’t know how much memory is going to be required for creating a neural network.
* A few other advantages of using PyTorch are it’s multiGPU support, custom data loaders and simplified preprocessors.

**3) OpenCV**

**Installation: pip install opencv-python**

* OpenCV was started at Intel in 1999 by Gary Bradsky and the first release came out in 2000.Right now, OpenCV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by-day.
* OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision.
* It can be used for Reading and Writing images, Detection of faces and its features, Detection of shapes like Circle,rectangle etc in a image,Text recognition in images, Modifying image quality and colors, Developing Augmented reality apps,etc.
* Currently OpenCV supports a wide variety of programming languages like C++, Python, Java etc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc.
* Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations.

**4) LightGBM**

**Installation: pip install lightgbm**

LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:

* Faster training speed and higher efficiency.
* Lower memory usage.
* Better accuracy.
* Support of parallel and GPU learning.
* Capable of handling large-scale data.

**5) Scikit-learn**

**Installation: pip install scikit-learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. The library is focused on modeling data. It is not focused on loading, manipulating and summarizing data. Some popular groups of models provided by scikit-learn include:

* Clustering: for grouping unlabeled data such as KMeans.
* Cross Validation: for estimating the performance of supervised models on unseen data.
* Datasets: for test datasets and for generating datasets with specific properties for investigating model behavior.
* Dimensionality Reduction: for reducing the number of attributes in data for summarization, visualization and feature selection such as Principal component analysis.
* Ensemble methods: for combining the predictions of multiple supervised models.
* Feature extraction: for defining attributes in image and text data.
* Feature selection: for identifying meaningful attributes from which to create supervised models.
* Parameter Tuning: for getting the most out of supervised models.
* Manifold Learning: For summarizing and depicting complex multi-dimensional data.
* Supervised Models: a vast array not limited to generalized linear models, discriminate analysis, naive bayes, lazy methods, neural networks, support vector machines and decision trees.

**6) Keras**

**Installation: pip install keras**

Keras is considered as one of the coolest machine learning libraries in Python. It provides an easier mechanism to express neural networks. Keras also provides some of the best utilities for compiling models, processing data-sets, visualization of graphs, and much more. In the backend, Keras uses either Theano or TensorFlow internally. Some of the most popular neural networks like CNTK can also be used. Keras is comparatively slow when we compare it with other machine learning libraries. Because it creates a computational graph by using back-end infrastructure and then makes use of it to perform operations. All the models in Keras are portable. Features Of Keras are-

* It runs smoothly on both CPU and GPU.
* Keras supports almost all the models of a neural network – fully connected, convolutional, pooling, recurrent, embedding, etc. - Furthermore, these models can be combined to build more complex models.
* Keras, being modular in nature, is incredibly expressive, flexible, and apt for innovative research.
* Keras is a completely Python-based framework, which makes it easy to debug and explore.

**Steps required to install any Python library in any given environment**

* Packages can be installed using pip or virtual environment manager: either venv for Python 3 or virtualenv for Python 2. These are the lowest-level tools for managing Python packages and are recommended if higher-level tools do not suit your needs.

**Installing pip**

pip is the reference Python package manager. It’s used to install and update packages. You’ll need to make sure you have the latest version of pip installed.

**Windows**

* The Python installers for Windows include pip. You should be able to access pip using:
* py -m pip --version

pip 9.0.1 from c:\python36\lib\site-packages (Python 3.6.1)

* You can make sure that pip is up-to-date by running:

py -m pip install --upgrade pip

**Installing virtualenv**

* If you are using Python 3.3 or newer, the venv module is the preferred way to create and manage virtual environments. venv is included in the Python standard library and requires no additional installation. If you are using venv, you may skip this section.
* virtualenv is used to manage Python packages for different projects. Using virtualenv allows you to avoid installing Python packages globally which could break system tools or other projects. You can install virtualenv using pip.
* On macOS and Linux:

python3 -m pip install --user virtualenv

* On Windows:

py -m pip install --user virtualenv

**Creating a virtual environment**

* venv (for Python 3) and virtualenv (for Python 2) allow you to manage separate package installations for different projects. They essentially allow you to create a “virtual” isolated Python installation and install packages into that virtual installation. When you switch projects, you can simply create a new virtual environment and not have to worry about breaking the packages installed in the other environments. It is always recommended to use a virtual environment while developing Python applications.
* To create a virtual environment, go to your project’s directory and run venv. If you are using Python 2, replace venv with virtualenv in the below commands.
* On macOS and Linux:

python3 -m venv env

* On Windows:

py -m venv env

* The second argument is the location to create the virtual environment. Generally, you can just create this in your project and call it env.
* venv will create a virtual Python installation in the env folder.

**Activating a virtual environment**

* Before you can start installing or using packages in your virtual environment you’ll need to activate it. Activating a virtual environment will put the virtual environment-specific python and pip executables into your shell’s PATH.
* On macOS and Linux:

source env/bin/activate

* On Windows:

.\env\Scripts\activate

**Installing packages**

* Now that you’re in your virtual environment you can install packages.
* Example: Let’s install the Requests library from the Python Package Index (PyPI): #### pip install requests

**Installing specific versions**

* pip allows you to specify which version of a package to install using version specifiers. For example, to install a specific version of requests:

pip install requests==2.18.4

* To install the latest 2.x release of requests:

pip install requests>=2.0.0,<3.0.0

* To install pre-release versions of packages, use the --pre flag:

pip install --pre requests

**Using other package indexes**

* If you want to download packages from a different index than the Python Package Index (PyPI), you can use the --index-url flag:

pip install --index-url http://index.example.com/simple/ SomeProject

* If you want to allow packages from both the Python Package Index (PyPI) and a separate index, you can use the --extra-index-url flag instead:

pip install --extra-index-url http://index.example.com/simple/ SomeProject

**Upgrading packages**

* pip can upgrade packages in-place using the --upgrade flag.
* For example, to install the latest version of requests and all of its dependencies:

pip install --upgrade requests

**Leaving the virtual environment**

* If you want to switch projects or otherwise leave your virtual environment, simply run:

deactivate