30 Days Of Python: Day 20 - PIP

Python PIP - Python Package Manager

What is PIP ?

PIP stands for Preferred installer program. We use *pip* to install different Python packages. Package is a Python module that can contain one or more modules or other packages. A module or modules that we can install to our application is a package. In programming, we do not have to write every utility program, instead we install packages and import them to our applications.

Installing PIP

If you did not install pip, let us install it now. Go to your terminal or command prompt and copy and paste this:

```
$ pip install pip
```

Check if pip is installed by writing

```
pip --version
```

```
$ pip --version
pip 21.1.3 from /usr/local/lib/python3.7/site-packages/pip (python 3.9.6)
```

As you can see, I am using pip version 21.1.3, if you see some number a bit below or above that, means you have pip installed.

Let us check some of the packages used in the Python community for different purposes. Just to let you know that there are lots of packages available for use with different applications.

Installing packages using pip

Let us try to install *numpy*, called numeric python. It is one of the most popular packages in machine learning and data science community.

- NumPy is the fundamental package for scientific computing with Python. It contains among other things:
  - a powerful N-dimensional array object
  - sophisticated (broadcasting) functions
  - tools for integrating C/C++ and Fortran code
  - useful linear algebra, Fourier transform, and random number capabilities

```
$ pip install numpy
```

Let us start using numpy. Open your python interactive shell, write python and then import numpy as follows:

```
$ python
Python 3.9.6 (default, Jun 28 2021, 15:26:21)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> numpy.version.version
'1.20.1'
>>> lst = [1, 2, 3,4, 5]
>>> np_arr = numpy.array(lst)
>>> np_arr
array([1, 2, 3, 4, 5])
>>> len(np_arr)
5
>>> np_arr * 2
array([ 2,  4,  6,  8, 10])
>>> np_arr  + 2
array([3, 4, 5, 6, 7])
>>>
```

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Let us install the big brother of numpy, *pandas*:

```
$ pip install pandas
```

```
$ python
Python 3.9.6 (default, Jun 28 2021, 15:26:21)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas
```

This section is not about numpy nor pandas, here we are trying to learn how to install packages and how to import them. If it is needed, we will talk about different packages in other sections.

Let us import a web browser module, which can help us to open any website. We do not need to install this module, it is already installed by default with Python 3. For instance if you like to open any number of websites at any time or if you like to schedule something, this *webbrowser* module can be used.

```
import webbrowser # web browser module to open websites

# list of urls: python
url_lists = [
    'http://www.python.org',
    'https://www.linkedin.com/in/Suniksha/',
    'https://github.com/Suniksha',
    'https://twitter.com/Suniksha',
]

# opens the above list of websites in a different tab
for url in url_lists:
    webbrowser.open_new_tab(url)
```

Uninstalling Packages

If you do not like to keep the installed packages, you can remove them using the following command.

```
pip uninstall packagename
```

List of Packages

To see the installed packages on our machine. We can use pip followed by list.

```
pip list
```

Show Package

To show information about a package

```
pip show packagename
```

```
$ pip show pandas
Name: pandas
Version: 1.2.3
Summary: Powerful data structures for data analysis, time series, and
statistics
Home-page: http://pandas.pydata.org
Author: None
Author-email: None
License: BSD
Location: /usr/local/lib/python3.7/site-packages
```

```
Requires: python-dateutil, pytz, numpy
Required-by:
```

If we want even more details, just add --verbose

```
$ pip show --verbose pandas
Name: pandas
Version: 1.2.3
Summary: Powerful data structures for data analysis, time series, and
statistics
Home-page: http://pandas.pydata.org
Author: None
Author-email: None
License: BSD
Location: /usr/local/lib/python3.7/site-packages
Requires: numpy, pytz, python-dateutil
Required-by:
Metadata-Version: 2.1
Installer: pip
Classifiers:
  Development Status :: 5 - Production/Stable
  Environment :: Console
  Operating System :: OS Independent
  Intended Audience :: Science/Research
  Programming Language :: Python
  Programming Language :: Python :: 3
  Programming Language :: Python :: 3.5
  Programming Language :: Python :: 3.6
  Programming Language :: Python :: 3.7
  Programming Language :: Python :: 3.8
  Programming Language :: Cython
  Topic :: Scientific/Engineering
Entry-points:
  [pandas_plotting_backends]
  matplotlib = pandas:plotting._matplotlib
```

PIP Freeze

Generate installed Python packages with their version and the output is suitable to use it in a requirements file. A requirements.txt file is a file that should contain all the installed Python packages in a Python project.

```
~$ pip freeze
docutils==0.11
Jinja2==2.7.2
MarkupSafe==0.19
Pygments==1.6
Sphinx==1.2.2
```

The pip freeze gave us the packages used, installed and their version. We use it with requirements.txt file for deployment.

Reading from URL

By now you are familiar with how to read or write on a file located on you local machine. Sometimes, we would like to read from a website using url or from an API. API stands for Application Program Interface. It is a means to exchange structured data between servers primarily as json data. To open a network connection, we need a package called *requests* - it allows to open a network connection and to implement CRUD(create, read, update and delete) operations. In this section, we will cover only reading ore getting part of a CRUD.

Let us install *requests*:

```
$ pip install requests
```

We will see *get*, *status_code*, *headers*, *text* and *json* methods in *requests* module:

- *get()*: to open a network and fetch data from url - it returns a response object
- *status_code*: After we fetched data, we can check the status of the operation (success, error, etc)
- *headers*: To check the header types
- *text*: to extract the text from the fetched response object
- *json*: to extract json data Let's read a txt file from this website, https://www.w3.org/TR/PNG/iso_8859-1.txt.

```
import requests # importing the request module

url = 'https://www.w3.org/TR/PNG/iso_8859-1.txt' # text from a website

response = requests.get(url) # opening a network and fetching a data
print(response)
print(response.status_code) # status code, success:200
print(response.headers)     # headers information
print(response.text) # gives all the text from the page
```

```
<Response [200]>
200
{'date': 'Sun, 08 Dec 2019 18:00:31 GMT', 'last-modified': 'Fri, 07 Nov
2003 05:51:11 GMT', 'etag': '"17e9-3cb82080711c0;50c0b26855880-gzip"',
'accept-ranges': 'bytes', 'cache-control': 'max-age=31536000', 'expires':
'Mon, 07 Dec 2020 18:00:31 GMT', 'vary': 'Accept-Encoding', 'content-
encoding': 'gzip', 'access-control-allow-origin': '*', 'content-length':
'1616', 'content-type': 'text/plain', 'strict-transport-security': 'max-
age=15552000; includeSubdomains; preload', 'content-security-policy':
'upgrade-insecure-requests'}
```

- Let us read from an API. API stands for Application Program Interface. It is a means to exchange structure data between servers primarily a json data. An example of an API:https://restcountries.eu/rest/v2/all. Let us read this API using *requests* module.

```
import requests
url = 'https://restcountries.eu/rest/v2/all'  # countries api
response = requests.get(url)  # opening a network and fetching a data
```

```
print(response) # response object
print(response.status_code)  # status code, success:200
countries = response.json()
print(countries[:1])  # we sliced only the first country, remove the
slicing to see all countries
```

```
<Response [200]>
200
[{'alpha2Code': 'AF',
  'alpha3Code': 'AFG',
  'altSpellings': ['AF', 'Afġānistān'],
  'area': 652230.0,
  'borders': ['IRN', 'PAK', 'TKM', 'UZB', 'TJK', 'CHN'],
  'callingCodes': ['93'],
  'capital': 'Kabul',
  'cioc': 'AFG',
  'currencies': [{'code': 'AFN', 'name': 'Afghan afghani', 'symbol': '؋'}],
  'demonym': 'Afghan',
  'flag': 'https://restcountries.eu/data/afg.svg',
  'gini': 27.8,
  'languages': [{'iso639_1': 'ps',
                 'iso639_2': 'pus',
                 'name': 'Pashto',
                 'nativeName': 'پښتو'},
                {'iso639_1': 'uz',
                 'iso639_2': 'uzb',
                 'name': 'Uzbek',
                 'nativeName': 'Oʻzbek'},
                {'iso639_1': 'tk',
                 'iso639_2': 'tuk',
                 'name': 'Turkmen',
                 'nativeName': 'Türkmen'}],
  'latlng': [33.0, 65.0],
  'name': 'Afghanistan',
  'nativeName': 'افغانستان',
  'numericCode': '004',
  'population': 27657145,
  'region': 'Asia',
  'regionalBlocs': [{'acronym': 'SAARC',
                     'name': 'South Asian Association for Regional
Cooperation',
                     'otherAcronyms': [],
                     'otherNames': []}],
  'subregion': 'Southern Asia',
  'timezones': ['UTC+04:30'],
  'topLevelDomain': ['.af'],
  'translations': {'br': 'Afeganistão',
                   'de': 'Afghanistan',
                   'es': 'Afganistán',
                   'fa': 'افغانستان',
                   'fr': 'Afghanistan',
                   'hr': 'Afganistan',
                   'it': 'Afghanistan',
                   'ja': 'アフガニスタン',
                   'nl': 'Afghanistan',
                   'pt': 'Afeganistão'}}]
```

We use *json()* method from response object, if the we are fetching JSON data. For txt, html, xml and other file formats we can use *text*.

Creating a Package

We organize a large number of files in different folders and sub-folders based on some criteria, so that we can find and manage them easily. As you know, a module can contain multiple objects, such as classes, functions, etc. A package can contain one or more relevant modules. A package is actually a folder containing one or more module files. Let us create a package named mypackage, using the following steps:

Create a new folder named mypacakge inside 30DaysOfPython folder Create an empty **init**.py file in the mypackage folder. Create modules arithmetic.py and greet.py with following code:

```
# mypackage/arithmetics.py
# arithmetics.py
def add_numbers(*args):
    total = 0
    for num in args:
        total += num
    return total


def subtract(a, b):
    return (a - b)


def multiple(a, b):
    return a * b


def division(a, b):
    return a / b


def remainder(a, b):
    return a % b


def power(a, b):
    return a ** b
```

```
# mypackage/greet.py
# greet.py
def greet_person(firstname, lastname):
    return f'{firstname} {lastname}, welcome to 30DaysOfPython Challenge!'
```

The folder structure of your package should look like this:

```
─ mypackage
    ├── __init__.py
    ├── arithmetic.py
    └── greet.py
```

Now let's open the python interactive shell and try the package we have created:

```
~/Desktop/30DaysOfPython$ python
Python 3.9.6 (default, Jun 28 2021, 15:26:21)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
```

```
Type "help", "copyright", "credits" or "license" for more information.
>>> from mypackage import arithmetics
>>> arithmetics.add_numbers(1, 2, 3, 5)
11
>>> arithmetics.subtract(5, 3)
2
>>> arithmetics.multiple(5, 3)
15
>>> arithmetics.division(5, 3)
1.6666666666666667
>>> arithmetics.remainder(5, 3)
2
>>> arithmetics.power(5, 3)
125
>>> from mypackage import greet
>>> greet.greet_person('Suniksha', 'Patel')
'Suniksha Patel, welcome to 30DaysOfPython Challenge!'
>>>
```

As you can see our package works perfectly. The package folder contains a special file called **init**.py - it stores the package's content. If we put **init**.py in the package folder, python start recognizes it as a package. The **init**.py exposes specified resources from its modules to be imported to other python files. An empty **init**.py file makes all functions available when a package is imported. The **init**.py is essential for the folder to be recognized by Python as a package.

Further Information About Packages

- Database
    - SQLAlchemy or SQLObject - Object oriented access to several different database systems
        - *pip install SQLAlchemy*
- Web Development
    - Django - High-level web framework.
        - *pip install django*
    - Flask - micro framework for Python based on Werkzeug, Jinja 2. (It's BSD licensed)
        - *pip install flask*
- HTML Parser
    - Beautiful Soup - HTML/XML parser designed for quick turnaround projects like screen-scraping, will accept bad markup.
        - *pip install beautifulsoup4*
    - PyQuery - implements jQuery in Python; faster than BeautifulSoup, apparently.

- XML Processing
    - ElementTree - The Element type is a simple but flexible container object, designed to store hierarchical data structures, such as simplified XML infosets, in memory. --Note: Python 2.5 and up has ElementTree in the Standard Library
- GUI
    - PyQt - Bindings for the cross-platform Qt framework.
    - TkInter - The traditional Python user interface toolkit.
- Data Analysis, Data Science and Machine learning
    - Numpy: Numpy(numeric python) is known as one of the most popular machine learning library in Python.
    - Pandas: is a data analysis, data science and a machine learning library in Python that provides data structures of high-level and a wide variety of tools for analysis.
    - SciPy: SciPy is a machine learning library for application developers and engineers. SciPy library contains modules for optimization, linear algebra, integration, image processing, and statistics.
    - Scikit-Learn: It is NumPy and SciPy. It is considered as one of the best libraries for working with complex data.
    - TensorFlow: is a machine learning library built by Google.
    - 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.
- Network:
    - requests: is a package which we can use to send requests to a server(GET, POST, DELETE, PUT)
        - *pip install requests*

🌕 You are always progressing and you are a head of 20 steps to your way to greatness. Now do some exercises for your brain and muscles.

Exercises: Day 20

1. Read this url and find the 10 most frequent words. romeo_and_juliet = 'http://www.gutenberg.org/files/1112/1112.txt'

2. Read the cats API and cats_api = 'https://api.thecatapi.com/v1/breeds' and find :
   i. the min, max, mean, median, standard deviation of cats' weight in metric units.
   ii. the min, max, mean, median, standard deviation of cats' lifespan in years.
   iii. Create a frequency table of country and breed of cats
3. Read the countries API and find
   i. the 10 largest countries
   ii. the 10 most spoken languages
   iii. the total number of languages in the countries API
4. UCI is one of the most common places to get data sets for data science and machine learning. Read the content of UCL (https://archive.ics.uci.edu/ml/datasets.php). Without additional libraries it will be difficult, so you may try it with BeautifulSoup4

🎉 CONGRATULATIONS ! 🎉