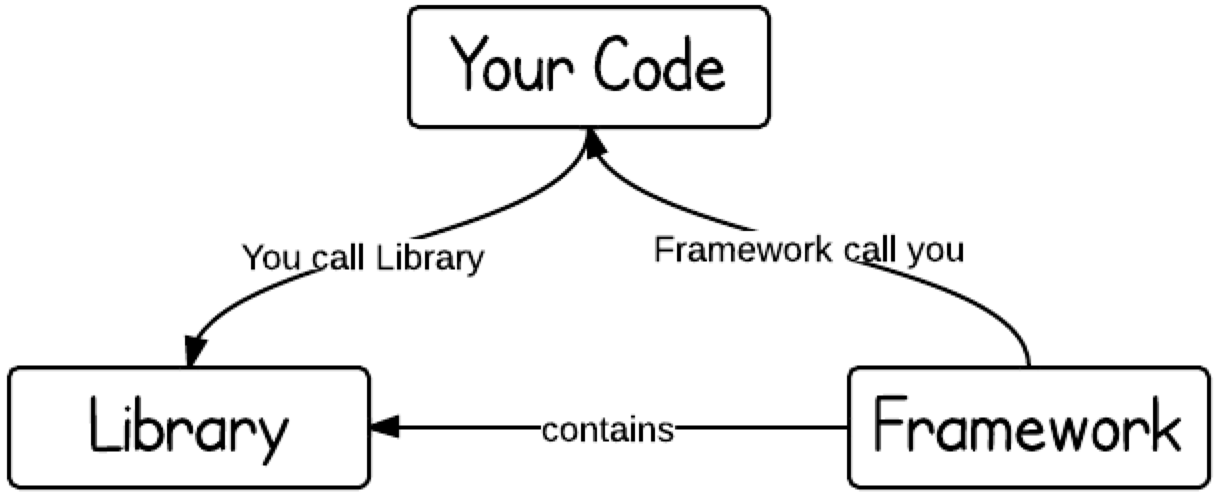
Python Modules and Packages

Introduction

- A *library* is basically a collection of programs (or parts of programs, if you want). Sometimes it’s larger, sometimes it’s smaller. Basically a bunch of handy code someone’s written for you, which you can use in your programs.
- A *framework* is usually a bunch of libraries, and sometimes (but not always) some external utilities, all geared towards solving a particular task.
- A *package* is a method of *isolating* (and often *distributing*) the code. There might be a library package. There might be a framework package. So packages can’t be really compared to libraries or frameworks, since that’s not even apples and oranges, more like apples and cardboard boxes.

Library and Framework



Why Modules used??

- Reusability
- As your program grows more in the size you may want to split it into several files for easier maintenance as well as reusability of the code.
- You can define your most used functions in a module and import it, instead of copying their definitions into different programs.
- A module can be imported by another program to make use of its functionality. This is how you can use the Python standard library as well.
- Simply put, a module is a file consisting of Python code. It can define functions, classes, and variables, and can also include runnable code.
- Any Python file can be referenced as a module. A file containing Python code, for example: test.py, is called a module, and its name would be test.

Writing and Importing Modules

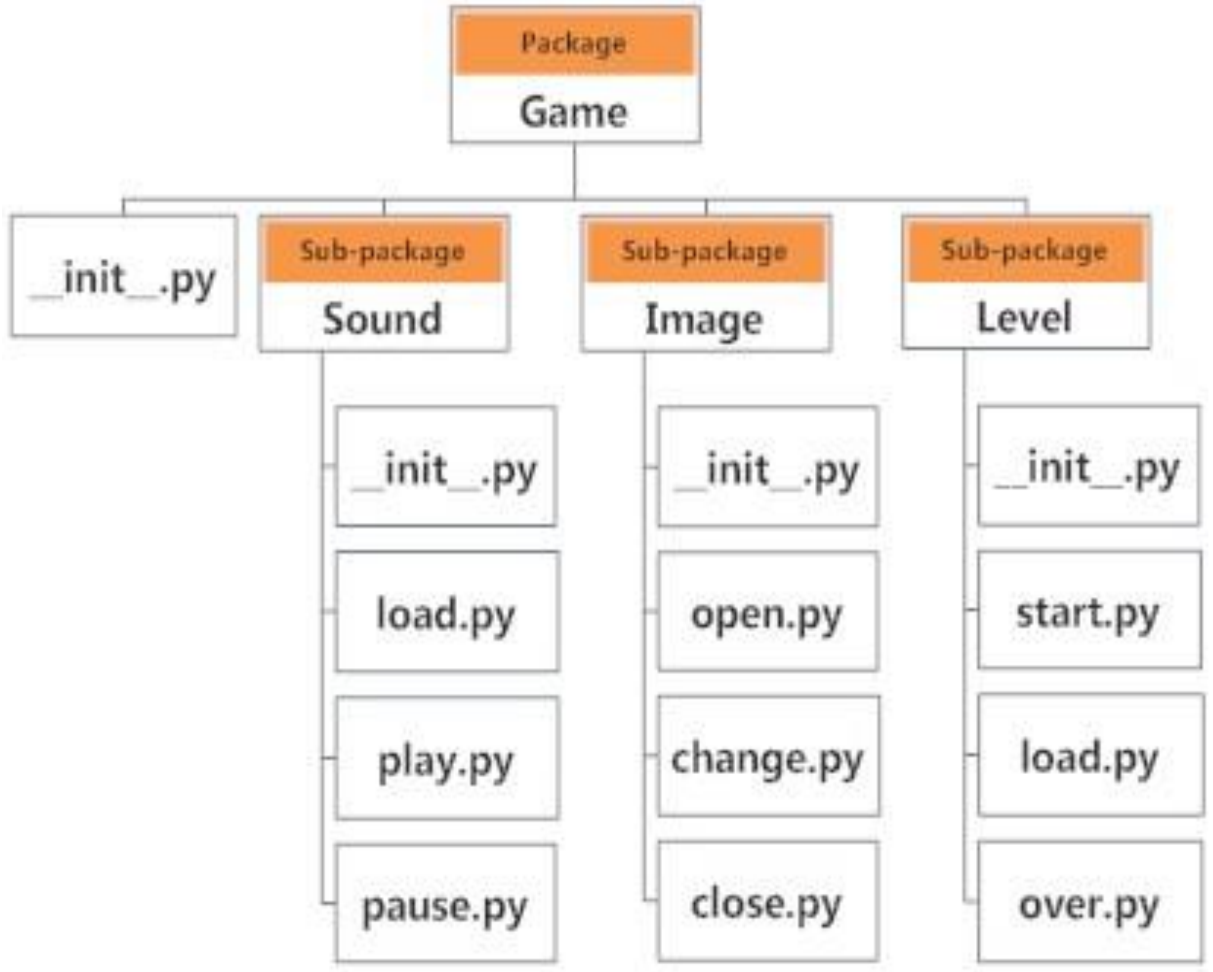
- Creating a new python program which should contains Variables, Functions and classes
- If we run the program on the command line with python hello.py nothing will happen since we have not told the program to do anything.
- We need to create a second file and This file needs to be in the same directory so that Python knows where to find the module since it’s not a built-in module.

Keyword : import

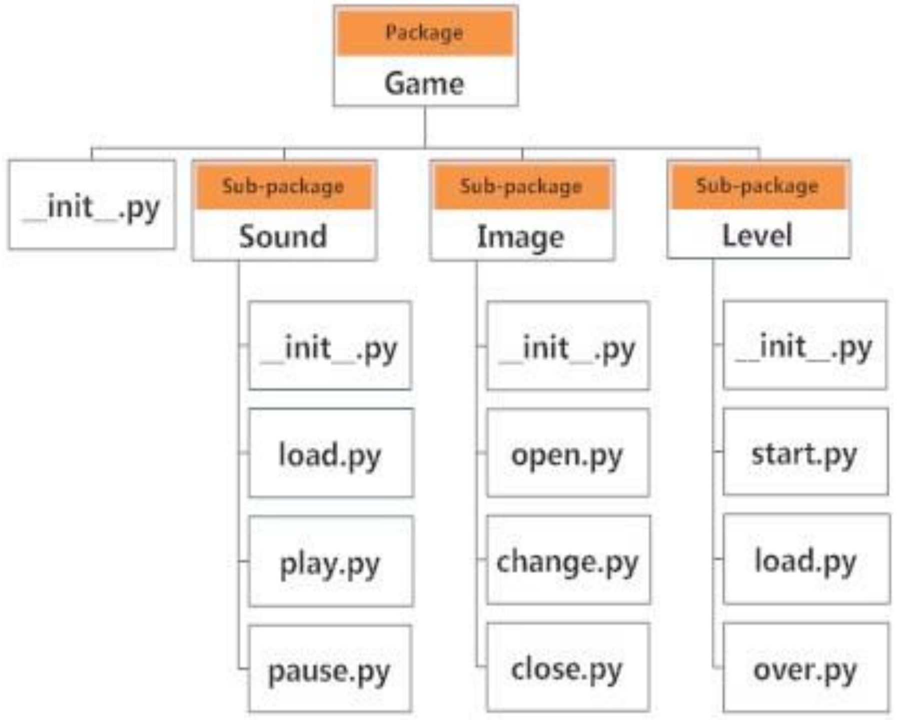
- import is a keyword which is used to import one python file or package in to another module
- import framework_name
  - the from...import Statement
    - The statement is used to get particular package from framework
    - For Example: from tkinter import messagebox
      - Tkinter in a framework for Desktop application
      - Messagebox is a sub package of tkinter
  - The from...import * Statement
    - The from tkinter import * which indicates all framework from python

Absolute and Relative Import

- Absolute Direct import from file
- Relative from parent Directory, importing the Function
- Lets Assume you have folder like following
  - sound/load.py contains a function, function1.
  - Image/__init__.py contains a class, class1.
  - Level/start/module5.py contains a function, function2.



Absolute Import



- **Following are Absolute Import Syntax**
  - from sound import load
  - from sound.play import function
  - from image import class1
- **Pros:**
  - clear and straightforward.
  - It is easy to tell exactly where the imported statement
- **Cons**
  - Difficult to handle the Multiple Subpackages

Relative Import

- A relative import specifies the resource to be imported relative to the current location.
- **There are two types of relative imports:**
  - *implicit and explicit*. Implicit relative imports have been deprecated in Python3
- **Syntax**:
  - from .some_module import some_class (it is in same Dir)
  - from ..some_package import some_function (Parent Dir in same Location)
  - from . import some_class
- **Pros:** it is short(less code)
- **Cons:** it will be messy while importing

Locating Modules

- When you import a module, the Python interpreter searches for the module in the following sequences The current directory.
- If the module isn't found, Python then searches each directory in the shell variable PYTHONPATH.

```
import os
os.getcwd()
```

- If all else fails, Python checks the default path. On UNIX, this default path is normally /usr/local/lib/python/.
- The module search path is stored in the system module sys as the **sys.path** variable.The sys.path variable contains the current directory, PYTHONPATH, and the installation-dependent default.

Installing Modules or Framework or API

- PIP is a recursive acronym that stands for “PIP Installs Packages” or “Preferred Installer Program”. It’s a command-line utility that allows you to install, reinstall, or uninstall PyPI packages with a simple and straightforward command: pip.

- Command to install framework in windows:

  pip install Frameworkname

- To install in Linux: **(Python 3.x)**

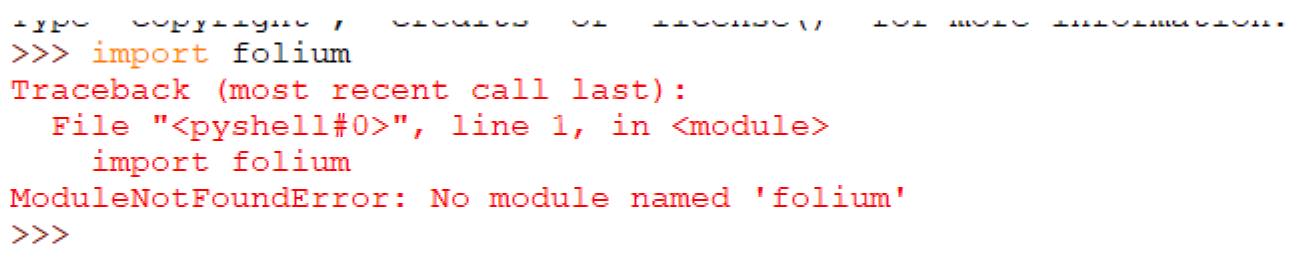
  sudo apt-get install python3-pip

Folium Introduction

- Folium is a Framework which is used to get **street map** in your python

Module not found Error

- When you are getting module not found error, then the particular framework is not available. We have to install it

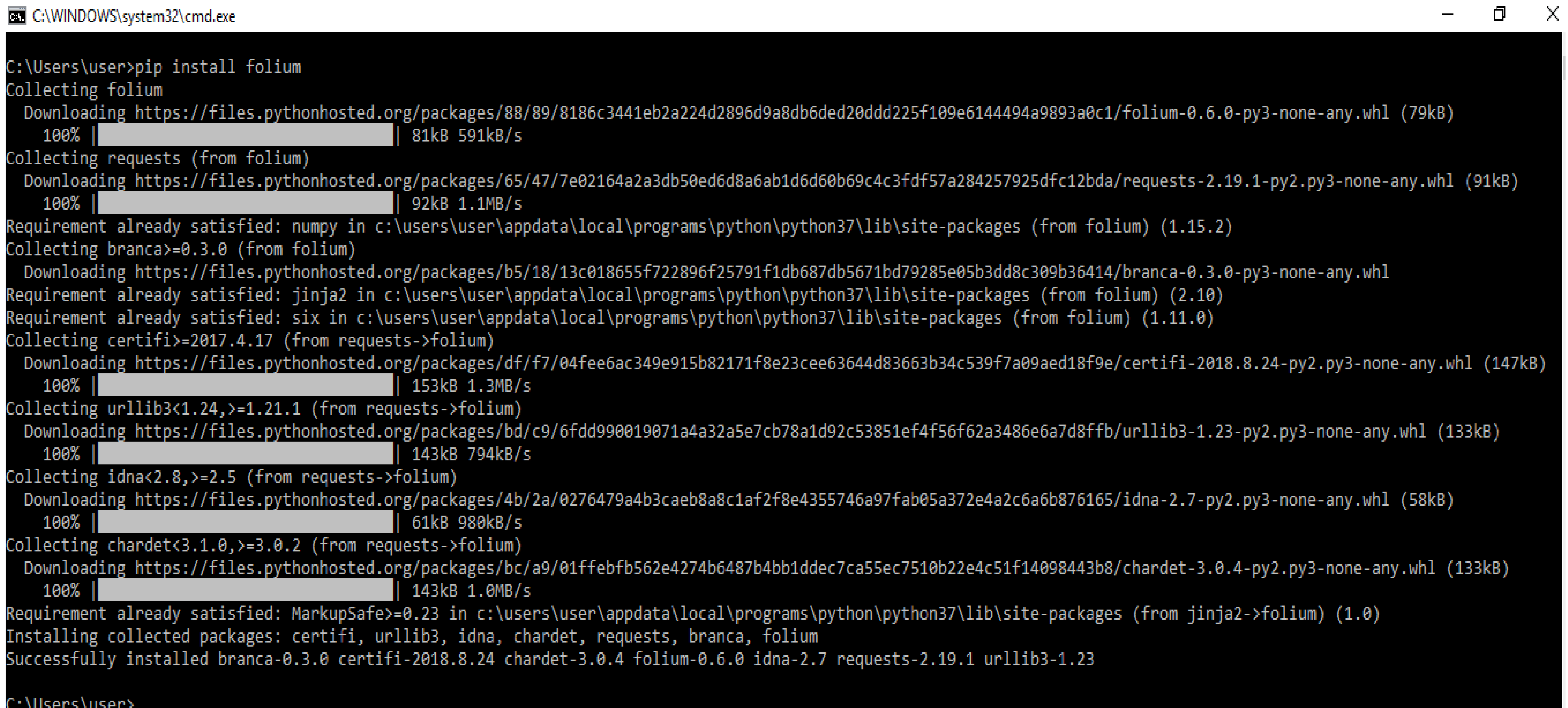


```
Type "copyright", "credits" or "license()" for more information.
>>> import folium
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    import folium
ModuleNotFoundError: No module named 'folium'
>>>
```

- To install open Command Prompt and have to give *pip install folium*
- **Internet connection required**

Installing Folium

Python will collect required and similar packages from internet , after it will extract and save it in to site

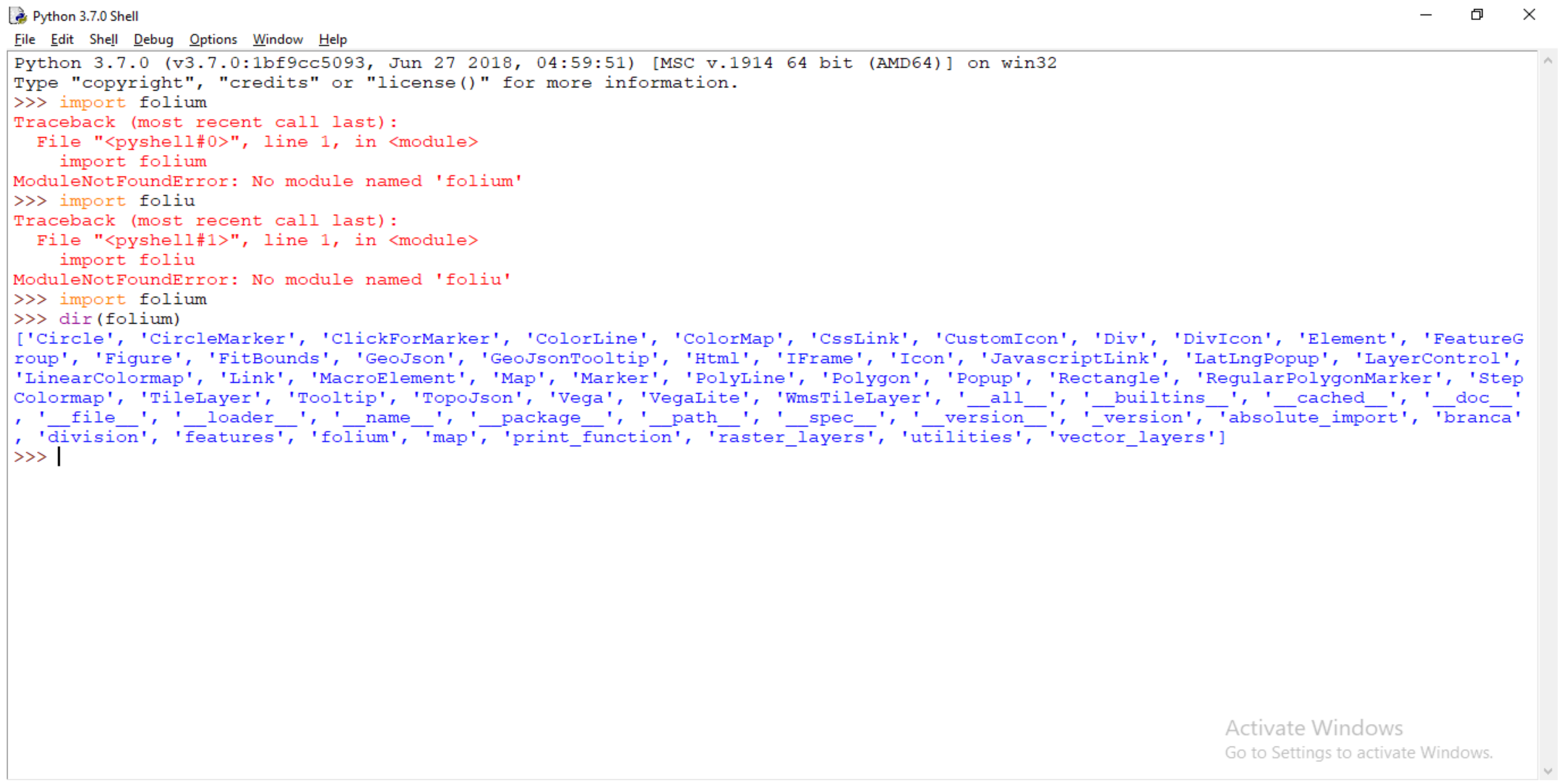


```
C:\Users\user>pip install folium
Collecting folium
  Downloading https://files.pythonhosted.org/packages/88/89/8186c3441eb2a224d2896d9a8db6ded20ddd225f109e6144494a9893a0c1/folium-0.6.0-py3-none-any.whl (79kB)
    100% |████████████████████████████████| 81kB 591kB/s
Collecting requests (from folium)
  Downloading https://files.pythonhosted.org/packages/65/47/7e02164a2a3db50ed6d8a6ab1d6d60b69c4c3fdf57a284257925dfc12bda/requests-2.19.1-py2.py3-none-any.whl (91kB)
    100% |████████████████████████████████| 92kB 1.1MB/s
Requirement already satisfied: numpy in c:\users\user\appdata\local\programs\python\python37\lib\site-packages (from folium) (1.15.2)
Collecting branca>=0.3.0 (from folium)
  Downloading https://files.pythonhosted.org/packages/b5/18/13c018655f722896f25791f1db687db5671bd79285e05b3dd8c309b36414/branca-0.3.0-py3-none-any.whl
Requirement already satisfied: jinja2 in c:\users\user\appdata\local\programs\python\python37\lib\site-packages (from folium) (2.10)
Requirement already satisfied: six in c:\users\user\appdata\local\programs\python\python37\lib\site-packages (from folium) (1.11.0)
Collecting certifi>=2017.4.17 (from requests->folium)
  Downloading https://files.pythonhosted.org/packages/df/f7/04fee6ac349e915b82171f8e23cee63644d83663b34c539f7a09aed18f9e/certifi-2018.8.24-py2.py3-none-any.whl (147kB)
    100% |████████████████████████████████| 153kB 1.3MB/s
Collecting urllib3<1.24,>=1.21.1 (from requests->folium)
  Downloading https://files.pythonhosted.org/packages/bd/c9/6fdd990019071a4a32a5e7cb78a1d92c53851ef4f56f62a3486e6a7d8ffb/urllib3-1.23-py2.py3-none-any.whl (133kB)
    100% |████████████████████████████████| 143kB 794kB/s
Collecting idna<2.8,>=2.5 (from requests->folium)
  Downloading https://files.pythonhosted.org/packages/4b/2a/0276479a4b3caeb8a8c1af2f8e4355746a97fab05a372e4a2c6a6b876165/idna-2.7-py2.py3-none-any.whl (58kB)
    100% |████████████████████████████████| 61kB 980kB/s
Collecting chardet<3.1.0,>=3.0.2 (from requests->folium)
  Downloading https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8/chardet-3.0.4-py2.py3-none-any.whl (133kB)
    100% |████████████████████████████████| 143kB 1.0MB/s
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\user\appdata\local\programs\python\python37\lib\site-packages (from jinja2->folium) (1.0)
Installing collected packages: certifi, urllib3, idna, chardet, requests, branca, folium
Successfully installed branca-0.3.0 certifi-2018.8.24 chardet-3.0.4 folium-0.6.0 idna-2.7 requests-2.19.1 urllib3-1.23

C:\Users\user>
```

dir() function

- The dir() is a inbuild function which is used to find out the available function to apply in modules or frameworks



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import folium
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    import folium
ModuleNotFoundError: No module named 'folium'
>>> import foliu
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    import foliu
ModuleNotFoundError: No module named 'foliu'
>>> import folium
>>> dir(folium)
['Circle', 'CircleMarker', 'ClickForMarker', 'ColorLine', 'ColorMap', 'CssLink', 'CustomIcon', 'Div', 'DivIcon', 'Element', 'FeatureG
roup', 'Figure', 'FitBounds', 'GeoJson', 'GeoJsonTooltip', 'Html', 'IFrame', 'Icon', 'JavascriptLink', 'LatLngPopup', 'LayerControl',
'LinearColormap', 'Link', 'MacroElement', 'Map', 'Marker', 'PolyLine', 'Polygon', 'Popup', 'Rectangle', 'RegularPolygonMarker', 'Step
Colormap', 'TileLayer', 'Tooltip', 'TopoJson', 'Vega', 'VegaLite', 'WmsTileLayer', '__all__', '__builtins__', '__cached__', '__doc__'
, '__file__', '__loader__', '__name__', '__package__', '__path__', '__spec__', '__version__', '_version', 'absolute_import', 'branca'
, 'division', 'features', 'folium', 'map', 'print_function', 'raster_layers', 'utilities', 'vector_layers']
>>> 
```

Folium Framework

- import folium
- map=folium.Map(location=[45,-121],zoom_start=20)
- map.save(outfile=“mapp11.html”)

Framework

- Folium (Module or Framework)
  - Map
  - Location
  - Figure

Packages

Great Job

Next Topic: anonymous Function