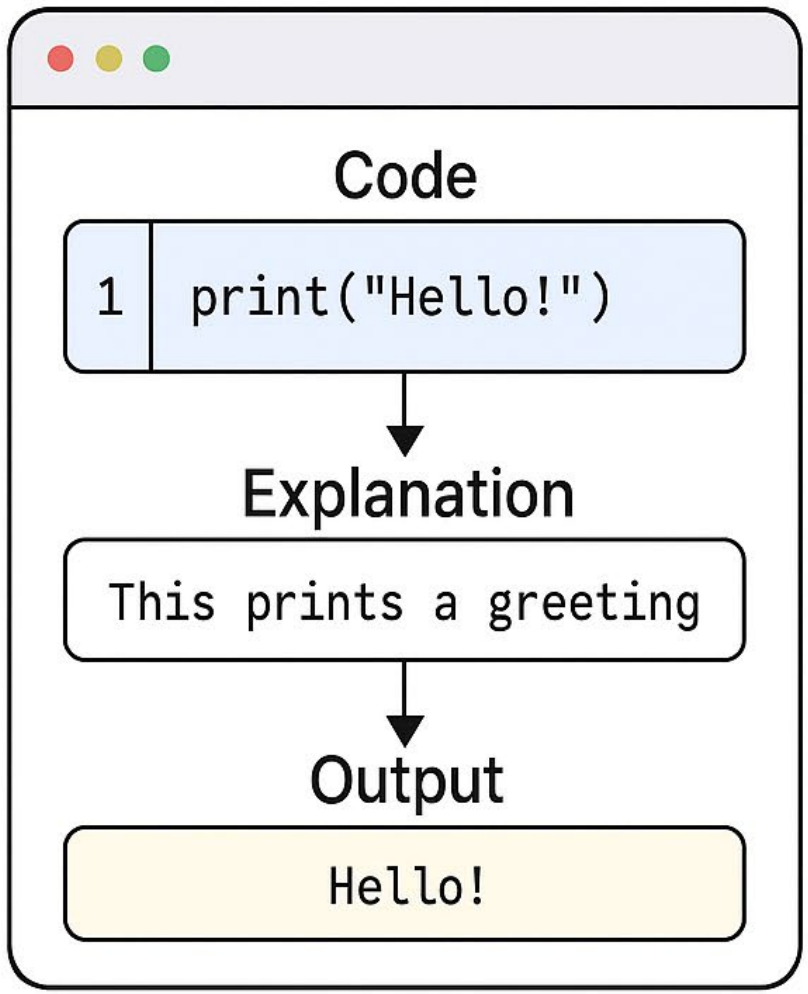
An Introduction to Jupyter Notebooks

Haroon Malik

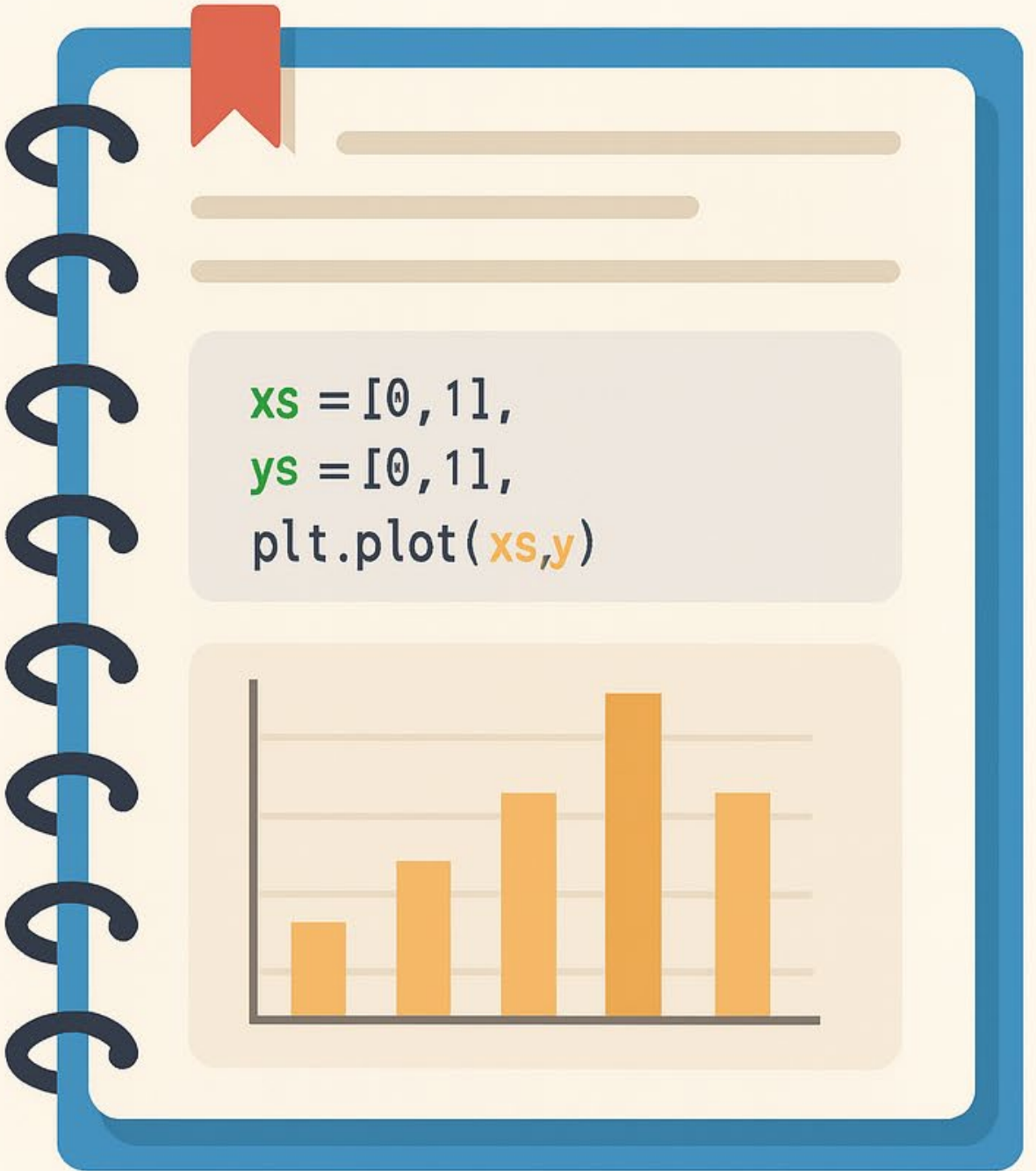
What is Jupyter Notebook?



Jupyter Notebook = Code + Explanation + Output in one place

Perfect for learning, data analysis, and sharing ideas!

"Your coding lab notebook – where code meets explanation and output in real-time."



What Makes Jupyter Notebook Different & Better for Learning

Feature	Jupyter Notebook	Traditional IDE (e.g., PyCharm)
Coding + Explanation	✅ Together	❌ Separate
Easy to Run & Edit	✅ Cell-by-cell	❌ Full script execution
Great for Visualization	✅ In-line plots	⚠️ Requires config
Learning Curve	🟢 Very low	🔴 Higher for beginners
Documenting Work	✅ Markdown Supported	❌ Not integrated


Software Installation

Anaconda | The World's Most Po

anaconda.com

Marshall | GW Open Science... | Chapter 7 Usage ca... | Jupyter Notebook T... | Jupyter Notebook T... | Data Visualization u... | Quick introduction... | Home Page - Select...

ANACONDA.

Products | Pricing | Solutions | Resources | Blog | Company | Get Started

Data science technology for human sensemaking.

A movement that brings together millions of data science practitioners, data-driven enterprises, and the open source community.

Get Started

Anaconda

- **Anaconda distribution** comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, **Anaconda Navigator**,[13] as a graphical alternative to the command line interface (CLI).

Anaconda

Anaconda is a conditional free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012

Difference

- The big difference between conda and the pip package manager is in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists.
- When pip installs a package, it automatically installs any dependent Python packages without checking if these conflict with previously installed packages. It will install a package and any of its dependencies regardless of the state of the existing installation

Anaconda Navigator

File Help

ANACONDA NAVIGATOR

Sign in to Anaconda Cloud

Home

Environments

Learning

Community

Applications on base (root) Channels Refresh

JupyterLab
0.35.3
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch

Notebook
5.7.4
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

Orange 3
3.17.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Launch

Qt Console
↗ 4.4.3
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch

Spyder
3.3.2
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch

VS Code
1.33.1
Streamlined code editor with support for development operations like debugging, task running and version control.
Launch

Glueviz
0.13.3
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install

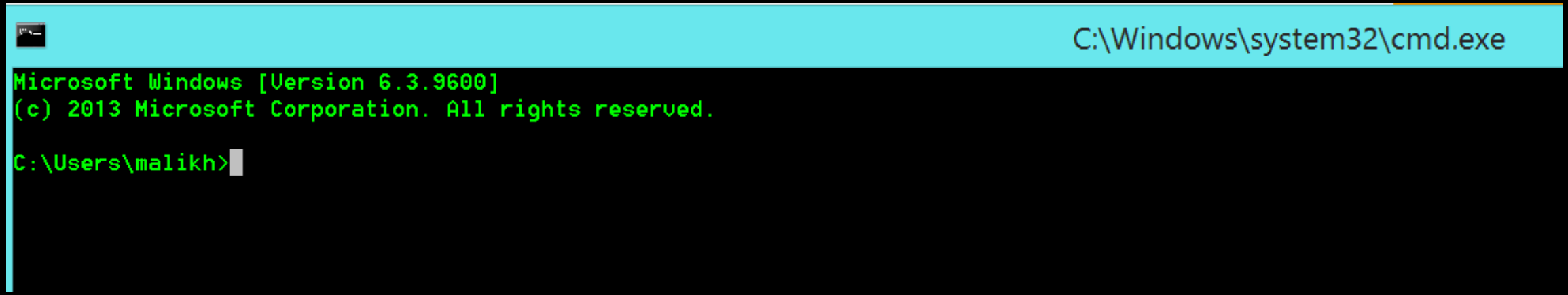
RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.
Install

Documentation

Developer Blog

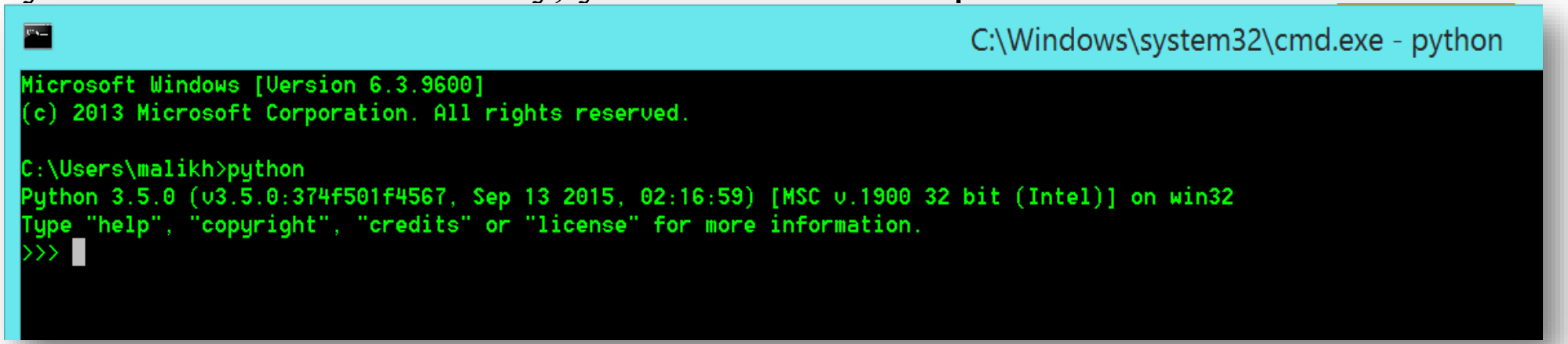
For windows – Go to Command prompt

- To open a windows command prompt, press the “**Windows Key+R**” to open a “Run” dialog box. Next, type in “**cmd**”, and then click “**OK**”. This open a normal Command Prompt.
- To open a command prompt as an administrator, press the “**Windows Key+R**” to open a “**Run**” dialog box, then type “**cmd**” and then press **Ctrl+Shift+Enter** to open an administrator Command Prompt.



Confirm That Python is Installed

The simplest way to test for a Python installation on your Windows server is to open a command prompt. Once a command prompt window opens, type **python** and press Enter. If Python is installed correctly, you should see output similar to what is shown below.



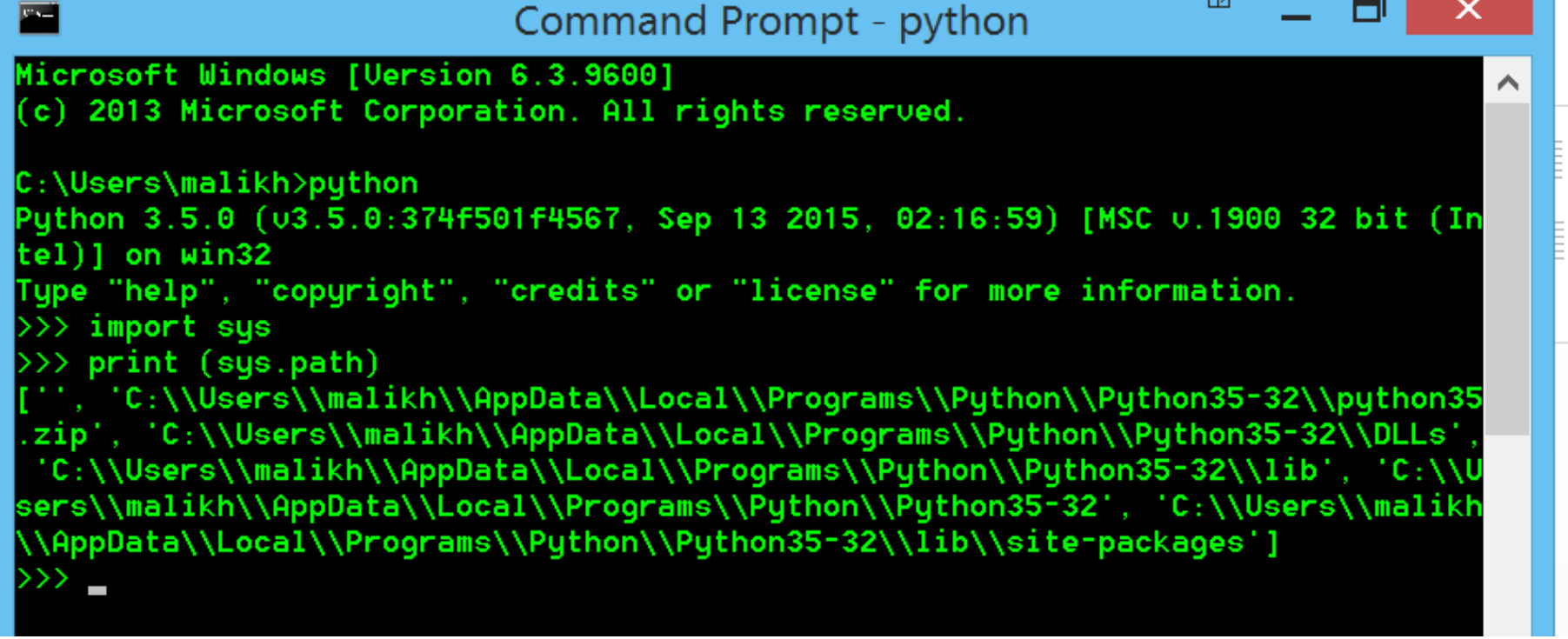
If you receive a message like:

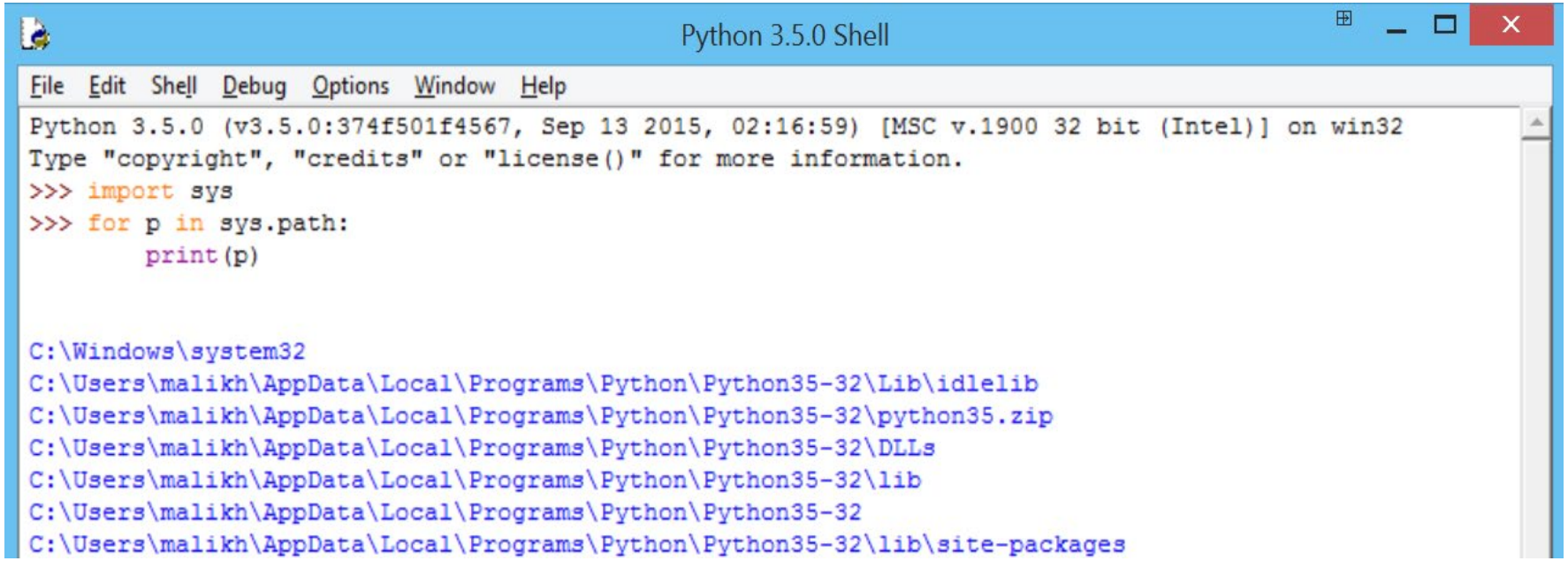
Python is not recognized as an internal or external command, operable program or batch file.

Python is either not installed or the system variable path has not been set. You will need to either launch Python from the folder in which it is installed or adjust your system variables to allow Python to be launched from any location. For more information about installing and using Python, see the article on how to install python on Windows.

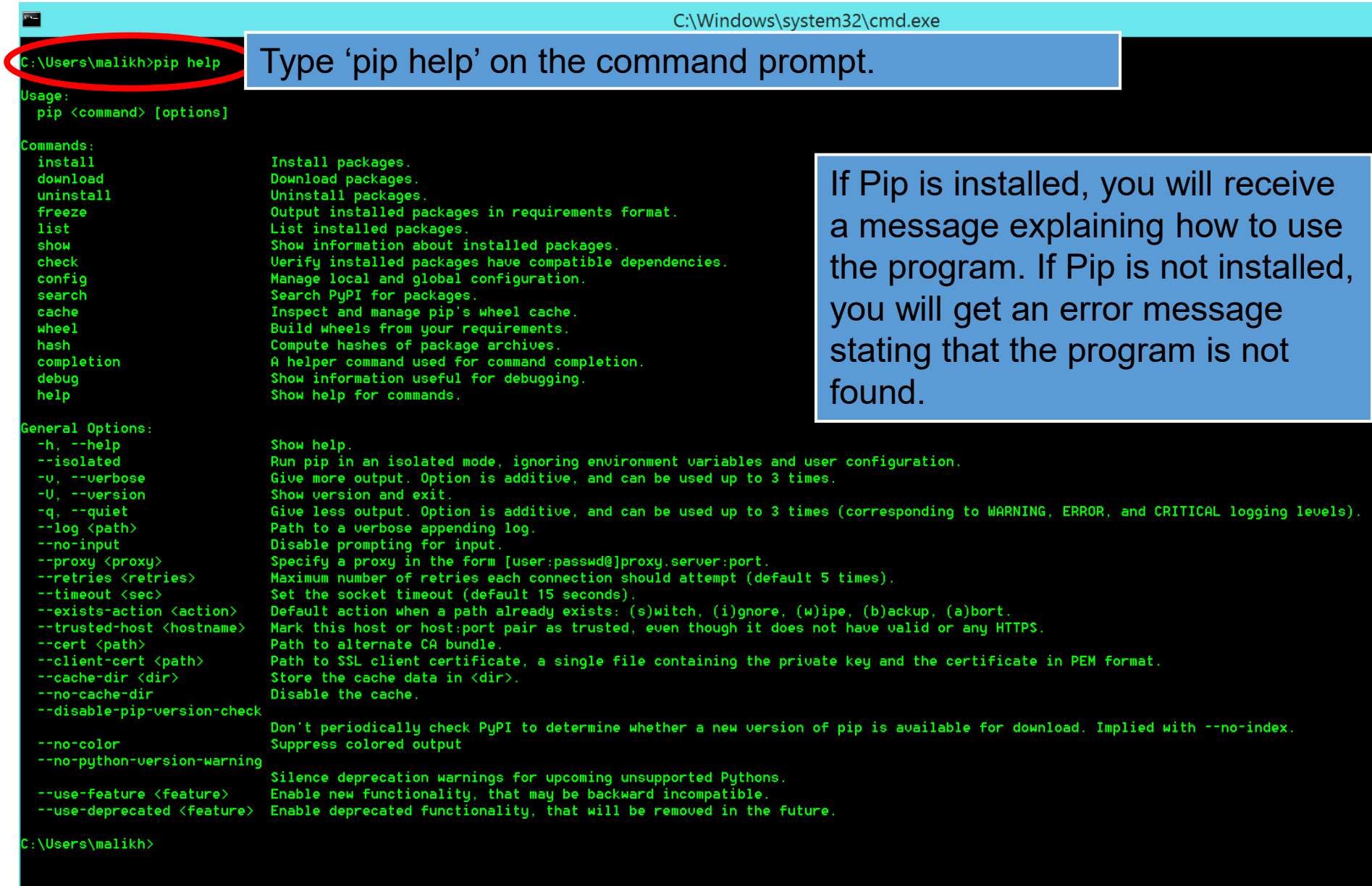
Where is Python Installed on MY machine????

- Depends where you installed it ☺
- However, there is a quick way to find it out


Command Prompt - python
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Users\malikh>python
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:16:59) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> print (sys.path)
['', 'C:\\Users\\malikh\\AppData\\Local\\Programs\\Python\\Python35-32\\python35.zip', 'C:\\Users\\malikh\\AppData\\Local\\Programs\\Python\\Python35-32\\DLLs', 'C:\\Users\\malikh\\AppData\\Local\\Programs\\Python\\Python35-32\\lib', 'C:\\Users\\malikh\\AppData\\Local\\Programs\\Python\\Python35-32', 'C:\\Users\\malikh\\AppData\\Local\\Programs\\Python\\Python35-32\\lib\\site-packages']
>>>


Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:16:59) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import sys
>>> for p in sys.path:
print(p)
C:\Windows\system32
C:\Users\malikh\AppData\Local\Programs\Python\Python35-32\Lib\idlelib
C:\Users\malikh\AppData\Local\Programs\Python\Python35-32\python35.zip
C:\Users\malikh\AppData\Local\Programs\Python\Python35-32\DLLs
C:\Users\malikh\AppData\Local\Programs\Python\Python35-32\lib
C:\Users\malikh\AppData\Local\Programs\Python\Python35-32
C:\Users\malikh\AppData\Local\Programs\Python\Python35-32\lib\site-packages

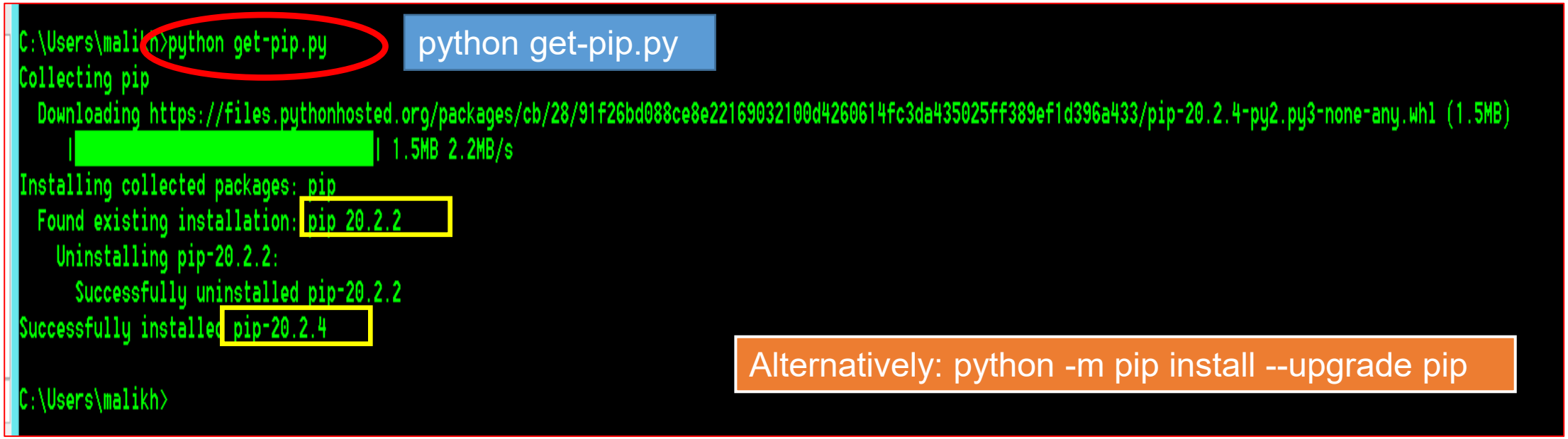
Check if PIP is already installed on your machine



Installing Pip on Windows

get-pip.py

https://bootstrap.pypa.io/get-pip.py

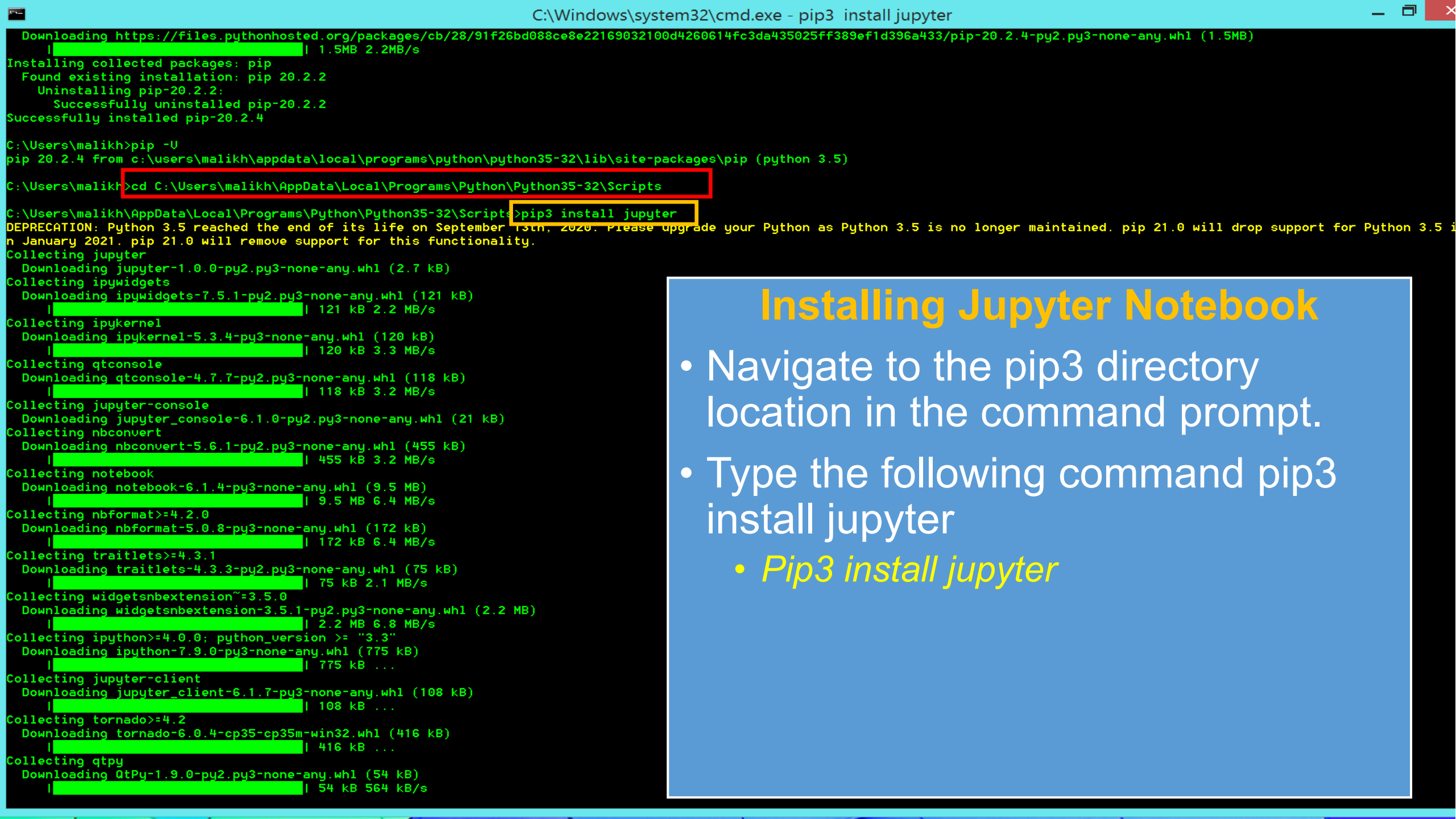


Check If Pip is Installed Correctly

```
  --use-deprecated <feature>  Enable deprecated functionality, that will be removed in the future.

C:\Users\malikh>pip -V
pip 20.2.2 from c:\users\malikh\appdata\local\programs\python\python35-32\lib\site-packages\pip (python 3.5)

C:\Users\malikh>python get-pip.py
Collecting pip
  Downloading https://files.pythonhosted.org/packages/cb/28/91f26bd088ce8e22169032100d4260614fc3da435025ff389ef1d396a433/pip-20.2.4-py2.py3-none-any.whl (1.5MB)
     |                                | 1.5MB 2.2MB/s
Installing collected packages: pip
  Found existing installation: pip 20.2.2
    Uninstalling pip-20.2.2:
      Successfully uninstalled pip-20.2.2
Successfully installed pip-20.2.4

C:\Users\malikh>pip -V
pip 20.2.4 from c:\users\malikh\appdata\local\programs\python\python35-32\lib\site-packages\pip (python 3.5)

C:\Users\malikh>
```

Pip -V

For help with installation issues, refer to : https://pip.pypa.io/en/stable/installing/



Installing Jupyter Notebook

- Navigate to the pip3 directory location in the command prompt.
- Type the following command pip3 install jupyter
  - *Pip3 install jupyter*

```
  Downloading wcwidth-0.2.5-py2.py3-none-any.whl (30 kB)
Collecting MarkupSafe>=0.23
  Downloading MarkupSafe-1.1.1-cp35-cp35m-win32.whl (15 kB)
Collecting packaging
  Downloading packaging-20.4-py2.py3-none-any.whl (37 kB)
Collecting webencodings
  Downloading webencodings-0.5.1-py2.py3-none-any.whl (11 kB)
Collecting pywinpty>=0.5; os_name == "nt"
  Downloading pywinpty-0.5.7-cp35-cp35m-win32.whl (1.3 MB)
     |████████████████████████████████| 1.3 MB 3.3 MB/s
Collecting cffi>=1.0.0
  Downloading cffi-1.14.3-cp35-cp35m-win32.whl (166 kB)
     |████████████████████████████████| 166 kB 6.8 MB/s
Collecting attrs>=17.4.0
  Downloading attrs-20.2.0-py2.py3-none-any.whl (48 kB)
     |████████████████████████████████| 48 kB 1.4 MB/s
Collecting importlib-metadata; python_version < "3.8"
  Downloading importlib_metadata-2.0.0-py2.py3-none-any.whl (31 kB)
Collecting pyrsistent>=0.14.0
  Downloading pyrsistent-0.17.3.tar.gz (106 kB)
     |████████████████████████████████| 106 kB 6.8 MB/s
Collecting parso<0.8.0,>=0.7.0
  Downloading parso-0.7.1-py2.py3-none-any.whl (109 kB)
     |████████████████████████████████| 109 kB 6.8 MB/s
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\malikh\appdata\local\programs\python\python35-32\lib\site-packages (from packaging->bleach->nbconvert->jupyter) (2.2.0)
Collecting pycparser
  Downloading pycparser-2.20-py2.py3-none-any.whl (112 kB)
     |████████████████████████████████| 112 kB 6.4 MB/s
Collecting zipp>=0.5
  Downloading zipp-1.2.0-py2.py3-none-any.whl (4.8 kB)
Building wheels for collected packages: pandocfilters, win-unicode-console, pyrsistent
  Building wheel for pandocfilters (setup.py) ... done
  Created wheel for pandocfilters: filename=pandocfilters-1.4.2-py3-none-any.whl size=7861 sha256=09e386588f70b6ff497deb52f78735d9bbdbf567d6a50dee3c672f7162e2f0c8
  Stored in directory: c:\users\malikh\appdata\local\pip\cache\wheels\f4\ea\bc\f17502354ff54a520dde887e6db7ca29d6587b02197c6d5c24
  Building wheel for win-unicode-console (setup.py) ... done
  Created wheel for win-unicode-console: filename=win_unicode_console-0.5-py3-none-any.whl size=20183 sha256=0c1c0ef41b4d0c278a6f681d898f6bc7d1b05d5e122c3b08c6c0638b8f01419d
  Stored in directory: c:\users\malikh\appdata\local\pip\cache\wheels\20\ec\fa\01c3b911edfed1e869f650767280522b8f742fd12e69e72174
  Building wheel for pyrsistent (setup.py) ... done
  Created wheel for pyrsistent: filename=pyrsistent-0.17.3-cp35-cp35m-win32.whl size=55861 sha256=0399ca0b4877f4c3aa8a6d8c49f1e4fbf5200f16e7a1aae572dce01131f7c79c
  Stored in directory: c:\users\malikh\appdata\local\pip\cache\wheels\6e\ce\d7\a149c2d007203e74294d9d641ea6a6504a3d71a0540790e452
Successfully built pandocfilters win-unicode-console pyrsistent
Installing collected packages: pyzmq, tornado, pywin32, decorator, ipython-genutils, traitlets, jupyter-core, jupyter-client, wcwidth, prompt-toolkit, setuptools, colorama, win-unicode-c
onsole, pickleshare, pygments, parso, jedi, backcall, ipython, ipykernel, attrs, zipp, importlib-metadata, pyrsistent, jsonschema, nbformat, mistune, pandocfilters, MarkupSafe, jinja2, p
ackaging, webencodings, bleach, testpath, entrypoints, defusedxml, nbconvert, pywinpty, terminado, Send2Trash, prometheus-client, pycparser, cffi, argon2-cffi, notebook, widgetsnbextensi
on, ipywidgets, qtpy, qtconsole, jupyter-console, jupyter
  Attempting uninstall: setuptools
    Found existing installation: setuptools 18.2
    Uninstalling setuptools-18.2:
      Successfully uninstalled setuptools-18.2
Successfully installed MarkupSafe-1.1.1 Send2Trash-1.5.0 argon2-cffi-20.1.0 attrs-20.2.0 backcall-0.2.0 bleach-3.2.1 cffi-1.14.3 colorama-0.4.4 decorator-4.4.2 defusedxml-0.6.0 entrypoin
ts-0.3 importlib-metadata-2.0.0 ipykernel-5.3.4 ipython-7.9.0 ipython-genutils-0.2.0 ipywidgets-7.5.1 jedi-0.17.2 jinja2-2.11.2 jsonschema-3.2.0 jupyter-1.0.0 jupyter-client-6.1.7 jupyte
r-console-6.1.0 jupyter-core-4.6.3 mistune-0.8.4 nbconvert-5.6.1 nbformat-5.0.8 notebook-6.1.4 packaging-20.4 pandocfilters-1.4.2 parso-0.7.1 pickleshare-0.7.5 prometheus-client-0.8.0 pr
ompt-toolkit-2.0.10 pycparser-2.20 pygments-2.7.1 pyrsistent-0.17.3 pywin32-228 pywinpty-0.5.7 pyzmq-19.0.2 qtconsole-4.7.7 qtpy-1.9.0 setuptools-50.3.2 terminado-0.8.3 testpath-0.4.4 to
rnado-6.0.4 traitlets-4.3.3 wcwidth-0.2.5 webencodings-0.5.1 widgetsnbextension-3.5.1 win-unicode-console-0.5 zipp-1.2.0

C:\Users\malikh\AppData\Local\Programs\Python\Python35-32\Scripts>
```

```
  Using cached pywin32-228-cp39-cp39-win_amd64.whl (9.1 MB)
Collecting MarkupSafe>=0.23
  Using cached MarkupSafe-1.1.1.tar.gz (19 kB)
Collecting wcwidth
  Using cached wcwidth-0.2.5-py2.py3-none-any.whl (30 kB)
Collecting nest-asyncio
  Downloading nest_asyncio-1.4.2-py3-none-any.whl (5.3 kB)
Collecting async-generator
  Using cached async_generator-1.10-py3-none-any.whl (18 kB)
Collecting packaging
  Using cached packaging-20.4-py2.py3-none-any.whl (37 kB)
Collecting webencodings
  Using cached webencodings-0.5.1-py2.py3-none-any.whl (11 kB)
Collecting parso<0.8.0,>=0.7.0
  Using cached parso-0.7.1-py2.py3-none-any.whl (109 kB)
Collecting pyrsistent>=0.14.0
  Using cached pyrsistent-0.17.3.tar.gz (106 kB)
Collecting attrs>=17.4.0
  Downloading attrs-20.3.0-py2.py3-none-any.whl (49 kB)
     |████████████████████████████████| 49 kB 629 kB/s
Collecting pycparser
  Using cached pycparser-2.20-py2.py3-none-any.whl (112 kB)
Collecting pyparsing>=2.0.2
  Using cached pyparsing-2.4.7-py2.py3-none-any.whl (67 kB)
Using legacy 'setup.py install' for pandocfilters, since package 'wheel' is not installed.
Using legacy 'setup.py install' for pywinpty, since package 'wheel' is not installed.
Using legacy 'setup.py install' for MarkupSafe, since package 'wheel' is not installed.
Using legacy 'setup.py install' for pyrsistent, since package 'wheel' is not installed.
Installing collected packages: decorator, pickleshare, parso, jedi, backcall, pygments, wcwidth, prompt-toolkit, colorama, ipython-genutils, traitlets, ipython, tornado, six, python-date
util, pywin32, jupyter-core, pyzmq, jupyter-client, ipykernel, pycparser, cffi, argon2-cffi, pyrsistent, attrs, jsonschema, nbformat, Send2Trash, pywinpty, terminado, prometheus-client,
MarkupSafe, jinja2, testpath, mistune, entrypoints, pandocfilters, defusedxml, nest-asyncio, async-generator, nbclient, pyparsing, packaging, webencodings, bleach, n
bconvert, notebook, widgetsnbextension, ipywidgets, jupyter-console, qtpy, qtconsole, jupyter
    Running setup.py install for pyrsistent ... done
    Running setup.py install for pywinpty ... done
    Running setup.py install for MarkupSafe ... done
    Running setup.py install for pandocfilters ... done
Successfully installed MarkupSafe-1.1.1 Send2Trash-1.5.0 argon2-cffi-20.1.0 async-generator-1.10 attrs-20.3.0 backcall-0.2.0 bleach-3.2.1 cffi-1.14.3 colorama-0.4.4 decorator-4.4.2 defus
edxml-0.6.0 entrypoints-0.3 ipykernel-5.3.4 ipython-7.19.0 ipython-genutils-0.2.0 ipywidgets-7.5.1 jedi-0.17.2 jinja2-2.11.2 jsonschema-3.2.0 jupyter-1.0.0 jupyter-client-6.1.7 jupyter-c
onsole-6.2.0 jupyter-core-4.6.3 jupyterlab-pygments-0.1.2 mistune-0.8.4 nbclient-0.5.1 nbconvert-6.0.7 nbformat-5.0.8 nest-asyncio-1.4.2 notebook-6.1.5 packaging-20.4 pandocfilters-1.4.3
 parso-0.7.1 pickleshare-0.7.5 prometheus-client-0.8.0 prompt-toolkit-3.0.8 pycparser-2.20 pygments-2.7.2 pyparsing-2.4.7 pyrsistent-0.17.3 python-dateutil-2.8.1 pywin32-228 pywinpty-0.5
.7 pyzmq-19.0.2 qtconsole-4.7.7 qtpy-1.9.0 six-1.15.0 terminado-0.9.1 testpath-0.4.4 tornado-6.1 traitlets-5.0.5 wcwidth-0.2.5 webencodings-0.5.1 widgetsnbextension-3.5.1
WARNING: You are using pip version 20.2.3; however, version 20.2.4 is available.
You should consider upgrading via the 'c:\users\malikh\appdata\local\programs\python\python39\python.exe -m pip install --upgrade pip' command.

C:\Users\malikh\AppData\Local\Programs\Python\Python39>python.exe -m pip install --upgrade pip
Collecting pip
  Using cached pip-20.2.4-py2.py3-none-any.whl (1.5 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 20.2.3
    Uninstalling pip-20.2.3:
      Successfully uninstalled pip-20.2.3
Successfully installed pip-20.2.4

C:\Users\malikh\AppData\Local\Programs\Python\Python39>
```

```
Collecting pyrsistent>=0.14.0
  Using cached pyrsistent-0.17.3.tar.gz (106 kB)
Collecting attrs>=17.4.0
  Downloading attrs-20.3.0-py2.py3-none-any.whl (49 kB)
     |████████████████████████████████| 49 kB 629 kB/s
Collecting pycparser
  Using cached pycparser-2.20-py2.py3-none-any.whl (112 kB)
Collecting pyparsing>=2.0.2
  Using cached pyparsing-2.4.7-py2.py3-none-any.whl (67 kB)
Using legacy 'setup.py install' for pandocfilters, since package 'wheel' is not installed.
Using legacy 'setup.py install' for pywinpty, since package 'wheel' is not installed.
Using legacy 'setup.py install' for MarkupSafe, since package 'wheel' is not installed.
Using legacy 'setup.py install' for pyrsistent, since package 'wheel' is not installed.
Installing collected packages: decorator, pickleshare, parso, jedi, backcall, pygments, wcwidth, prompt-toolkit, colorama, ipython-genutils, traitlets, ipython, tornado, six, python-date
util, pywin32, jupyter-core, pyzmq, jupyter-client, ipykernel, pycparser, cffi, argon2-cffi, pyrsistent, attrs, jsonschema, nbformat, Send2Trash, pywinpty, terminado, prometheus-client,
MarkupSafe, jinja2, testpath, mistune, entrypoints, jupyterlab-pygments, pandocfilters, defusedxml, nest-asyncio, async-generator, nbclient, pyparsing, packaging, webencodings, bleach, n
bconvert, notebook, widgetsnbextension, ipywidgets, jupyter-console, qtpy, qtconsole, jupyter
    Running setup.py install for pyrsistent ... done
    Running setup.py install for pywinpty ... done
    Running setup.py install for MarkupSafe ... done
    Running setup.py install for pandocfilters ... done
Successfully installed MarkupSafe-1.1.1 Send2Trash-1.5.0 argon2-cffi-20.1.0 async-generator-1.10 attrs-20.3.0 backcall-0.2.0 bleach-3.2.1 cffi-1.14.3 colorama-0.4.4 decorator-4.4.2 defus
edxml-0.6.0 entrypoints-0.3 ipykernel-5.3.4 ipython-7.19.0 ipython-genutils-0.2.0 ipywidgets-7.5.1 jedi-0.17.2 jinja2-2.11.2 jsonschema-3.2.0 jupyter-1.0.0 jupyter-client-6.1.7 jupyter-c
onsole-6.2.0 jupyter-core-4.6.3 jupyterlab-pygments-0.1.2 mistune-0.8.4 nbclient-0.5.1 nbconvert-6.0.7 nbformat-5.0.8 nest-asyncio-1.4.2 notebook-6.1.5 packaging-20.4 pandocfilters-1.4.3
 parso-0.7.1 pickleshare-0.7.5 prometheus-client-0.8.0 prompt-toolkit-3.0.8 pycparser-2.20 pygments-2.7.2 pyparsing-2.4.7 pyrsistent-0.17.3 python-dateutil-2.8.1 pywin32-228 pywinpty-0.5
.7 pyzmq-19.0.2 qtconsole-4.7.7 qtpy-1.9.0 six-1.15.0 terminado-0.9.1 testpath-0.4.4 tornado-6.1 traitlets-5.0.5 wcwidth-0.2.5 webencodings-0.5.1 widgetsnbextension-3.5.1
WARNING: You are using pip version 20.2.3; however, version 20.2.4 is available.
You should consider upgrading via the 'c:\users\malikh\appdata\local\programs\python\python39\python.exe -m pip install --upgrade pip' command.

C:\Users\malikh\AppData\Local\Programs\Python\Python39>python.exe -m pip install --upgrade pip
Collecting pip
  Using cached pip-20.2.4-py2.py3-none-any.whl (1.5 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 20.2.3
    Uninstalling pip-20.2.3:
      Successfully uninstalled pip-20.2.3
Successfully installed pip-20.2.4

C:\Users\malikh\AppData\Local\Programs\Python\Python39>jupyter notebook
[I 20:13:37.405 NotebookApp] Writing notebook server cookie secret to C:\Users\malikh\AppData\Roaming\jupyter\runtime\notebook_cookie_secret
[W 20:13:38.295 NotebookApp] Terminals not available (error was No module named 'winpty.cywinpty')
[I 20:13:38.596 NotebookApp] Serving notebooks from local directory: C:\Users\malikh\AppData\Local\Programs\Python\Python39
[I 20:13:38.596 NotebookApp] Jupyter Notebook 6.1.5 is running at:
[I 20:13:38.596 NotebookApp] http://localhost:8888/?token=97c8f935f66155750230889c562dccf4e6644df59b8f0ec6
[I 20:13:38.596 NotebookApp]  or http://127.0.0.1:8888/?token=97c8f935f66155750230889c562dccf4e6644df59b8f0ec6
[I 20:13:38.597 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 20:13:38.695 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/malikh/AppData/Roaming/jupyter/runtime/nbserver-8104-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=97c8f935f66155750230889c562dccf4e6644df59b8f0ec6
     or http://127.0.0.1:8888/?token=97c8f935f66155750230889c562dccf4e6644df59b8f0ec6
```

Running the Notebook — Jupyte X | Home Page - Select or create a n X | +

localhost:8888/tree

Marshall | Home Page - Select... | Social Network for... | 30 Examples to Ma... | Thought Experimen... | Python Read JSON... | Python Dictionary... | Article Processing C... | If I know how to us... | AutoML Tables | G... | Journal of Traffic an... | International Journ... | eTransportation - J... | SN Computer Scien... | Clean Code in Jupyt... | Amazon Found Eve...

jupyter

Quit | Logout

Files | Running | Clusters

Select items to perform actions on them.

Upload | New | Refresh

0 | / | Name | Last Modified | File size

DLLs	20 days ago	
Doc	20 days ago	
etc	29 minutes ago	
include	20 days ago	
Lib	20 days ago	
libs	20 days ago	
Scripts	27 minutes ago	
share	29 minutes ago	
tcl	20 days ago	
Tools	20 days ago	
LICENSE.txt	a month ago	32.6 kB
NEWS.txt	a month ago	1.02 MB
python.exe	a month ago	100 kB
python3.dll	a month ago	58.4 kB
python39.dll	a month ago	4.46 MB

DEMO ;)

Why Jupyter Notebook is Essential in Today's World

Widely Used in Industry & Research
From data science to AI to engineering simulations – used by Google, NASA, Tesla, etc.

Interactive and Documented Coding
Combine code, output, and explanations in one file – ideal for learning and collaboration

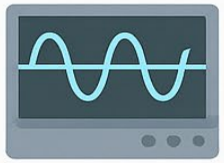
Reproducible Workflows
Integrated with test, tweak, and rerun computations with visible results

Supports Visualizations
Integrated with matplotib, seaborn, Plotly – perfect for analyzing & plotting data

Rich Ecosystem
Works well with Python, R. Julia, machine learning, IoT, etc.

Empowering Engineering Students with Jupyter Notebooks

Simulations & Modeling
Run Python-based simulations for control systems, circuits, thermodynamics, etc.



Data Analysis in Labs
Analyze lab sensor data using Pandas or NumPy in real time

Bridges Theory and Practice
Combine equations, plots, and code to understand physical systems better

Team-Based Projects & Reports
Write readable reports with code + Markdown for capstone or class presentations

Hands-On Coding Without Hassle
No need to compile, just run and learn. Beginner-friendly interface

Why Jupyter Notebook is Essential in Today's World

Widely Used in Industry & Research
From data science to AI to engineering simulations – used by Google, NASA, Tesla, etc.

Interactive and Documented Coding
Combine code, output, and explanations in one file – ideal for learning and collaboration

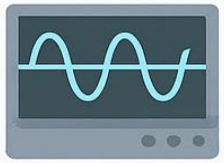
Reproducible Workflows
Integrated with test, tweak, and rerun computations with visible results

Supports Visualizations
Integrated with matplotib, seaborn, Plotly – perfect for analyzing & plotting data

Rich Ecosystem
Works well with Python, R. Julia, machine learning, IoT, etc.

Empowering Engineering Students with Jupyter Notebooks

Simulations & Modeling
Run Python-based simulations for control systems, circuits, thermodynamics, etc.



Data Analysis in Labs
Analyze lab sensor data using Pandas or NumPy in real time

Bridges Theory and Practice
Combine equations, plots, and code to understand physical systems better

Team-Based Projects & Reports
Write readable reports with code + Markdown for capstone or class presentations

Hands-On Coding Without Hassle
No need to compile, just run and learn. Beginner-friendly interface

Why Jupyter Notebook is Essential in Today's World

Widely Used in Industry & Research
From data science to AI to engineering simulations – used by Google, NASA, Tesla, etc.

Interactive and Documented Coding
Combine code, output, and explanations in one file – ideal for learning and collaboration

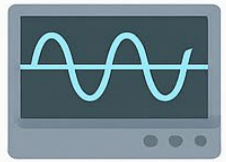
Reproducible Workflows
Integrated with test, tweak, and rerun computations with visible results

Supports Visualizations
Integrated with matplotib, seaborn, Plotly – perfect for analyzing & plotting data

Rich Ecosystem
Works well with Python, R. Julia, machine learning, IoT, etc.

Empowering Engineering Students with Jupyter Notebooks

Simulations & Modeling
Run Python-based simulations for control systems, circuits, thermodynamics, etc.



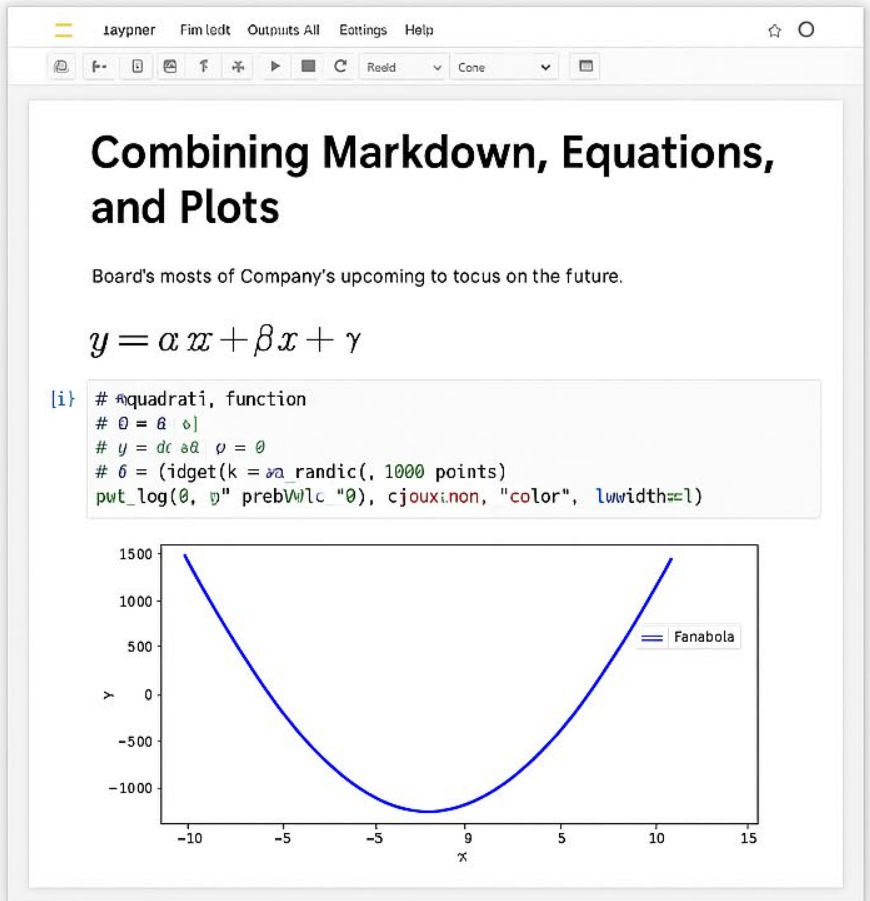
Data Analysis in Labs
Analyze lab sensor data using Pandas or NumPy in real time

Bridges Theory and Practice
Combine equations, plots, and code to understand physical systems better

Team-Based Projects & Reports
Write readable reports with code + Markdown for capstone or class presentations

Hands-On Coding Without Hassle
No need to compile, just run and learn. Beginner-friendly interface

Rich Text and Visualiizations



- Combine code with Markdown, equations, plots, etc.

Why Jupyter Notebook is Essential in Today's World

Widely Used in Industry & Research
From data science to AI to engineering simulations – used by Google, NASA, Tesla, etc.

Interactive and Documented Coding
Combine code, output, and explanations in one file – ideal for learning and collaboration

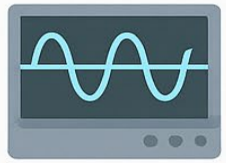
Reproducible Workflows
Integrated with test, tweak, and rerun computations with visible results

Supports Visualizations
Integrated with matplotib, seaborn, Plotly – perfect for analyzing & plotting data

Rich Ecosystem
Works well with Python, R. Julia, machine learning, IoT, etc.

Empowering Engineering Students with Jupyter Notebooks

Simulations & Modeling
Run Python-based simulations for control systems, circuits, thermodynamics, etc.



Data Analysis in Labs
Analyze lab sensor data using Pandas or NumPy in real time

Bridges Theory and Practice
Combine equations, plots, and code to understand physical systems better

Team-Based Projects & Reports
Write readable reports with code + Markdown for capstone or class presentations

Hands-On Coding Without Hassle
No need to compile, just run and learn. Beginner-friendly interface

jupyter Tiled_Linear_Regression_Notebook (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Kernel

Run Markdown

Linear Regression Example

Introduction

This notebook demonstrates simple linear regression using `scikit-learn`.
We will:

- Generate synthetic data
- Train a linear regression model
- Visualize the results

In [ ]:

```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

Generate Sample Data

In [ ]:

```
# Set seed and generate random data
np.random.seed(42)
X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)

# Visualize the data
plt.scatter(X, y)
plt.xlabel("X")
plt.ylabel("y")
plt.title("Generated Data")
plt.grid(True)
plt.show()
```

Train the Linear Regression Model

In [ ]:

```
# Initialize and train the model
model = LinearRegression()
model.fit(X, y)

# Display the learned parameters
print(f"Intercept: {model.intercept_[0]:.2f}")
print(f"Coefficient: {model.coef_[0][0]:.2f}")
```

Visualize the Regression Line

In [ ]:

```
# Predict new values for the line
X_new = np.array([[0], [2]])
y_pred = model.predict(X_new)

# Plot regression line
plt.scatter(X, y, label="Data")
plt.plot(X_new, y_pred, color='red', linewidth=2, label="Regression Line")
plt.xlabel("X")
plt.ylabel("y")
plt.title("Linear Regression Fit")
plt.legend()
plt.grid(True)
plt.show()
```

Conclusion

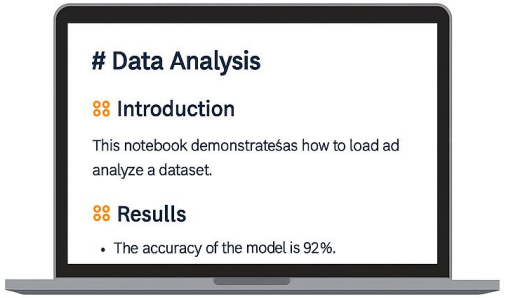
This notebook illustrates how to:

- Use Markdown to explain steps
- Organize cells logically
- Visualize code and output together

A clean notebook helps communicate your analysis clearly.

Make Your Notebook Easy to Follow

- Use Markdown to describe what, why, and how
- Separate sections with headings
- Comment on assumptions and results
- Keep formatting clean and consistent



"Your notebook should explain itself, even without you there."

Cell and Line Magic

What Are Magic Commands?

These are special functions unique to Jupyter Notebooks.

Cell Magic vs. Line Magic

Cell magic commands work on entire cells

Line magic commands work on single lines

Magic Commands in Jupyter Notebooks

Cell Magics

Apply to entire cells
Used by %% followed by the command

```
%%time
run_expensive
 computation()
summarize_results
```

Line Magics

Apply to a single line
Used by % followed by the command

```
%matplotlib inline
```

Popular Line Magics in Jupyter

Command	Purpose / What It Does
%time	Times execution of a single line of code
%timeit	Repeats timing to give more accurate results
run script.py	Runs a Python script from within the notebook
%pwd	Prints the current working directory
%cd	Changes the current directory
%ls	Lists files in the current directory
matplotlib inline	Displays matplotlib plots inside the notebook
%who / %whos	Lists all defined variables in the current namespace
%reset	Clears all variables from the notebook's memory
load script.py	Loads a script's content into a cell

Popular Cell Magic Commands

- **%%time:** Measures the execution time of entire cell
- **%%timeit** Runs a cell multiple times and reports the best average timing
- **%%writefile file.py** Writes contents of the cell to an external file (useful for creating scripts)
- **%%bash** Runs Bash shell commands directly in the cell
- **%%HTML** Renders raw HTML in the output
- **%%latex** Renders LaTeX code (mathematical notation)
- **%%javascript** Runs Javascript code inside the notebook
- **%%capture** Captures output (stdout /stderr) of the cell, useful for suppressing or logging output
- **%%script language** Runs the code in the specified scripting language (e.g. *bash*, perl, ruby)
- **%%debug** Activates the interactive Python debugger after an error

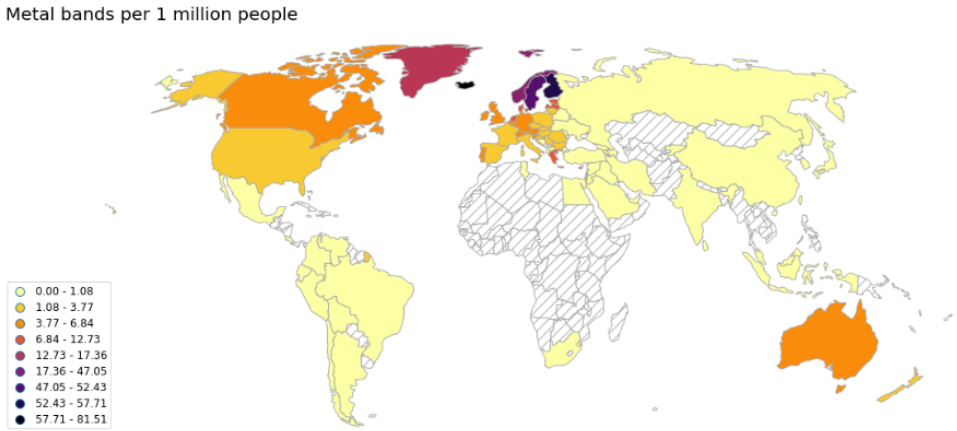
Plot the map

We'll use the handy `plot` method available on `GeoDataFrame` objects. To make sure the map shows all countries, including those without data on metal bands, we have to plot these two sets separately. If you like to learn why, check out this notebook on creating choropleth maps using GeoPandas.

In the final code section, we create two separate data frames `known` and `unknown`. The `known` countries will be plotted using a colormap that seemed appropriate and the Jenks classification method, that reduces the variance within classes and maximize the variance between classes. There will be `9` different classes with darker colors indicating higher band ratios.

The `unknown` countries will be shown with a white background and a striped pattern. We also add some descriptive text, move the legend to the lower left part of the map and set the legend's size.

```
known = world.dropna(subset=['band_ratio'])
unknown = world[world['band_ratio'].isna()]

ax = known.plot(column='band_ratio', cmap='inferno_r', figsize=(20, 12), scheme='fisher_jenks', k=9, legend=True, edgecolor='#aaaaaa')
unknown.plot(ax=ax, color='#ffffff', hatch='//', edgecolor='#aaaaaa')

ax.set_title('Metal bands per 1 million people', fontdict={'fontsize': 20}, loc='left')
description = '''
Based on existing and split-up bands listed on metalstorm.net in 2017 made available in the dataset Metal Bands by Nation kaggle.com/mrpanthers
and population estimates from naturalearthdata.com • Author: Ramiro Gómez - ramiro.org'''.strip()
ax.annotate(description, xy=(0.07, 0.1), size=12, xycoords='figure fraction')

ax.set_axis_off()
legend = ax.get_legend()
legend.set_bbox_to_anchor((.11, .4))
legend.prop.set_size(12)
```



Conclusion

The map above and the one posted on reddit six years ago show similar patterns regarding regions with high and low metal band ratios. Moreover, it is obvious that our dataset comprises less countries and, looking at the actual numbers, has a lot less records in total.

Explainable AI for Software Engineering

A Hands-on Guide on How To Make Software Analytics More Practical, Explainable, and Actionable

About this Book

The success of software engineering projects largely depends on complex decision-making. For example, which tasks should a developer do first, who should perform this task, is the software of high quality, is a software system reliable and resilient enough to deploy, etc. However, erroneous decision-making for these complex questions is costly in terms of money and reputation. Thus, Artificial Intelligence/Machine Learning (AI/ML) techniques have been widely used in software engineering for developing software analytics tools and techniques to improve decision-making, developer productivity, and software quality. However, the predictions of such AI/ML models for software engineering are still not practical (i.e., fine-grained), not explainable, and not actionable. These concerns often hinder the adoption of AI/ML models in software engineering practices. In addition, many recent studies still focus on improving the accuracy, while a few of them focus on improving explainability. **Are we moving in the right direction? How can we better improve the SE community (both research and education)?** In this book, we first provide a concise yet essential introduction to the most important aspects of Explainable AI and a hands-on tutorial of Explainable AI tools and techniques. Then, we introduce the fundamental knowledge of defect prediction (an example application of AI for Software Engineering). Finally, we demonstrate three successful case studies on how Explainable AI techniques can be used to address the aforementioned challenges by making the predictions of software defect prediction models more practical, explainable, and actionable.

Search this book...

Explainable AI for Software Engineering

HANDS-ON EXERCISE

ASE2021 PyExplainer Live-Demo

ASE2021 Hands-on Exercise

PART1-WHAT IS EXPLAINABLE AI?

A Theory of Explanations

Techniques for Generating Explanations

Model-specific Techniques for Generating Global Explanations

Model-agnostic Techniques for Generating Local Explanations

PART2-DEFECT PREDICTION MODELS

Software Quality Assurance

Defect Prediction

(Step 1) Data Collection

(Step 2) Data Preprocessing

(Step 3) Model Construction

(Step 4) Model Evaluation

(Step 5) Model Ranking

PART3-EXPLAINABLE AI FOR SOFTWARE ENGINEERING

ASE2021 PyExplainer Live-Demo

Explaining instances with PyExplainer is simple. First, we need to load necesarry libraries as well as preparing datasets.

```
## Load Data and preparing datasets

# Import for Load Data
from os import listdir
from os.path import isfile, join
import pandas as pd

# Import for Split Data into Training and Testing Samples
from sklearn.model_selection import train_test_split

train_dataset = pd.read_csv(("../../datasets/lucene-2.9.0.csv"), index_col = 'File')
test_dataset = pd.read_csv(("../../datasets/lucene-3.0.0.csv"), index_col = 'File')

outcome = 'RealBug'
features = ['OWN_COMMIT', 'CountClassCoupled', 'AvgLine', 'RatioCommentToCode']
# OWN_COMMIT - # code ownership
# Added lines - # of added lines of code
# Count class coupled - # of classes that interact or couple with the class of interest
# RatioCommentToCode - The ratio of lines of comments to lines of code

# process outcome to 0 and 1
train_dataset[outcome] = pd.Categorical(train_dataset[outcome])
train_dataset[outcome] = train_dataset[outcome].cat.codes

test_dataset[outcome] = pd.Categorical(test_dataset[outcome])
test_dataset[outcome] = test_dataset[outcome].cat.codes

X_train = train_dataset.loc[:, features]
X_test = test_dataset.loc[:, features]

y_train = train_dataset.loc[:, outcome]
y_test = test_dataset.loc[:, outcome]


class_labels = ['Clean', 'Defective']

X_train.columns = features
X_test.columns = features
training_data = pd.concat([X_train, y_train], axis=1)
testing_data = pd.concat([X_test, y_test], axis=1)
```

Then, we construct a Random Forests model as a predictive model to be explained.

(1) Please construct a Random Forests model using the code cell below.

Tips

Contents

PyExplainer

Software Quality Assurance

Our society is now driven by software. But, one software defect could have an ultimate impact on our society, financial economy, and human lives, ranging from IT services disruption, massive overdose of radiotherapy of Therac-25, to an explosion of the Ariane 5 rocket. These failures are often due to the discovery of the majority of software defects after release, while they are often introduced in the early development life cycle like the requirements, design, release planning. Since the cost of fixing software defects is exponentially increased [BJ95], finding and fixing software defects prior to releasing software is usually much cheaper and faster.

Software Quality Assurance (SQA) is neither a process nor a phase but a must-need systematic practice to ensure that the product meets the quality standards, especially for the development of any life-impacting and safety-critical software systems. Thus, QA practices must be embedded as a quality culture throughout the life cycles from planning to release so teams can follow best practices to prevent defects, rather than wasting time detecting them.

However, modern software development practices like Agile pose several challenges to the current SQA practices. In modern practices, everyone is accountable and quality is everyone's responsibility. Thus, developers are encouraged to test their own code and to write sufficient and effective unit tests to ensure the new code has no obvious errors and to get notified quickly as soon as something is broken. Several defect detection tools are used during the software development life cycle. For example, automation testing (e.g., CI/CD) can be used to enable the automation of test execution where the code coverage is used to infer a certain level of quality in the meaningful test cases. Static analysis is used to detect bugs, but produce too many false positives or false alarms. Code review can be used to review all code changes prior to integrating a pull request into the main branch repository.

Despite these SQA tools being invested and adopted, software defects might still occur since these tools do not guarantee a defect-free software product. They also neither identify the problematic areas that may lead to software defects after release (i.e., post-release software defects) nor provide any actionable guidance how to mitigate risks. As a result, it could lead to ineffective and inefficient software quality assurance practices. Teams may release poor quality of software products to customers, causing slow project progress and high costs of software development, unsatisfactory software products, unhappy end-users, and serious monetary and reputation losses for that software organization.

Figure 2 illustrates a simplified software engineering workflow that includes SQA activities.

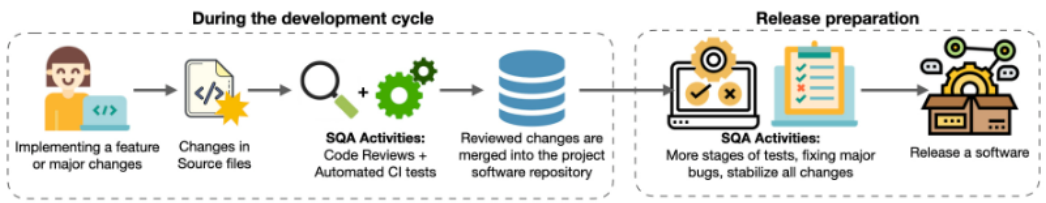


Fig. 2 An overview of the software quality assurance activities.

Precision

Precision measures the proportion between the number of lines that are correctly identified as defective and the number of lines that are identified by the models. Particularly, precision can be computed using a calculation of $\frac{TP}{(TP+FP)}$, where TP is the number of actual defective lines that are predicted as defective and FP is the number of clean lines that are predicted as defective. A high precision value indicates that the models can correctly identify high number of defective lines.

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score

# construct a confusion matrix
print('Construct a confusion matrix:')
tn, fp, fn, tp = confusion_matrix(y_test,rf_model.predict(X_test)).ravel()
print('(True Positive, False Positive) = (' + str(tp) +','+str(fp)+')')
print('(False Negative, True Negative) = (' + str(fn) +','+str(tn)+')\n')

# calculate precision manually
rf_precision_manual = tp/(tp + fp)

# calculate precision with a function
rf_precision_function = precision_score(y_test, rf_model.predict(X_test))

print('Precision (manual calculation)\t\t:', rf_precision_manual)
print('Precision (precision_score function)\t:', rf_precision_function)
```

```
Construct a confusion matrix:
(True Positive, False Positive) = (99,147)
(False Negative, True Negative) = (56,1035)

Precision (manual calculation)          : 0.4024390243902439
Precision (precision_score function)    : 0.4024390243902439
```

Recall

Recall measures the proportion between the number of lines that are correctly identified as defective and the number of actual defective lines. More specifically, we compute recall using a calculation of $\frac{TP}{(TP+FN)}$, where TP is the number of actual defective lines that are predicted as defective and FN is the number of actual defective lines that are predicted as clean. A high recall value indicates that the approach can identify more defective lines.

```
from sklearn.metrics import recall_score

# calculate recall manually
rf_recall_manual = tp/(tp + fn)

# calculate recall with a function
rf_recall_function = recall_score(y_test, rf_model.predict(X_test))
```

Health Care Spending

localhost:8888/notebooks/Documents/Dyalog/Jupyter%20Notebooks...

File Edit View Insert Cell Kernel Widgets Help Trusted Dyalog APL

Cleaning the Data

Apparently there's no GDP or health expenditure data for Antarctica... Our next step is to filter out any records that have "null" entries anywhere, remove countries with per capita GDP < 25,000 - each time displaying the resulting item count:

In [22]:
```
≠tab←(~∨/tab∈⊂'null')⌿tab      ⍝ remove countries with null data
≠tab←(tab[;3]≥25000)⌿tab       ⍝ remove counties with per capita GDP<25000
```

Out[22]: 191

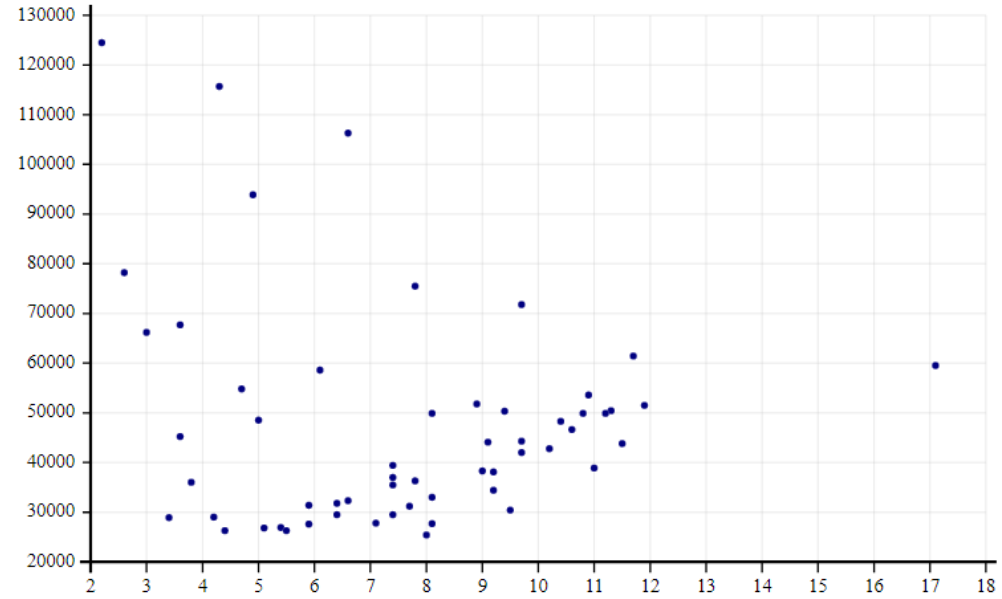
Out[22]: 57

Charting

Finally, we create our scatterplot showing Health Care Expenditure as a percentage of GDP versus per capita GDP:

In [23]:
```
InitCauseway ⍬
sp←⎕NEW Causeway.SharpPlot
sp.SetMarkers ⊂,Causeway.Marker.Bullet
sp.ScatterPlotStyle←Causeway.ScatterPlotStyles.ValueTags
sp.ValueTagStyle←Causeway.ValueTagStyles.Tips
sp.SetValueTags ⊂tab[;1]
sp.XAxisStyle←Causeway.XAxisStyles.GridLines
sp.YAxisStyle←Causeway.YAxisStyles.GridLines
sp.DrawScatterPlot (,⊂tab[;3])(tab[;2])
{⍬}3500⌶sp.RenderSvg Causeway.SvgMode.FixedAspect
```

Out[23]:



Example notebook

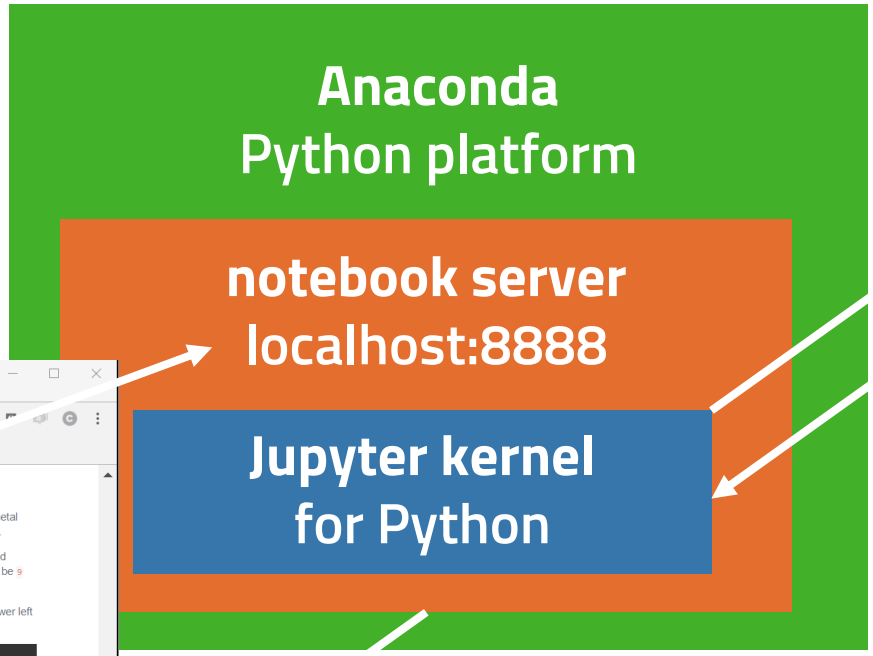
DEMO ;)

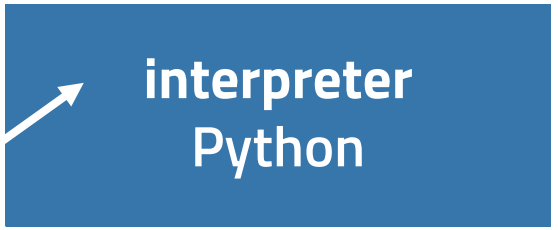
Notebook Benefits

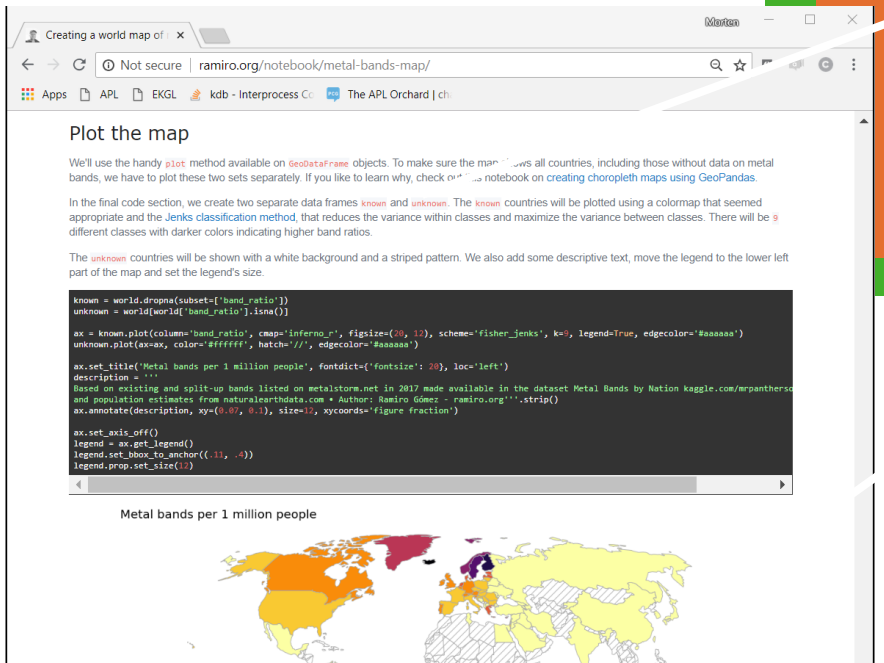
- A single document that combines explanations with executable code and its output — an ideal way to provide:
  - **R**eproducible research results
  - **D**ocumentation of processes
  - **I**nstructions
  - **T**utorials and training materials
  - **A** digital learning environment for computational thinking

What is Jupyter notebook?

Local notebook server — Python





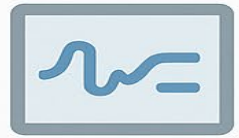


Note, Anaconda Platform is optional. I will demonstrate you how to install and use Jupyter without the Anaconda

Reasons to Use Jupyter Notebooks

Interactive Coding Environment

Write and execute code in an interactive, cell-based framework



Rich Text and Visualizations

Combine code with Markdown, equations, plots, etc.

Browser-Based and Accessible

Accessible from any device with a web browser

Popular in Industry and Academia

Widely used for data analysis, scientific computing, and education

Popular Cell Magic Commands

- **%%time** Time the execution of a cell
- **%writefile** Write cell contents to a file
- **%capture** Capture cell output
- **%bash** Run cell as a Bash script
- **%html** Render HTML in a cell
- **%Latex** Render LaTeX content in a cell
- **%debug** Activate debugger
- **%cython** Compile Cython code

Understanding Line and Cell Magics in Jupyter

What Are Magics?

- Special commands p/= pr≤ fixed with % or 1%
- Enhance functionality, allowing tu timing code, changing directories, running OS commands, etc.

Line Magics (%)

- Apply to a singec ⌣code cele cell

```
%timeit x = sun(range(1000))
%pwd
%ismagic
```

Popular Magic Commands to Teach

Magic	Purpose
%time	Time execution of 1line
%%timelt	Time execution on cell
%run	Run a Pythen script
%load	Load a script into cell
%%hash	Run shell commands

Cell Magics (%)

- Apply to an entire code code cell

```
%stimeit
  total = 0
  for i = Hello from bash'
```

- % bash

```
%sbash "Hiello from bash"
```

- %writefile my_script.py

```
print("This will be saved in a .py file")
```

Best Practices

- Use magics for exploration, testing, and system commands
- Avoid using them in production code (not portable to .py sccripts)
- Combine with Markdown freat teaching demonstrations