# Issues Tracking

Presented by Berk Bekiroglu

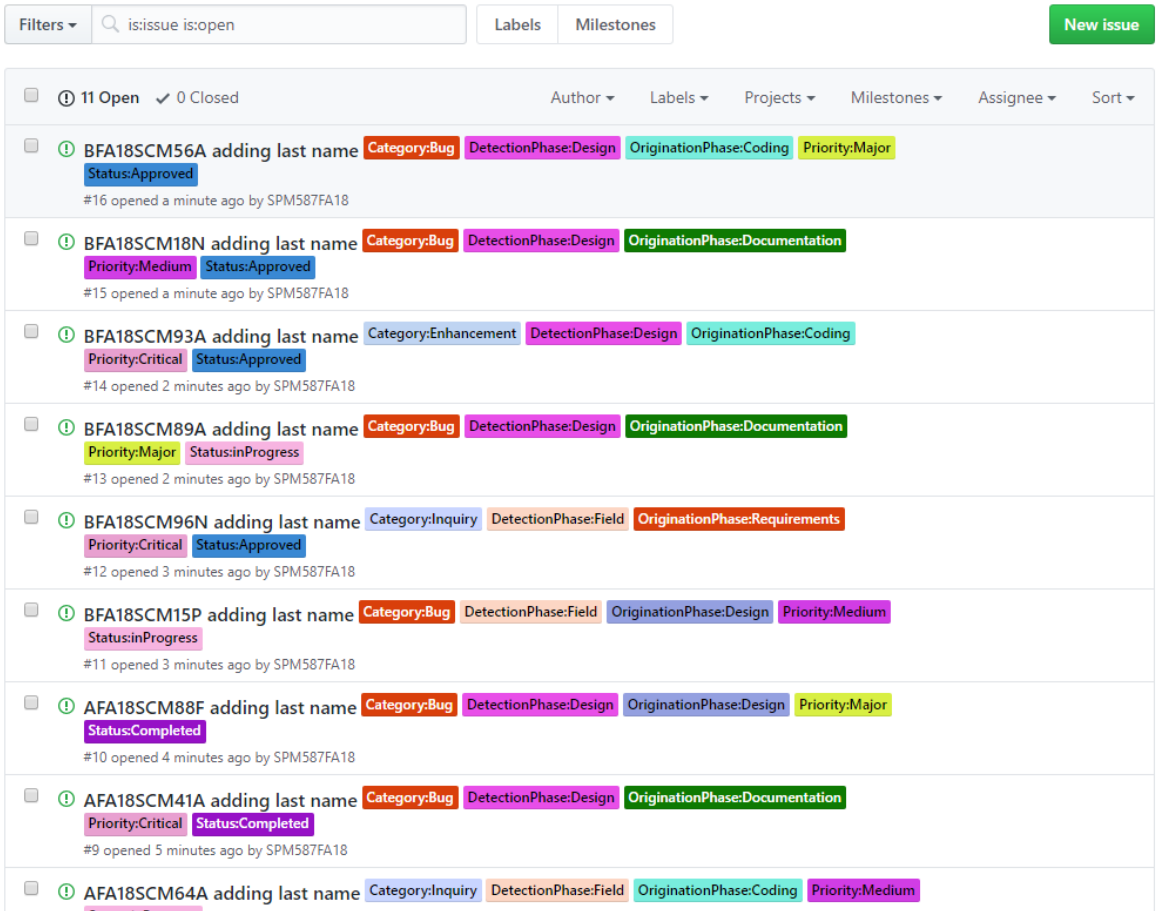
bbekirog@hawk.iit.edu

# An Overview

- We have seen how GitHub and Git commands works.
- This tutorial gives you an introduction to the issues tracking that you will be working with.
- Issues tracking is the process of getting all information about issues like issue created by, issue created at, issue label, issue number, issue closed at, issue state etc.
- How we will get these all information of issues explained in this tutorial will guide you through your assignments.

# How it works?

- In the following screenshot you can see issues.
- Now, we want detailed information about these issues.
- We will use python scripts to get detailed information about issues.



# Why is it needed?

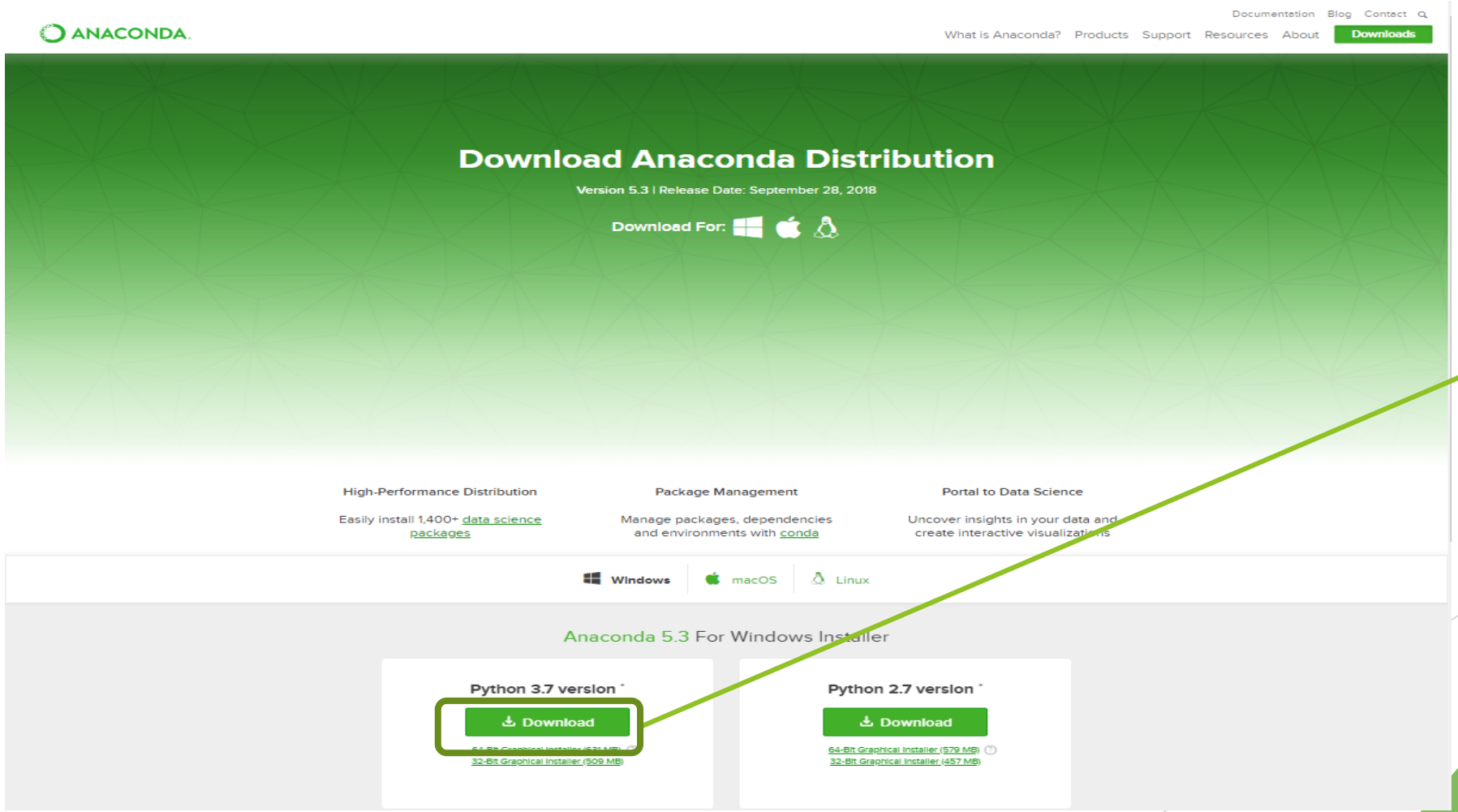
- Many engineers will be working on a single project.
- Number of issues are created/closed every day on GitHub.
- It is tedious work to know how many issues are created/closed on particular date.
- We can create python script for that and get all these information easily.

# Python – Overview:

- Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable.
- It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactical constructions than other languages.
- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

# ANACONDA

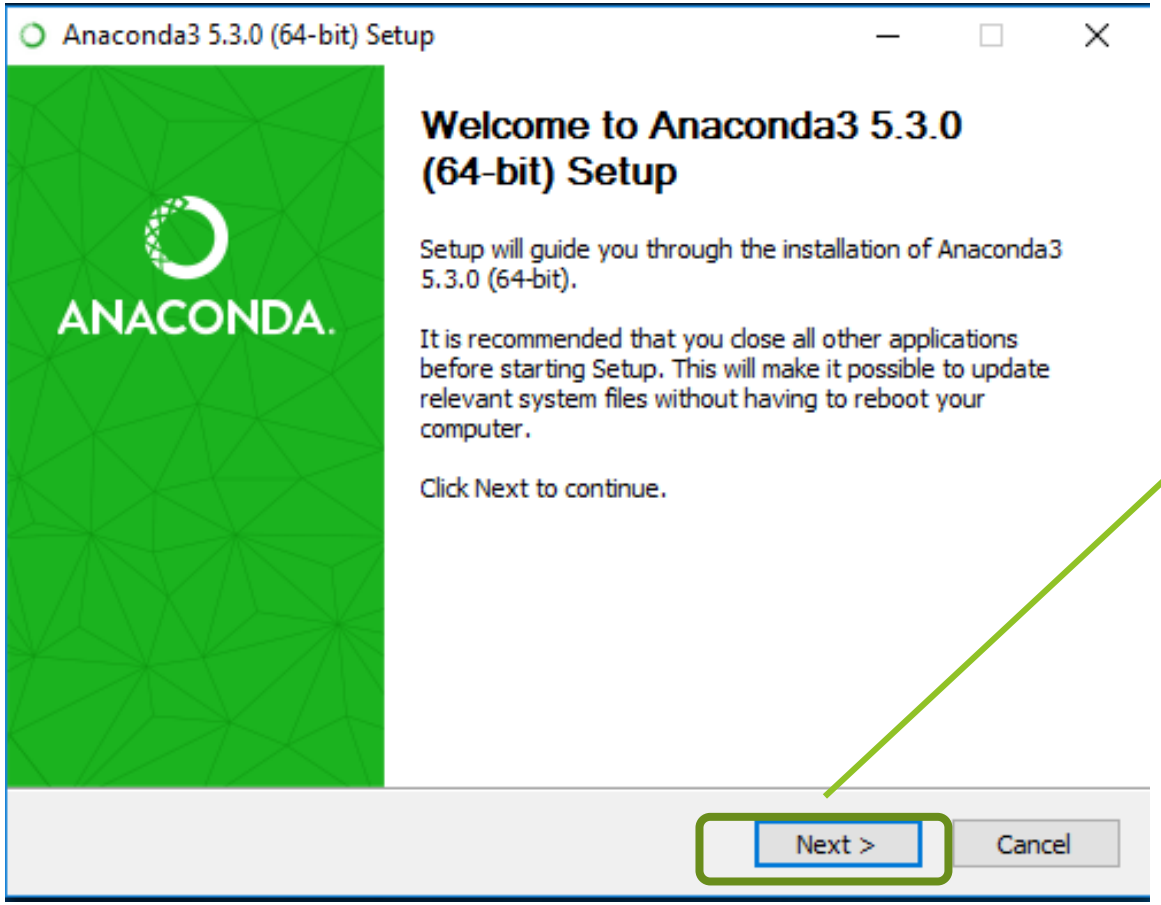
- https://www.anaconda.com/download/




Click on Download

# ANACONDA – Installation:

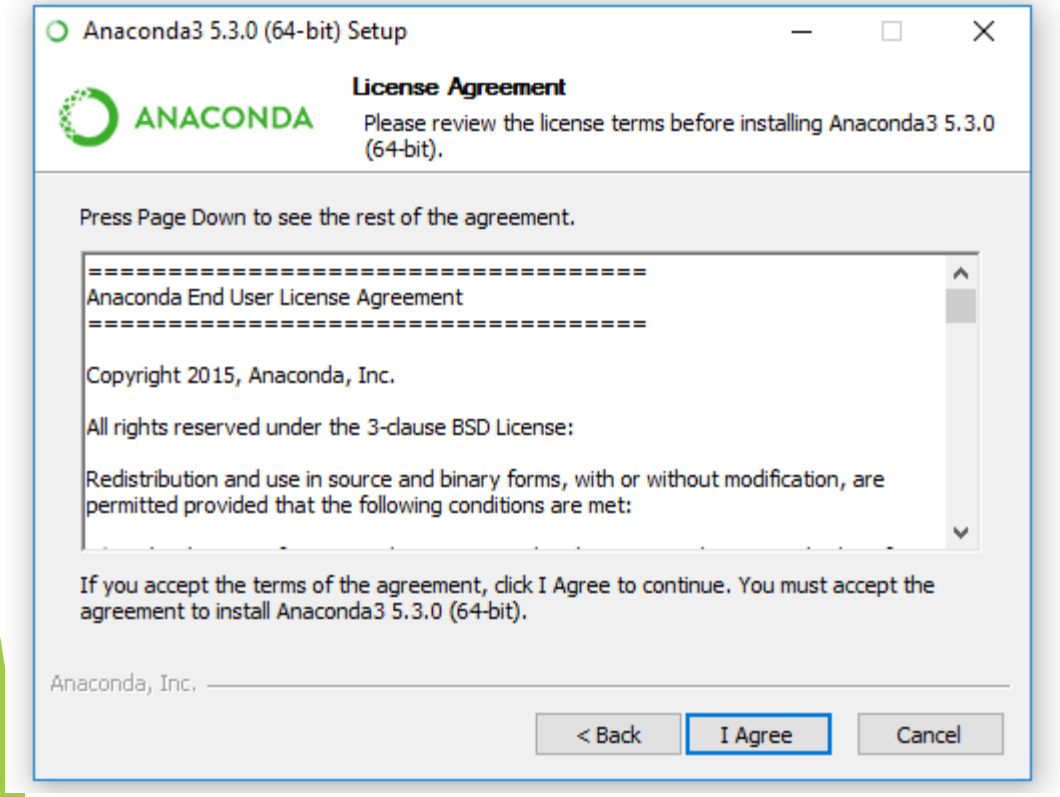
- Run Anaconda3-5.3.0-Windows-x86_64.exe
- Click Next to continue

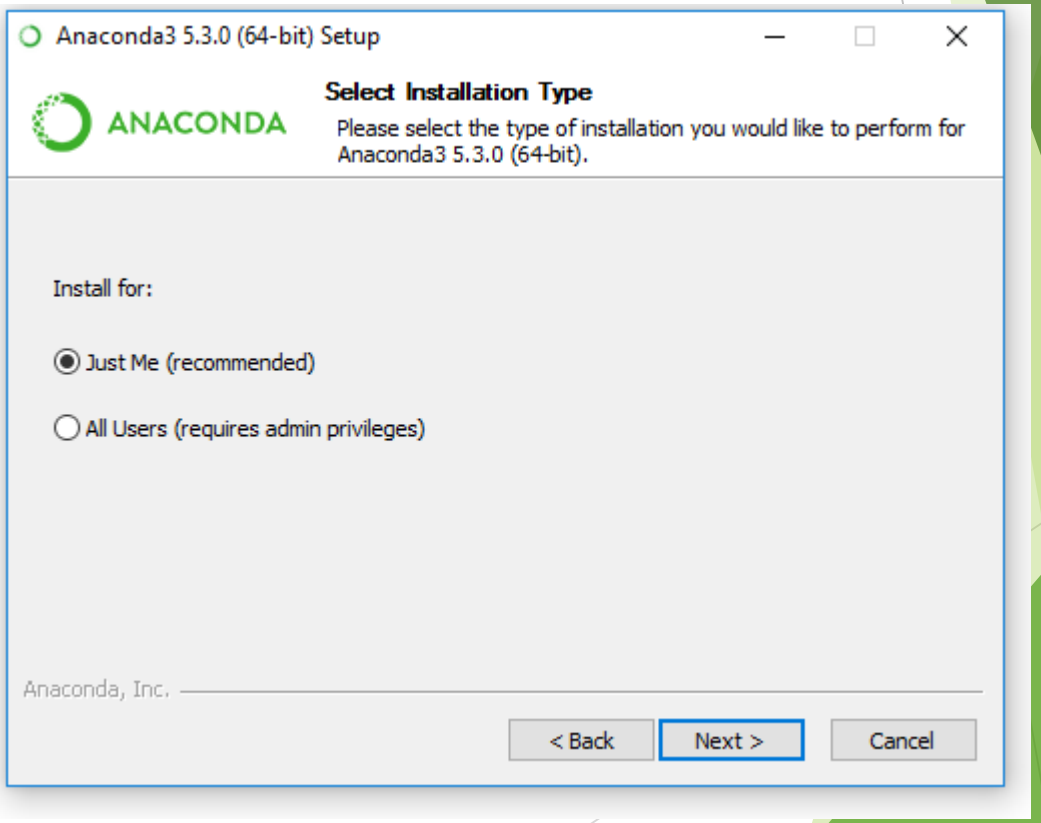




Click Next

# ANACONDA – Installation:

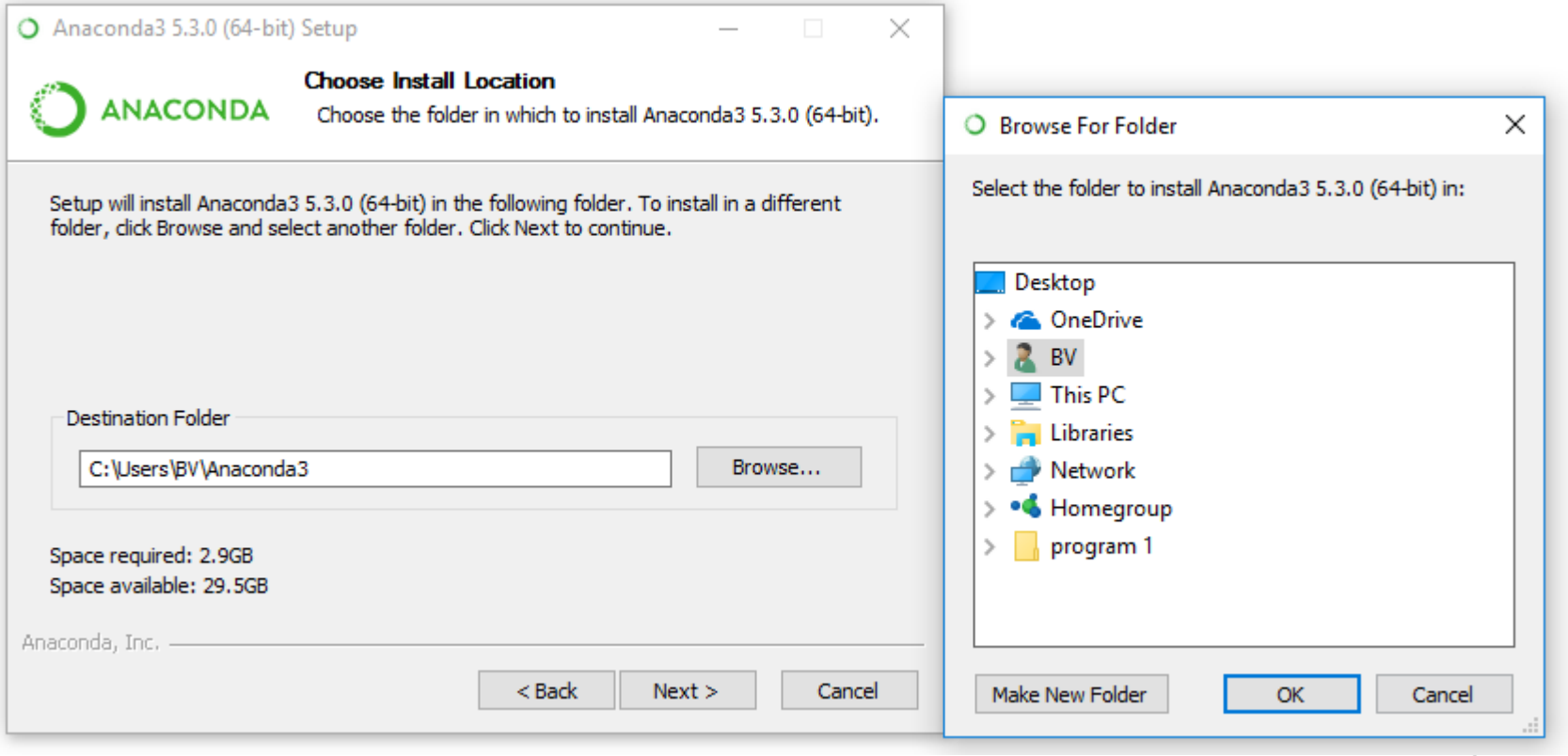
- Click I agree the License Agreement
- Then, select Installation Type



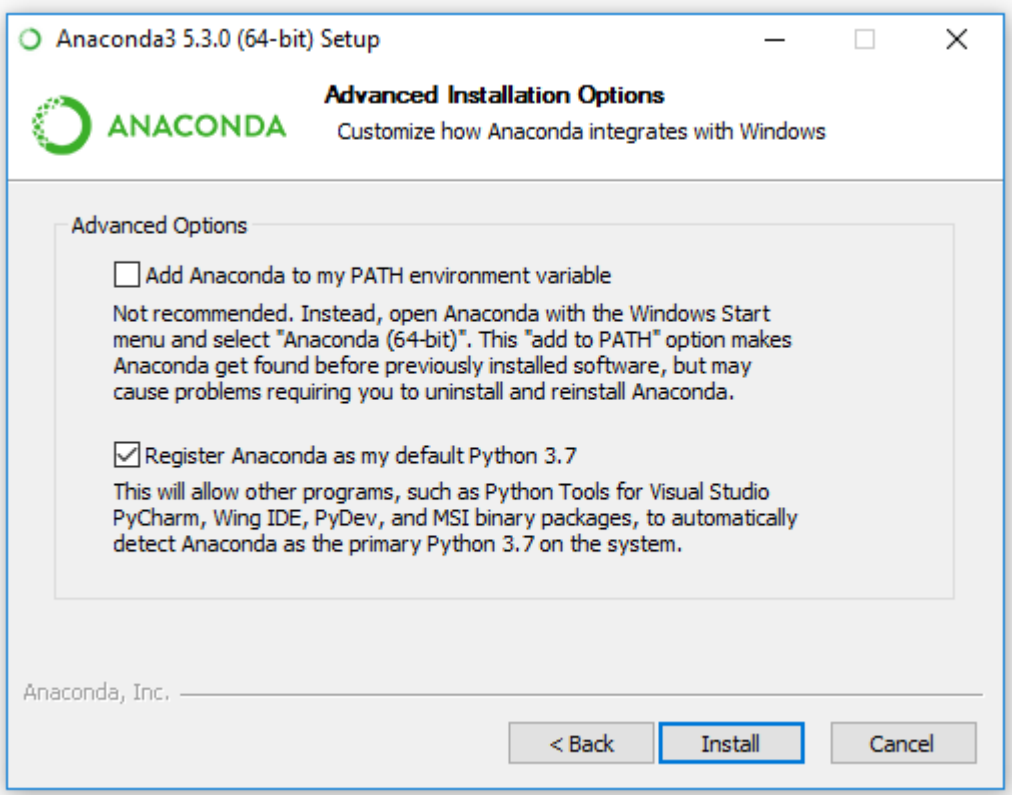
# ANACONDA – Installation:

- Choose Install Location
- Click Browse to select installation directory
- Click Next to continue



# ANACONDA – Installation:

- Advanced Installation Options
- Click Install



# Run Jupyter notebook

- You can start the notebook server from the command line
- Run Anaconda Prompt or Command Prompt on Windows
- Type command “jupyter notebook” to start notebook server

jupyter notebook

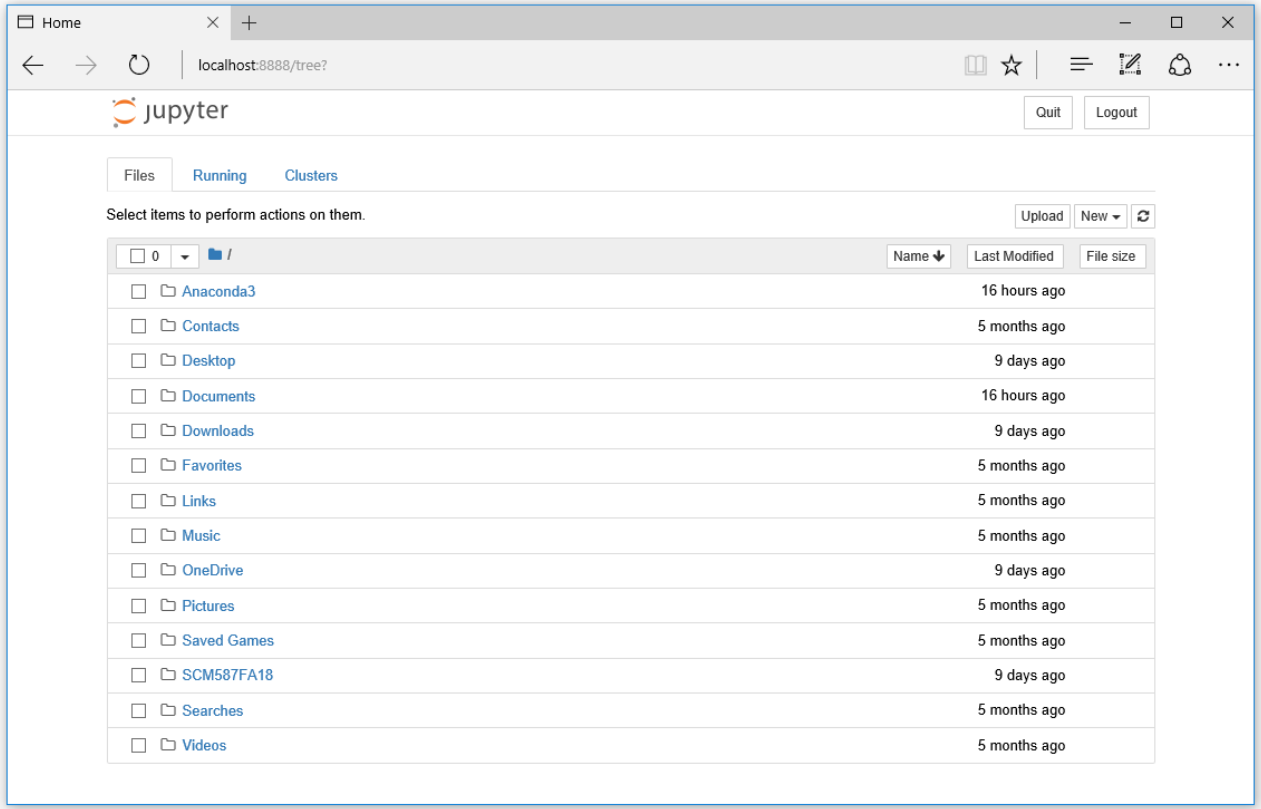
```
(base) C:\Users\BV>jupyter notebook
[I 14:48:35.662 NotebookApp] Writing notebook server cookie secret to C:\Users\BV\AppData\Roaming\jupyter\runtime\notebo
ok_cookie_secret
[I 14:48:38.427 NotebookApp] JupyterLab extension loaded from C:\Users\BV\Anaconda3\lib\site-packages\jupyterlab
[I 14:48:38.436 NotebookApp] JupyterLab application directory is C:\Users\BV\Anaconda3\share\jupyter\lab
[I 14:48:38.503 NotebookApp] Serving notebooks from local directory: C:\Users\BV
[I 14:48:38.508 NotebookApp] The Jupyter Notebook is running at:
[I 14:48:38.514 NotebookApp] http://localhost:8888/?token=59cc742480b8a08b58628f931774c39f7947f2e8937db434
[I 14:48:38.523 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 14:48:38.601 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8888/?token=59cc742480b8a08b58628f931774c39f7947f2e8937db434
[I 14:48:39.555 NotebookApp] Accepting one-time-token-authenticated connection from ::1
```

Use this URL if a browser needs a token to login

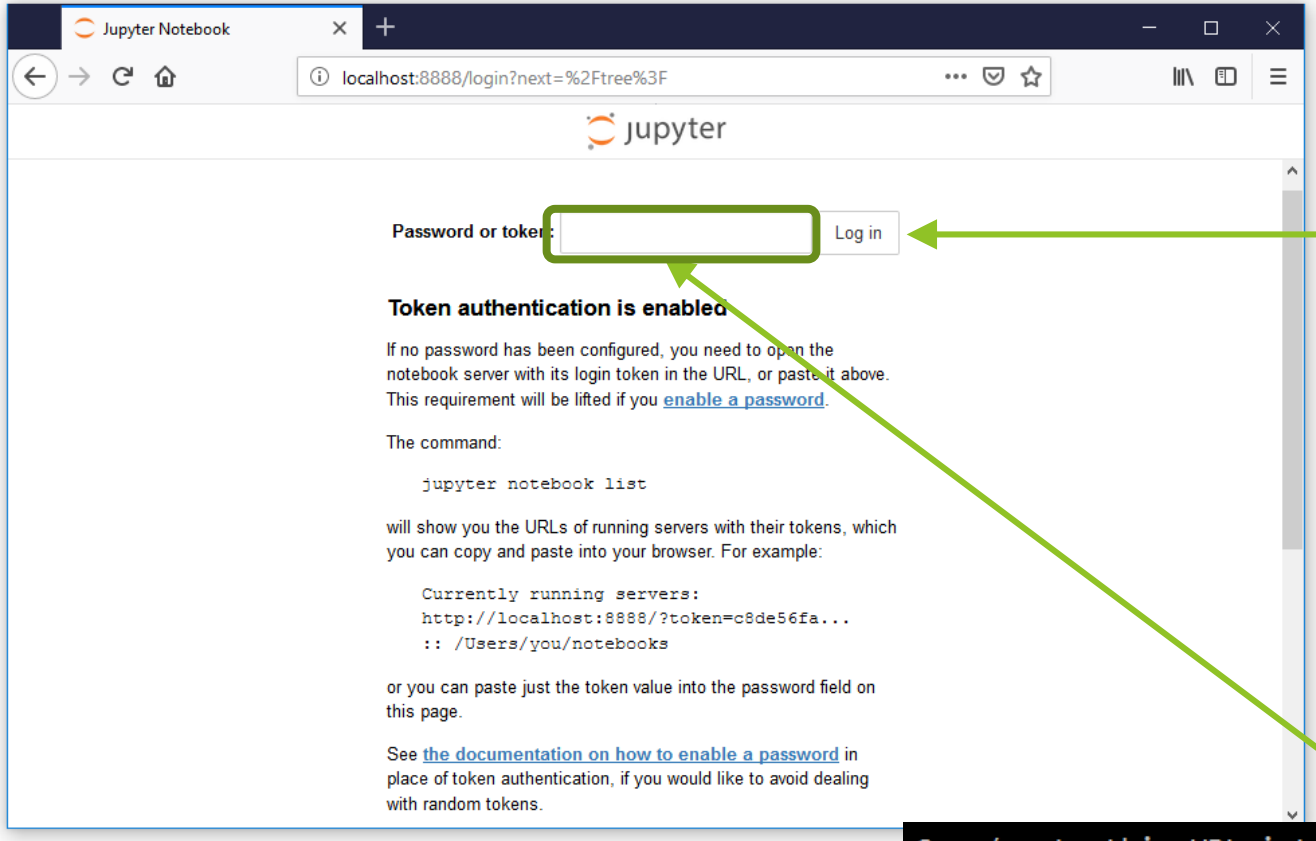
# Run Jupyter notebook

- Notebook server runs on http://localhost:8888
- Your default browser opens Jupyter notebook automatically after the "jupyter notebook" command
- You can copy and paste http://localhost:8888 into your browser if the browser does not open after the command



# Run Jupyter notebook on another browser

- To run Jupyter notebook on a browser rather than your default browser, you may need to provide a password or token to login
- The token was generated after “jupyter notebook” command



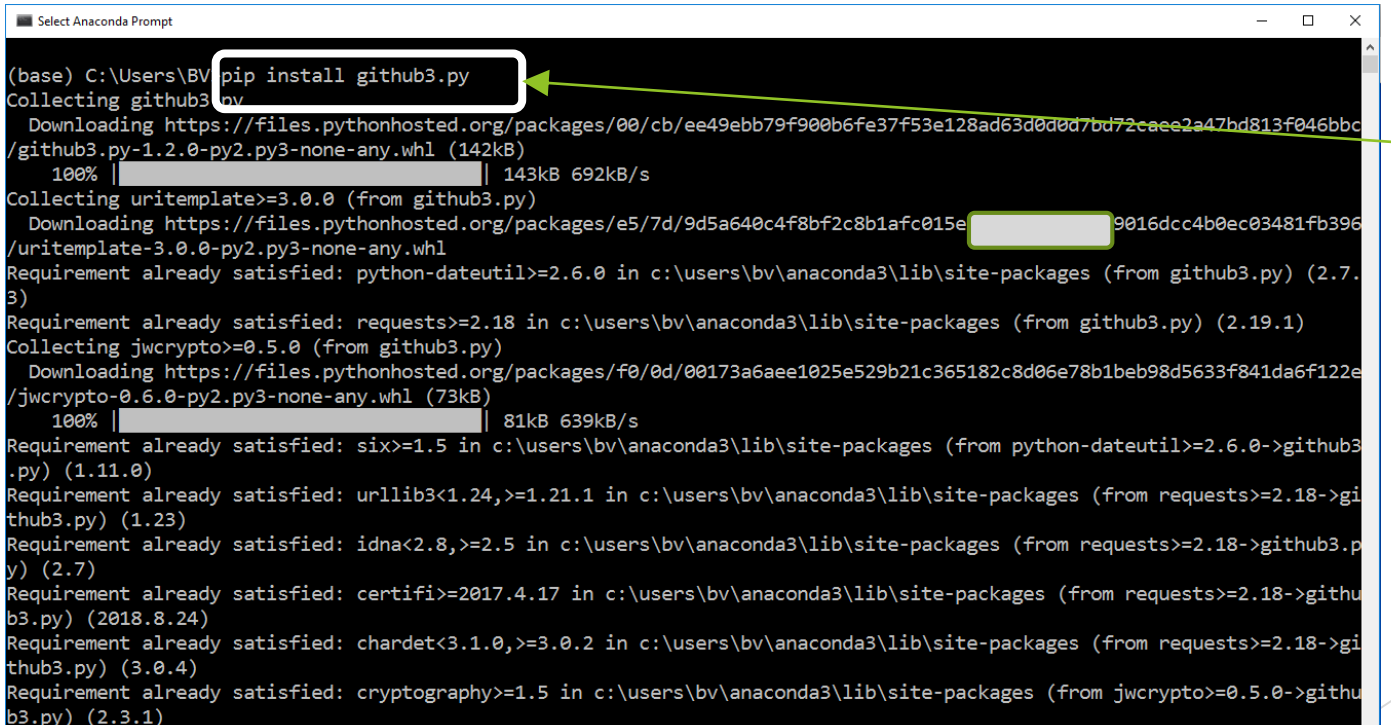

2. Click Log in

1. Enter token to login

```
Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
    http://localhost:8888/?token=59cc742480b8a08b58628f931774c39f7947f2e8937db434
```

# github3 API Download

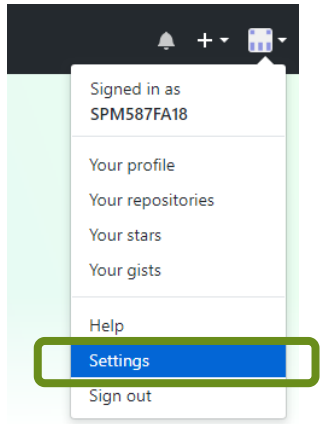
- github3 API is a library that allows python code to be connected with GitHub and get issues.
- pip is a package management system used to install and manage software packages written in Python
- Install github3 API using the command pip install github3.py in your Anaconda command prompt

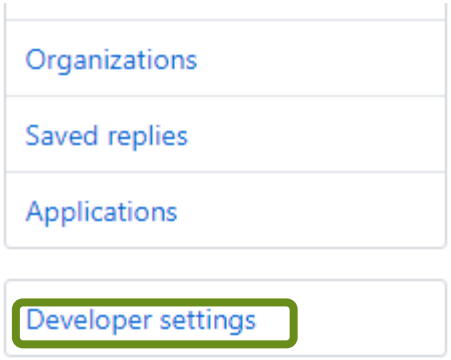



Download Github3 API using pip

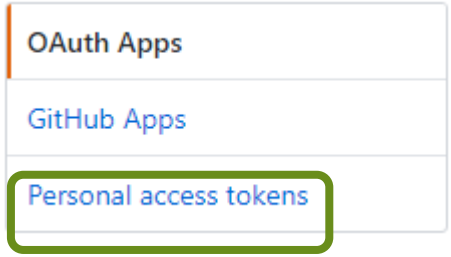
# Generating Access tokens for Github

- You can see steps to generate Github API Token in following link. https://help.github.com/articles/creating-an-access-token-for-command-line-use/
- In the upper-right corner of any page, click your profile photo, then click **Settings**. Then, in the personal settings sidebar, click **Developer settings**



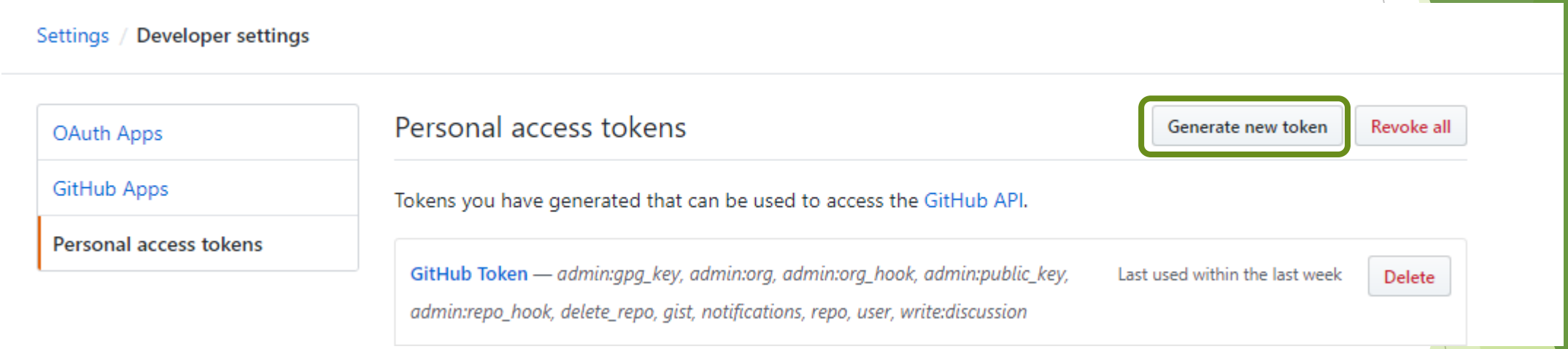




- In the Developer settings sidebar, click **Personal access tokens.**

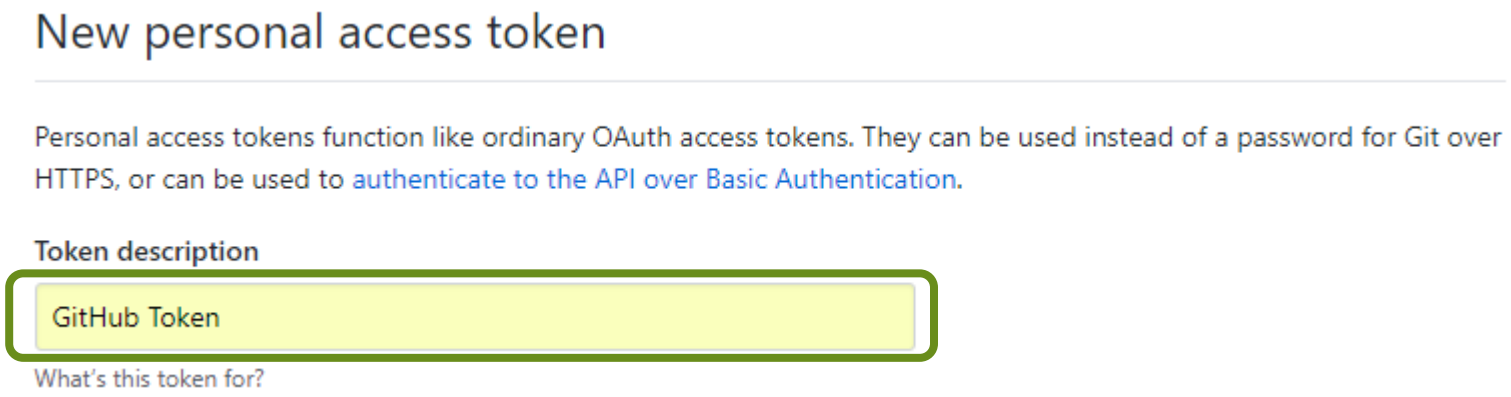


# Generating Access tokens for Github

- Click **Generate new token.**



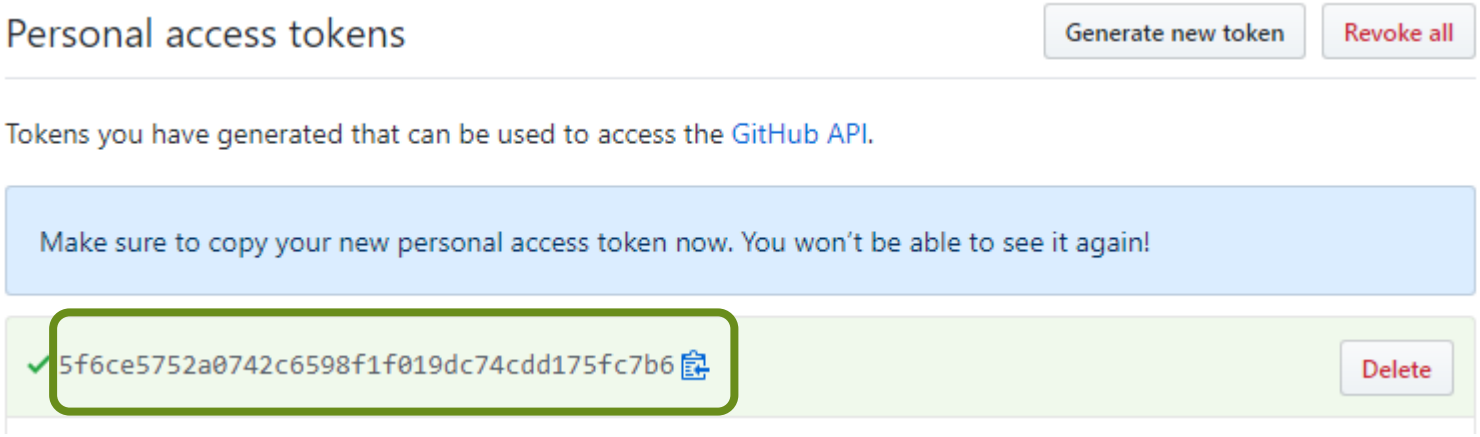

- Give your token a descriptive name.



# Generating Access tokens for Github

- Click **Generate token.**


- Copy the token to your clipboard. For security reasons, after you navigate off this page, no one will be able to see the token again.



# Open a specific notebook (ipynb script)

- To open Import_Issues.ipynb on Jupyter notebook
- Go to the directory that a notebook file is located
- Run command “jupyter notebook *fileName*.ipynb”

```
(base) C:\Users\BV\Desktop\CS587>jupyter notebook Import_Issues.ipynb
[I 15:41:16.985 NotebookApp] JupyterLab extension loaded from C:\Users\BV\Anaconda3\lib\site-packages\jupyterlab
[I 15:41:16.994 NotebookApp] JupyterLab application directory is C:\Users\BV\Anaconda3\share\jupyter\lab
[I 15:41:17.006 NotebookApp] Serving notebooks from local directory: C:\Users\BV\Desktop\CS587
[I 15:41:17.013 NotebookApp] The Jupyter Notebook is running at:
[I 15:41:17.017 NotebookApp] http://localhost:8888/?token=bfb1a7c32f6bb133f6568ed5faa732d97c1ae3848788f2b4
[I 15:41:17.025 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 15:41:17.091 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8888/?token=bfb1a7c32f6bb133f6568ed5faa732d97c1ae3848788f2b4
[I 15:41:23.473 NotebookApp] Accepting one-time-token-authenticated connection from ::1
[I 15:41:32.090 NotebookApp] Writing notebook-signing key to C:\Users\BV\AppData\Roaming\jupyter\notebook_secret
[W 15:41:32.114 NotebookApp] Notebook Import_Issues.ipynb is not trusted
[I 15:41:33.132 NotebookApp] Kernel started: 06a38ce6-b9cf-43b3-9dd7-e7a39815101f
[I 15:41:35.664 NotebookApp] Adapting to protocol v5.1 for kernel 06a38ce6-b9cf-43b3-9dd7-e7a39815101f
```

Jupyter notebook file name

# Jupyter Notebook:

- Jupyter notebook is a platform a server-client application that allows editing and running python notebook documents via a web browser.

Import_Issues

localhost:8888/notebooks/Import_Issues.ipynb

jupyter Import_Issues Last Checkpoint: 8 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

Run Code

In [4]:
```
# pip install github3.py command for install github3
# execute pip install github3.py command from the the command window/terminal

import github3, json
```

In [5]:
```
# Get an API key for GitHub and set it as GITHUB_TOKEN
# Here is the URL to generate your GITHUB_TOKEN
# https://help.github.com/articles/creating-an-access-token-for-command-line-use/

GITHUB_TOKEN = 'INSERT_YOUR_SECRET_GITHUB_API_KEY_HERE'
ORG = 'SPM587FA18'
REPO = 'SCM587FA18'
FILENAME_ISSUES = ORG + 'issues.json'
```

In [6]:
```
gh = github3.login(token=GITHUB_TOKEN)

f = open(FILENAME_ISSUES, 'w')
for issue in gh.search_issues('type:issue repo:SPM587FA18/SCM587FA18'):          # Find issues from given Repo
            label_name=[]
            data={}
            current_issue = issue.as_json()
            current_issue = json.loads(current_issue)
            data['issue_number']=current_issue["number"]                          # Get issue number
            data['created_at']= current_issue["created_at"][0:10]                 # Get created date of issue
            if current_issue["closed_at"] == None:
                data['closed_at']= current_issue["closed_at"]
            else:
                data['closed_at']= current_issue["closed_at"][0:10]               # Get closed date of issue
            for label in current_issue["labels"]:
                label_name.append(label["name"])                                  # Get label name of issue
            data['labels']= label_name
            data['State'] = current_issue["state"]                                # It gives state of issue like closed
            data['Author'] = current_issue["user"]["login"]                       # Get Author of issue
            out=json.dumps(data)                                                  # save this all information to a JSON
            f.write(out+ '\n')
f.close()
```

Your application will open with localhost in jupyter notebook

# Set Generated Github API Token in Script

- Double click on import_issues.ipynb file and it will open in browser.
- Set your generated Github API Token in “GITHUB_TOKEN” variable.

In [4]:
```
# pip install github3.py command for install github3
# execute pip install github3.py command from the the command window/terminal

import github3, json
```

In [5]:
```
# Get an API key for GitHub and set it as GITHUB_TOKEN
# Here is the URL to generate your GITHUB_TOKEN
# https://help.github.com/articles/creating-an-access-token-for-command-line-use/

GITHUB_TOKEN = 'INSERT_YOUR_SECRET_GITHUB_API_KEY_HERE'
ORG = 'SPM587FA18'
REPO = 'SCM587FA18'
FILENAME_ISSUES = ORG + 'issues.json'
```

Add your Github API Token here

In [6]:
```
gh = github3.login(token=GITHUB_TOKEN)

f = open(FILENAME_ISSUES, 'w')
for issue in gh.search_issues('type:issue repo:SPM587FA18/SCM587FA18'):          # Find issues from given Repo
            label_name=[]
            data={}
            current_issue = issue.as_json()
```

# Run import_issues.ipynb file

Import_Issues

localhost:8888/notebooks/Import_Issues.ipynb

jupyter Import_Issues Last Checkpoint: 8 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 3

Run Cells
Run Cells and Select Below
Run Cells and Insert Below
Run All
Run All Above
Run All Below
Cell Type
Current Outputs
All Output

In [4]:
```
# pip install ; install github3
# execute pip ; nd from the the command window/terminal

import github3
```

In [5]:
```
# Get an API k as GITHUB_TOKEN
# Here is the HUB_TOKEN
# https://help ating-an-access-token-for-command-line-use/

GITHUB_TOKEN = HUB_API_KEY_HERE'
ORG = 'SPM587FA18'
REPO = 'SCM587FA18'
FILENAME_ISSUES = ORG + 'issues.json'
```

In [6]:
```
gh = github3.login(token=GITHUB_TOKEN)

f = open(FILENAME_ISSUES, 'w')
for issue in gh.search_issues('type:issue repo:SPM587FA18/SCM587FA18'):          # Find issues from given Repo
            label_name=[]
```

Select Run All

Each cell is represented by a box

# Json File

- After running import_issues.ipynb Go to your python File location and check one file SPM587FA18issues.json is created.

| Name | Date modified | Type | Size |
|---|---|---|---|
| .ipynb_checkpoints | 11/7/2018 3:43 PM | File folder | |
| charting_issues.ipynb | 11/6/2018 9:32 PM | IPython notebook | 165 KB |
| Import_Issues.ipynb | 11/7/2018 11:27 PM | IPython notebook | 3 KB |
| SPM587FA18issues.json | 11/7/2018 11:36 PM | JSON File | 1 KB |

SPM587FA18issues.json file

# Json File

- Open SPM587FA18issues.json file.
- You will see all information about issues like Author, Created_at, labels of issue, issue number, closed_at etc.

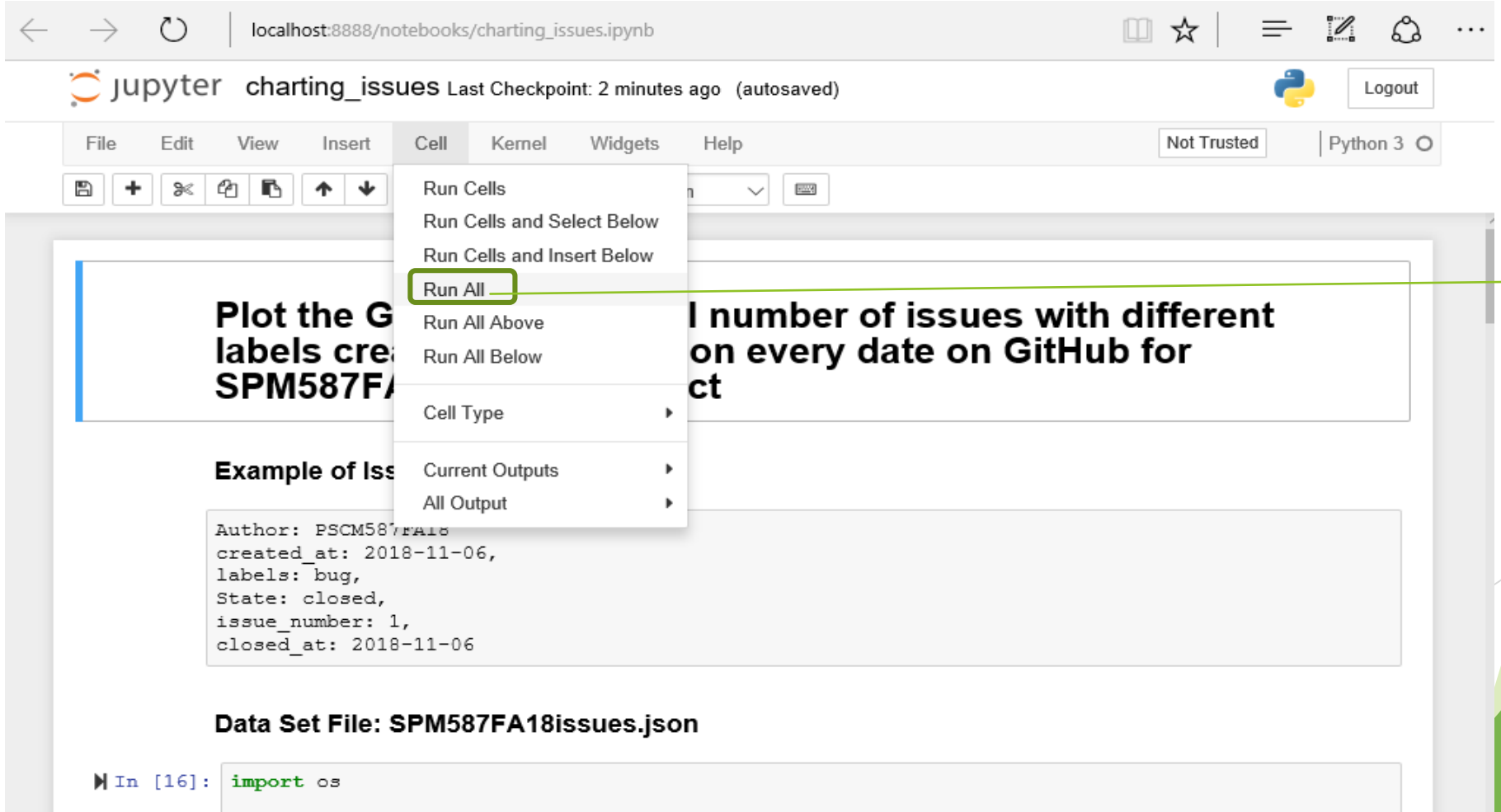
SPM587FA18issues.json - Notepad

File Edit Format View Help


```
{"Author": "AFA18SCM91A", "created_at": "2018-11-14", "labels": ["Category:Bug", "DetectionPhase:Testing", "OriginationPhase:Requirements", "Priority:Major",
{"Author": "AFA18SCM08J", "created_at": "2018-11-14", "labels": ["Category:Bug", "DetectionPhase:Design", "OriginationPhase:Requirements", "Priority:Critical"
{"Author": "AFA18SCM64A", "created_at": "2018-11-11", "labels": ["Category:Bug", "DetectionPhase:Testing", "OriginationPhase:Design", "Priority:Major", "Statu
{"Author": "AFA18SCM41A", "created_at": "2018-11-10", "labels": ["Category:Inquiry", "DetectionPhase:Field", "OriginationPhase:Coding", "Priority:High", "Stat
{"Author": "AFA18SCM88F", "created_at": "2018-11-10", "labels": ["Category:Bug", "DetectionPhase:Testing", "OriginationPhase:Requirements", "Priority:Critical
{"Author": "CFA18SCM37A", "created_at": "2018-11-10", "labels": ["Category:Bug", "DetectionPhase:Testing", "OriginationPhase:Requirements", "Priority:Critical
{"Author": "AFA18SCM18K", "created_at": "2018-11-10", "labels": ["Category:Inquiry", "DetectionPhase:Field", "OriginationPhase:Documentation", "Priority:Mediu
{"Author": "AFA18SCM18K", "created_at": "2018-11-10", "labels": ["Category:Inquiry", "DetectionPhase:Field", "OriginationPhase:Coding", "Priority:High", "Stat
{"Author": "CFA18SCM93S", "created_at": "2018-11-10", "labels": ["Category:Bug", "DetectionPhase:Testing", "OriginationPhase:Requirements", "Priority:Major",
{"Author": "SFA18SCM46N", "created_at": "2018-11-10", "labels": ["Category:Inquiry", "DetectionPhase:Field", "OriginationPhase:Coding", "Priority:High", "Stat
{"Author": "KFA18SCM32K", "created_at": "2018-11-10", "labels": ["Category:Inquiry", "DetectionPhase:Field", "OriginationPhase:Documentation", "Priority:High"
{"Author": "DFA18SCM62P", "created_at": "2018-11-10", "labels": ["Category:Inquiry", "DetectionPhase:Field", "OriginationPhase:Coding", "Priority:Medium", "St
{"Author": "MFA18SCM69L", "created_at": "2018-11-10", "labels": ["Category:Bug", "DetectionPhase:Testing", "OriginationPhase:Design", "Priority:Critical", "St
{"Author": "MFA18SCM37H", "created_at": "2018-11-10", "labels": ["Category:Bug", "DetectionPhase:Testing", "OriginationPhase:Design", "Priority:Major", "Statu
{"Author": "AFA18SCM06M", "created_at": "2018-11-10", "labels": ["Category:Inquiry", "DetectionPhase:Field", "OriginationPhase:Coding", "Priority:Medium", "St
{"Author": "JFA18SCM87M", "created_at": "2018-11-10", "labels": ["Category:Enhancement", "DetectionPhase:Testing", "OriginationPhase:Design", "Priority:Critic
{"Author": "CFA18SCM73P", "created_at": "2018-11-10", "labels": ["Category:Inquiry", "DetectionPhase:Field", "OriginationPhase:Testing", "Priority:Medium", "S
{"Author": "WFA18SCM51Y", "created_at": "2018-11-10", "labels": ["Category:Bug", "DetectionPhase:Testing", "OriginationPhase:Requirements", "Priority:Critical
{"Author": "KFA18SCM70P", "created_at": "2018-11-10", "labels": ["Category:Inquiry", "DetectionPhase:Field", "OriginationPhase:Coding", "Priority:Medium", "St
{"Author": "SFA18SCM73G", "created_at": "2018-11-10", "labels": ["Category:Enhancement", "DetectionPhase:Testing", "OriginationPhase:Design", "Priority:Critic
```

# Run charting_issues.ipynb file

- Double click on charting_issues.ipynb file and it will open in browser.
- Run charting_issues.ipynb file.




Select Run All

# charting_issues.ipynb file

In [26]:
```
# Sanity test: print first 10 rows in our DataFrame

issues_df
```

Out[26]:

| | Author | State | closed_at | created_at | issue_number | labels |
|---|---|---|---|---|---|---|
| 0 | AFA18SCM91A | closed | 2018-11-21 | 2018-11-14 | 524 | [Category:Bug, DetectionPhase:Testing, Origina... |
| 1 | AFA18SCM08J | closed | 2018-11-14 | 2018-11-14 | 523 | [Category:Bug, DetectionPhase:Design, Originat... |
| 2 | AFA18SCM64A | closed | 2018-11-12 | 2018-11-11 | 521 | [Category:Bug, DetectionPhase:Testing, Origina... |
| 3 | AFA18SCM41A | closed | 2018-11-13 | 2018-11-10 | 518 | [Category:Inquiry, DetectionPhase:Field, Origi... |
| 4 | AFA18SCM88F | closed | 2018-11-12 | 2018-11-10 | 516 | [Category:Bug, DetectionPhase:Testing, Origina... |
| 5 | CFA18SCM37A | closed | 2018-11-12 | 2018-11-10 | 511 | [Category:Bug, DetectionPhase:Testing, Origina... |
| 6 | AFA18SCM18K | open | None | 2018-11-10 | 509 | [Category:Inquiry, DetectionPhase:Field, Origi... |
| 7 | AFA18SCM18K | closed | 2018-11-10 | 2018-11-10 | 506 | [Category:Inquiry, DetectionPhase:Field, Origi... |
| 8 | CFA18SCM93S | closed | 2018-11-12 | 2018-11-10 | 505 | [Category:Bug, DetectionPhase:Testing, Origina... |
| 9 | SFA18SCM46N | closed | 2018-11-12 | 2018-11-10 | 511 | [Category:Inquiry, DetectionPhase:Field, Origi... |
| 10 | KFA18SCM32K | closed | 2018-11-12 | 2018-11-10 | 495 | [Category:Inquiry, DetectionPhase:Field, Origi... |
| 11 | DFA18SCM62P | closed | 2018-11-12 | 2018-11-10 | 494 | [Category:Inquiry, DetectionPhase:Field, Origi... |
| 12 | MFA18SCM69L | closed | 2018-11-12 | 2018-11-10 | 492 | [Category:Bug, DetectionPhase:Testing, Origina... |

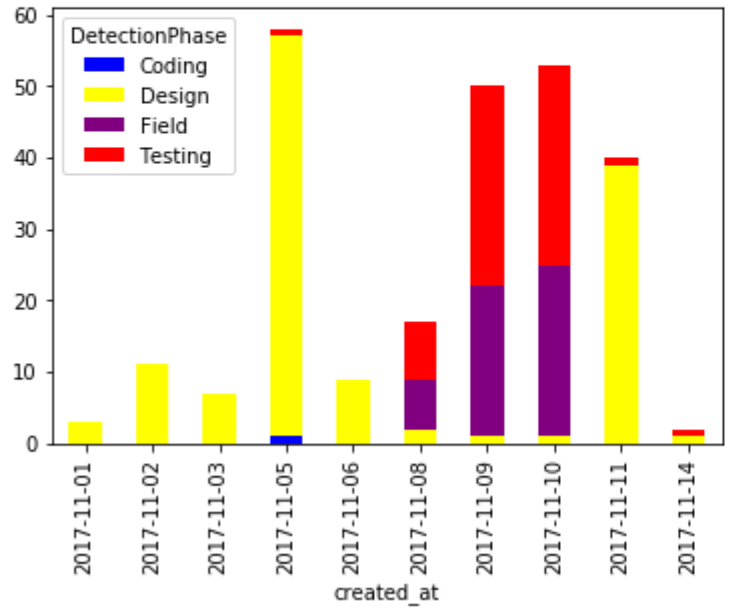
Gives all information about issues in the table

# charting_issues.ipynb file

In [20]:
```
# Plot in Bar Chart the total number of issues created every day for every Detaction Phase

LabelsReviewedByDate = wrangled_issues_df.groupby(['created_at','DetectionPhase']).created_at.count()

dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,  color=['blue','yellow', 'purple', 'red',
```



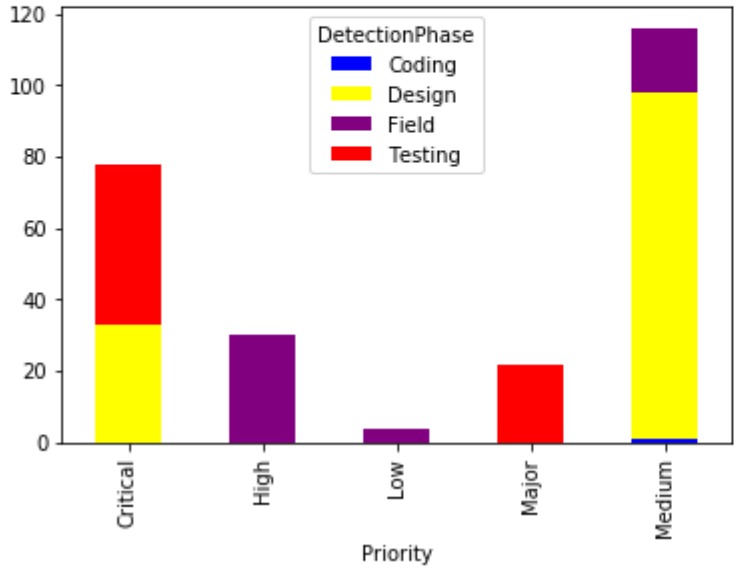

It shows total number of issues created every day for every Detection Phase

# charting_issues.ipynb file

```
In [21]: # Plot in Bar Chart the total number of issues created for every Phase based on thier priorites

LabelsReviewedByDate = wrangled_issues_df.groupby(['Priority','DetectionPhase']).created_at.count()

dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,  color=['blue','yellow', 'purple', 'red',
```



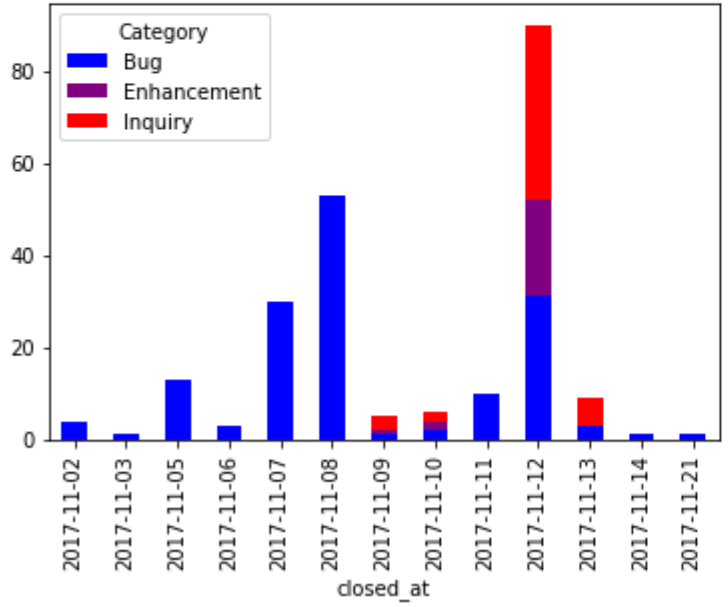

It shows total number of issues created for every Phase based on their Priorities

# charting_issues.ipynb file

In [22]:
```
# Plot in Bar Chart the total number of issues closed every day for every Category

LabelsReviewedByDate = wrangled_issues_df.groupby(['closed_at','Category']).closed_at.count()

dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,  color=['blue', 'purple', 'red'], grid=Fal
```




It shows total number of issues closed every day for every Category

# Questions?