Task 0

Software Installation (for Windows OS)

[*Last Updated on:* ***5th October 2020, 20:34 Hrs***]

This document contains instructions to install following software/libraries on **Windows OS**:

- **Miniconda for Python 3.8**
- **OpenCV**
- **NumPy**
- **Matplotlib**
- **Jupyter Notebook, JupyterLab**
- **CoppeliaSim 4.0.0**
- **VS Code** *(optional but highly recommended)*

> The installation of all software/libraries has been tested on **Windows 8** and **10**. We recommend you to use one of these versions of **Windows OS**. These software have to be installed **ONLY ON 64-bit OS**.

1. Miniconda for Python 3.8

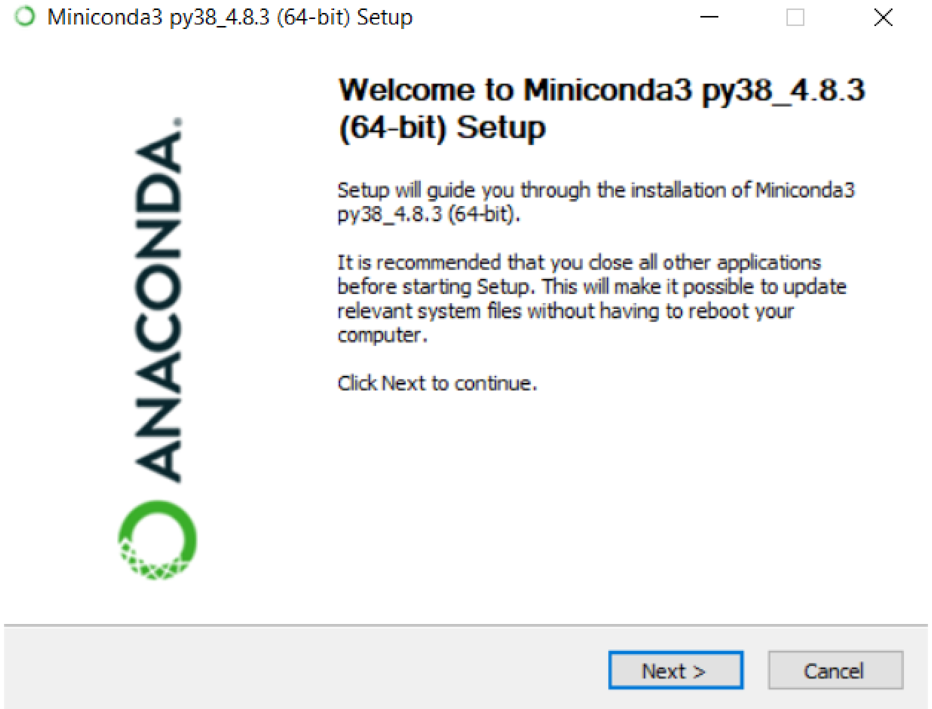
- In this section, we will be installing Miniconda and use it to create a Python 3.8 virtual environment in next section.
- We will be using **Python 3.8** in this theme. You don't have to install Python separately. It come by default with Miniconda installation.
- Conda is a package management system, created specifically to assist working with Python packages in a cross-platform fashion. Conda also has the concept of a virtual environment, which is an independent, self-contained install of Python and packages.
- There are two options for getting Conda: Anaconda and Miniconda. You can read in detail about these and which one to prefer, over here: reference1 and reference2.

  Note: If you already have Anaconda or Miniconda pre-installed in your system, kindly skip this section.

A. Installation Steps

- Download Miniconda for Python 3.8 from here (*file size - 55.7MB*).
- Navigate to the downloaded `.exe` file and double-click it. You would be greeted with a screen as shown in Figure 1.

Welcome to NB theme!
Rulebook
Task 0
(1a) Software Installation (Ubuntu)
(1b) Software Installation (Windows)
(2) Test Setup
(3) Learning Resources / Tutorials
(4) Solving Assignment
Task 1
Task 2
Task 3
Task 4
Task 5
Practice Task
Instructions for Task 6
Task 6 Scene Details
Coding Standard
Git and GitHub
Live Session 1 - 24th October 2020
Live Session 2 - 21st November 2020
Live Session 3 - 12th December 2020
Live Session 4 - 10th January 2021
Changelog



Miniconda3 py38_4.8.3 (64-bit) Setup

Welcome to Miniconda3 py38_4.8.3 (64-bit) Setup

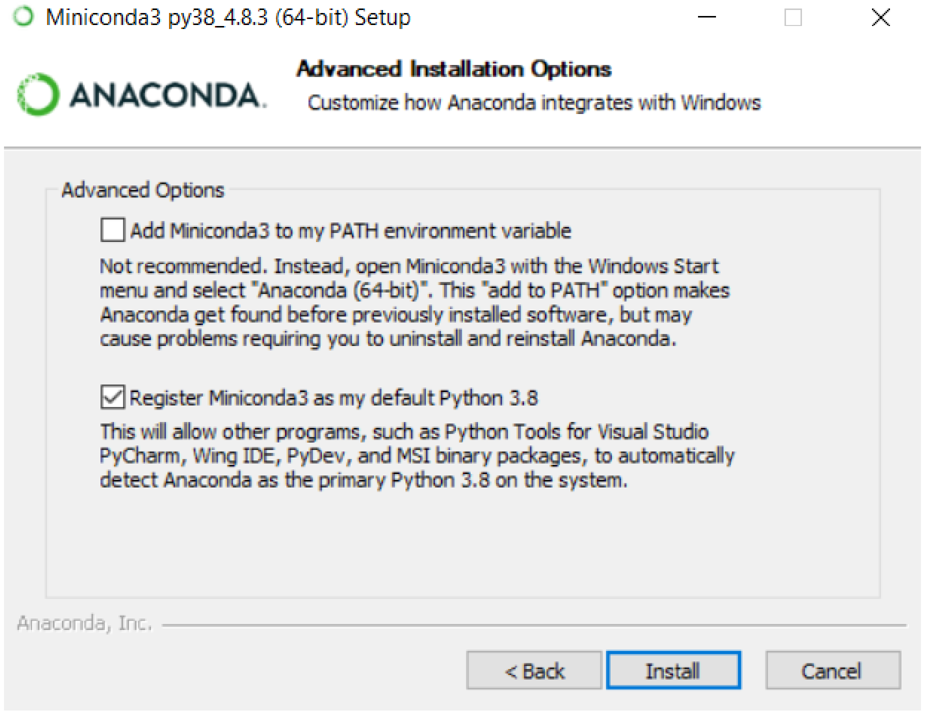
Setup will guide you through the installation of Miniconda3 py38_4.8.3 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Figure 1: Miniconda Windows Installer Start

- Click on **I Agree** to accept the terms of agreement.
- It is best and recommended to install Miniconda for the local user, which does not require administrator permissions and is the most robust type of installation. Thus, leave the default **Just Me** selected and click **Next**.
- Leave the destination folder to install Miniconda untouched. **Note:** If the directory path contains any spaces, it will throw a Warning ! saying that the directory name should no contai spaces but that's okay, click OK.
- At this point, as shown in Figure 2, you have two important options:
    - **First:** You can optionally add Miniconda to your PATH environment variable, if you want to access Miniconda through the Windows Command Prompt. We recommend **not adding** it, since this can interfere with other software.
    - **Second:** You can register Miniconda as your default Python 3.8 and hence will be detected by other IDEs and tools. We recommend to **accept** this default check to registe Miniconda, unless you plan on installing and running multiple versions of Miniconda or Anaconda.



Miniconda3 py38_4.8.3 (64-bit) Setup

Advanced Installation Options

Customize how Anaconda integrates with Windows

Advanced Options

☐ Add Miniconda3 to my PATH environment variable

Not recommended. Instead, open Miniconda3 with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

☑ Register Miniconda3 as my default Python 3.8

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.8 on the system.

Figure 2: Miniconda Advanced Installation Options

- Click on **Install**. Now sit back with ease as the installation process will take some time. Click **Next** when the installation completes. Skip the optional installation of Microsoft Visual Studio Code by clicking Skip, as we will install it manually later.
- Un-check the two boxes as shown in Figure 3 and click **Finish**.

Welcome to NB theme!

Rulebook ›

Task 0 ⌄

(1a) Software Installation (Ubuntu)

(1b) Software Installation (Windows)

(2) Test Setup

(3) Learning Resources / Tutorials

(4) Solving Assignment

Task 1 ›

Task 2 ›

Task 3 ›

Task 4 ›

Task 5 ›

Practice Task

Instructions for Task 6

Task 6 Scene Details

Coding Standard

Git and GitHub

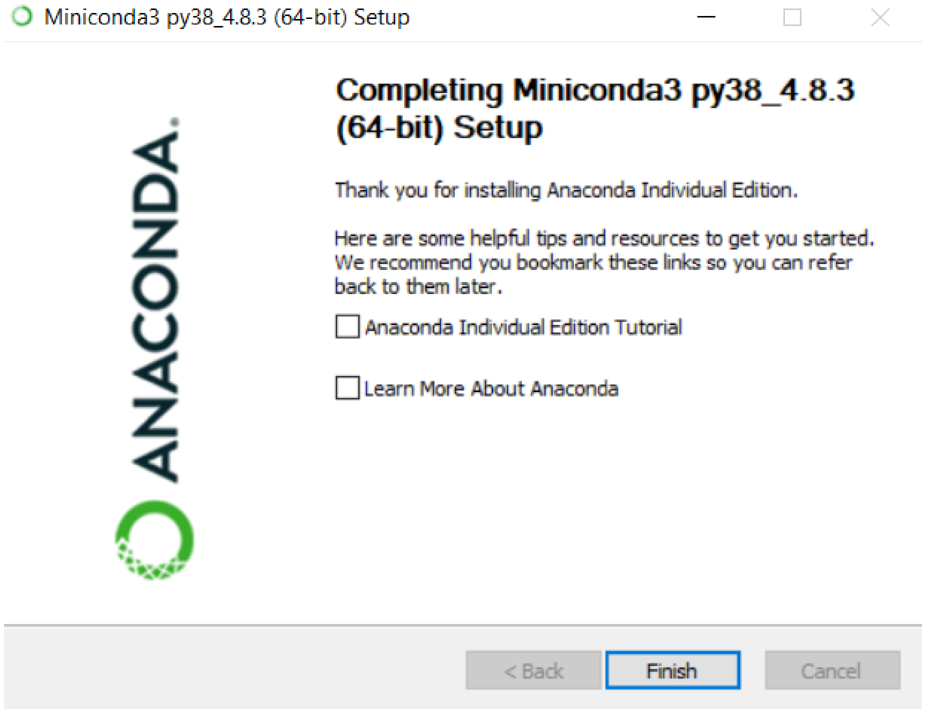
Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog


Miniconda3 py38_4.8.3 (64-bit) Setup

Completing Miniconda3 py38_4.8.3 (64-bit) Setup

Thank you for installing Anaconda Individual Edition.

Here are some helpful tips and resources to get you started. We recommend you bookmark these links so you can refer back to them later.

□ Anaconda Individual Edition Tutorial

□ Learn More About Anaconda

Figure 3: Miniconda Installation Complete

- Cheers, **Miniconda for Python 3.8** is successfully installed. Let's verify now.

B. Verification Steps

- Search for **Anaconda Prompt (miniconda3)** in Start menu. Open it.
- Type the following command in the prompt which has popped up.

```
conda info
```

You will get the information on Miniconda as in Figure 4.

Welcome to NB theme!

Rulebook

Task 0

(1a) Software Installation (Ubuntu)

(1b) Software Installation (Windows)

(2) Test Setup

(3) Learning Resources / Tutorials

(4) Solving Assignment

Task 1

Task 2

Task 3

Task 4

Task 5

Practice Task

Instructions for Task 6

Task 6 Scene Details

Coding Standard

Git and GitHub

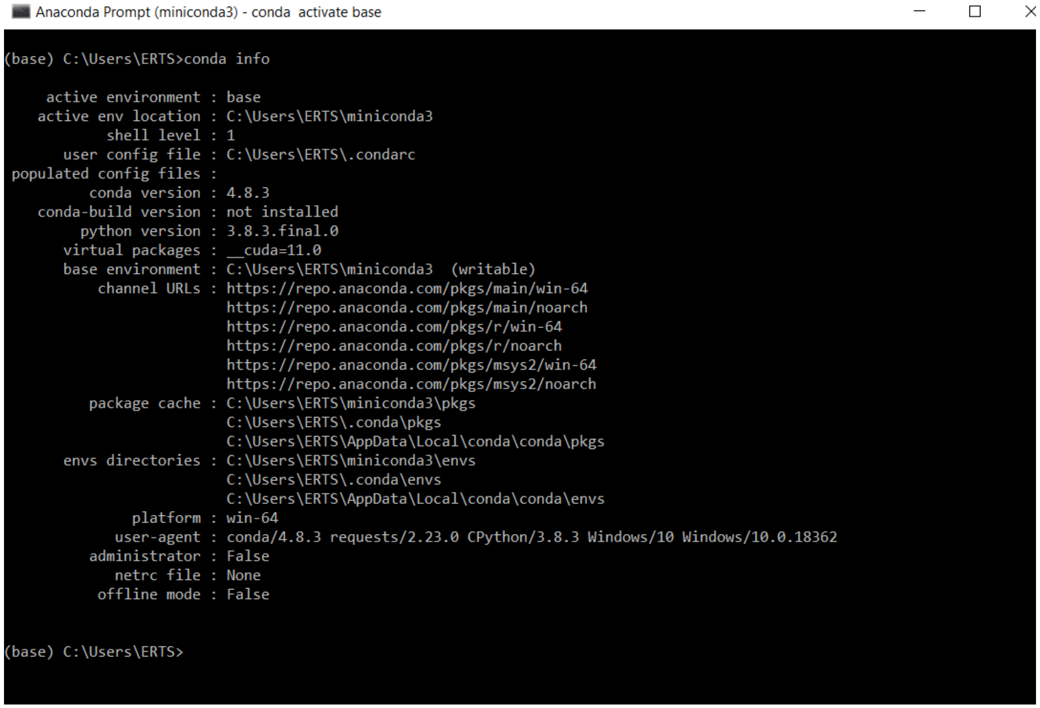
Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog



```
(base) C:\Users\ERTS>conda info

     active environment : base
    active env location : C:\Users\ERTS\miniconda3
            shell level : 1
       user config file : C:\Users\ERTS\.condarc
 populated config files :
          conda version : 4.8.3
    conda-build version : not installed
         python version : 3.8.3.final.0
       virtual packages : __cuda=11.0
       base environment : C:\Users\ERTS\miniconda3  (writable)
           channel URLs : https://repo.anaconda.com/pkgs/main/win-64
                          https://repo.anaconda.com/pkgs/main/noarch
                          https://repo.anaconda.com/pkgs/r/win-64
                          https://repo.anaconda.com/pkgs/r/noarch
                          https://repo.anaconda.com/pkgs/msys2/win-64
                          https://repo.anaconda.com/pkgs/msys2/noarch
          package cache : C:\Users\ERTS\miniconda3\pkgs
                          C:\Users\ERTS\.conda\pkgs
                          C:\Users\ERTS\AppData\Local\conda\conda\pkgs
       envs directories : C:\Users\ERTS\miniconda3\envs
                          C:\Users\ERTS\.conda\envs
                          C:\Users\ERTS\AppData\Local\conda\conda\envs
               platform : win-64
             user-agent : conda/4.8.3 requests/2.23.0 CPython/3.8.3 Windows/10 Windows/10.0.18362
          administrator : False
             netrc file : None
           offline mode : False


(base) C:\Users\ERTS>
```

Figure 4: Output of conda info command

- Miniconda creates a default Python environment named ***base*** at the installation directory, let' check this out. Type **one of the following three** commands to find all the existing virtual environments on your system with their path location:

```
conda env list
```

OR

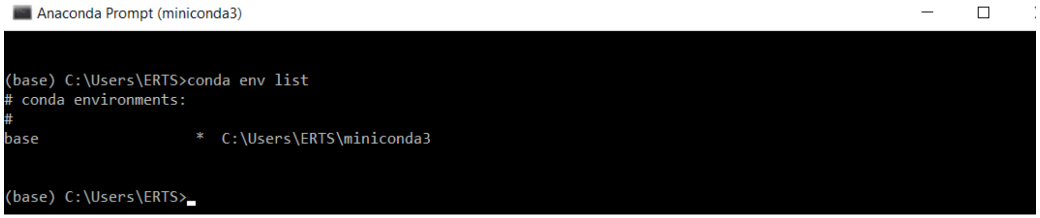
```
conda info --envs
```

OR

```
conda info -e
```

Since for now we just have one default environment **base** existing, there is one entry in Figure 5. The ***asterisk*** sign " * ", against the name indicates the current active environment, so it is ***base*** in this case.

Welcome to NB theme!
Rulebook
Task 0
(1a) Software Installation (Ubuntu)
(1b) Software Installation (Windows)
(2) Test Setup
(3) Learning Resources / Tutorials
(4) Solving Assignment
Task 1
Task 2
Task 3
Task 4
Task 5
Practice Task
Instructions for Task 6
Task 6 Scene Details
Coding Standard
Git and GitHub
Live Session 1 - 24th October 2020
Live Session 2 - 21st November 2020
Live Session 3 - 12th December 2020
Live Session 4 - 10th January 2021
Changelog



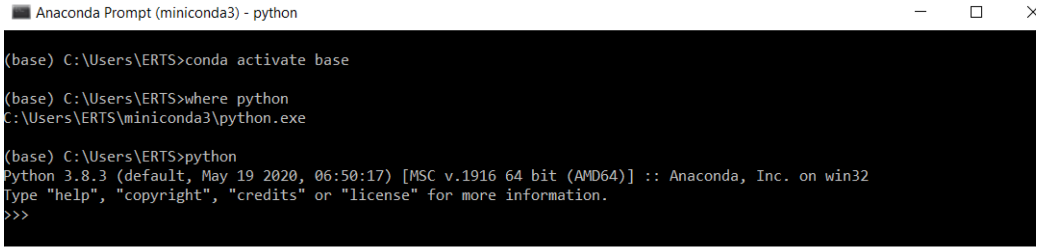
```
(base) C:\Users\ERTS>conda env list
# conda environments:
#
base                  *  C:\Users\ERTS\miniconda3


(base) C:\Users\ERTS>_
```

Figure 5: Output of conda env list command

- When any environment is activated, its name appears in round braces **(env_name)** before the prompt. As in previous step, the **(base)** appears at the start of the prompt.
- To activate the default Python environment that comes with Miniconda installation, type in following in sequence:

```
conda activate base
where python
python
```

You will see output similar to Figure 6. What we did here is, first we invoked the default environment, then checked whether the prompt is able to detect the path of Python that cam installed with Miniconda and at last we called the Python interpreter or console, you can see the very first message has Anaconda word which justifies it.



```
(base) C:\Users\ERTS>conda activate base

(base) C:\Users\ERTS>where python
C:\Users\ERTS\miniconda3\python.exe

(base) C:\Users\ERTS>python
Python 3.8.3 (default, May 19 2020, 06:50:17) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 6: Output of where python and python

- The Python version (***3.8.3***) might be different at your end but that's okay as long as it is ***3.8.x***. Type `exit()` to come out of Python console.
- Just as we use the command: `conda activate env_name` to activate the Python environment named **env_name**; to deactivate any active environment, type: `conda deactivate` . The **(env_name)** will now disappear from the start of prompt.

```
conda deactivate
```

Welcome to NB theme!

Rulebook ›

Task 0 ⌄

(1a) Software Installation (Ubuntu)

(1b) Software Installation (Windows)

(2) Test Setup

(3) Learning Resources / Tutorials

(4) Solving Assignment

Task 1 ›

Task 2 ›

Task 3 ›

Task 4 ›

Task 5 ›

Practice Task

Instructions for Task 6

Task 6 Scene Details

Coding Standard

Git and GitHub

Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog

2. OpenCV, NumPy, Matplotlib, Jupyter Notebook, JupyterLab

- In this section, let's create a virtual environment of **Python 3.8** with all the necessary package
- We will then see how to use these packages or tools in brief.

A. Setting up a Virtual Environment using Miniconda

- The necessary packages or libraries, tools are encapsulated into the **environment.yml** file which can be downloaded from **here** (*file size - 174 bytes*). Right-click on the hyperlink and select **Save Link As...** option to download.

- In the Miniconda Prompt, navigate to the folder where the **environment.yml** file is downloaded. Create a new environment of the name `NB_<Team-ID>` by typing the command:

  ```
  conda env create --name NB_<Team-ID> -f environment.yml
  ```

  Replace `<Team-ID>` with your ***Team ID***. **Note:** There should be no space in the environment name.

- For instance, if your Team ID is ***9999***, type:

  ```
  conda env create --name NB_9999 -f environment.yml
  ```

  You will see the output similar to Figure 7.

  Note: Make sure you have good Internet connection while running this command, as it downloads and installs all packages listed in the environment.yml file.

Welcome to NB theme!

Rulebook

Task 0

(1a) Software Installation (Ubuntu)

(1b) Software Installation (Windows)

(2) Test Setup

(3) Learning Resources / Tutorials

(4) Solving Assignment

Task 1

Task 2

Task 3

Task 4

Task 5

Practice Task

Instructions for Task 6

Task 6 Scene Details

Coding Standard

Git and GitHub

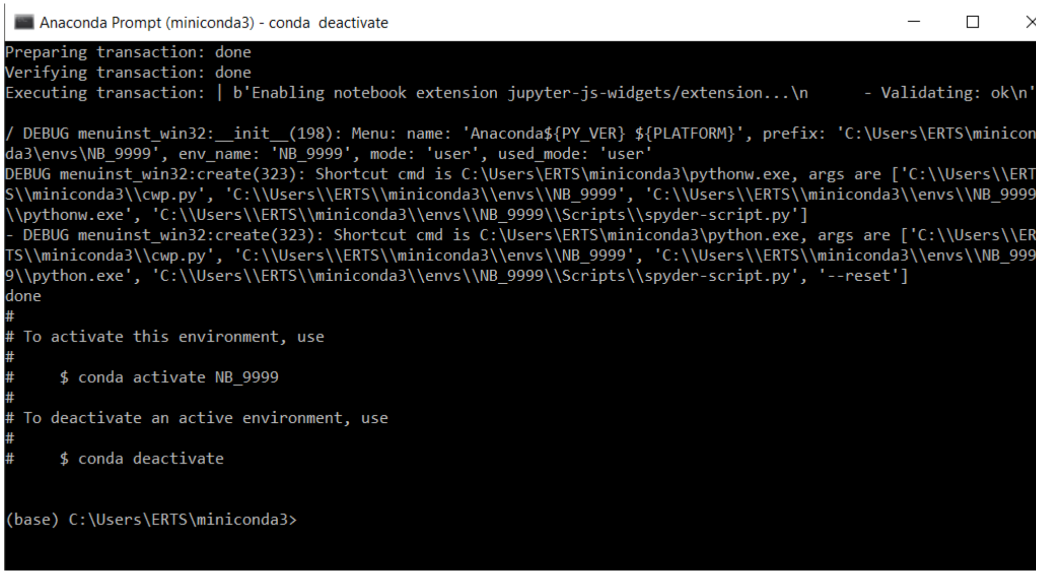
Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog

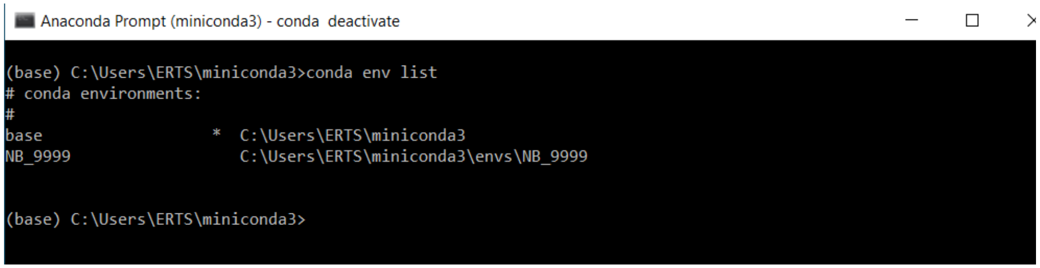


```
Preparing transaction: done
Verifying transaction: done
Executing transaction: | b'Enabling notebook extension jupyter-js-widgets/extension...\n      - Validating: ok\n'

/ DEBUG menuinst_win32:__init__(198): Menu: name: 'Anaconda${PY_VER} ${PLATFORM}', prefix: 'C:\Users\ERTS\minicon
da3\envs\NB_9999', env_name: 'NB_9999', mode: 'user', used_mode: 'user'
DEBUG menuinst_win32:create(323): Shortcut cmd is C:\Users\ERTS\miniconda3\pythonw.exe, args are ['C:\\Users\\ERT
S\\miniconda3\\cwp.py', 'C:\\Users\\ERTS\\miniconda3\\envs\\NB_9999', 'C:\\Users\\ERTS\\miniconda3\\envs\\NB_9999
\\pythonw.exe', 'C:\\Users\\ERTS\\miniconda3\\envs\\NB_9999\\Scripts\\spyder-script.py']
- DEBUG menuinst_win32:create(323): Shortcut cmd is C:\Users\ERTS\miniconda3\python.exe, args are ['C:\\Users\\ER
TS\\miniconda3\\cwp.py', 'C:\\Users\\ERTS\\miniconda3\\envs\\NB_9999', 'C:\\Users\\ERTS\\miniconda3\\envs\\NB_999
9\\python.exe', 'C:\\Users\\ERTS\\miniconda3\\envs\\NB_9999\\Scripts\\spyder-script.py', '--reset']
done
#
# To activate this environment, use
#
#     $ conda activate NB_9999
#
# To deactivate an active environment, use
#
#     $ conda deactivate


(base) C:\Users\ERTS\miniconda3>
```

Figure 7: Output of conda env create

B. Verifying the newly created Virtual Environment

- Let's verify the previous step. Type: **`conda env list`** now and you should now expect the environment name **`NB_<Team-ID>`** listed in the output as in Figure 8.



```
(base) C:\Users\ERTS\miniconda3>conda env list
# conda environments:
#
base                  *  C:\Users\ERTS\miniconda3
NB_9999                  C:\Users\ERTS\miniconda3\envs\NB_9999


(base) C:\Users\ERTS\miniconda3>
```

Figure 8: Output of conda env list after environment creation

- The **environment.yml** file used to setup the virtual environment has following packages, tool encapsulated:
  - **OpenCV**, **NumPy**, **Matplotlib** - these packages are useful for Image Processing, Array manipulations and Plotting.
  - **Jupyter Notebook, JupyterLab** - these tools extend the console-based approach to interactive computing, providing a web-based application suitable for developing, documenting and executing code, as well as communicating the results.

- You can find more commands and know how to install a new package inside the environment clone the environment, etc. in the **Conda Cheat Sheet**.

C. Using the packages and tools

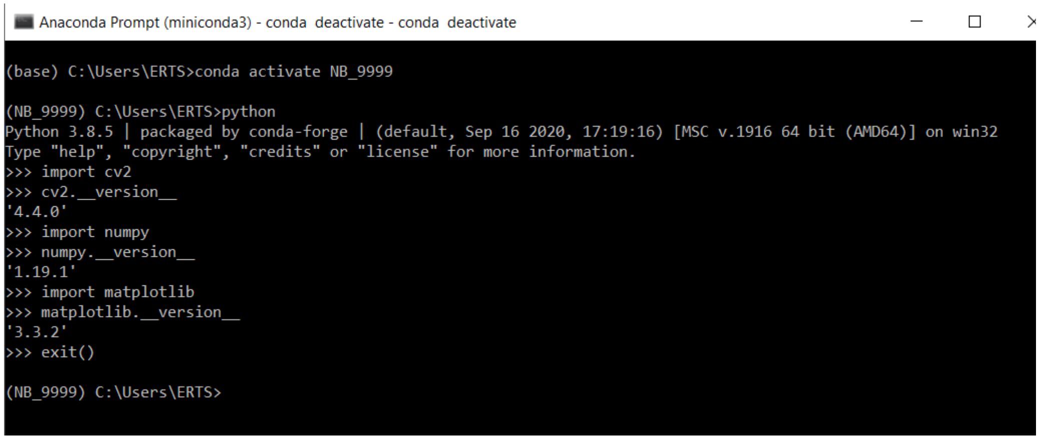
- **OpenCV**, **NumPy**, **Matplotlib**
  - Activate the environment `NB_9999` in Anaconda Prompt, and invoke the Python interpreter.
  - Enter the following commands:

```
conda activate NB_9999
python
```

```
import cv2
cv2.__version__
import numpy
numpy.__version__
import matplotlib
matplotlib.__version__
exit()
```

  - You will get the version numbers of each package in the outptut as shown in Figure 9. Th version numbers of some packages at your end **maybe** different.

- Once you get the similar results, exit the Python interpreter by typing `exit()` .


Anaconda Prompt (miniconda3) - conda deactivate - conda deactivate

```
(base) C:\Users\ERTS>conda activate NB_9999

(NB_9999) C:\Users\ERTS>python
Python 3.8.5 | packaged by conda-forge | (default, Sep 16 2020, 17:19:16) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'4.4.0'
>>> import numpy
>>> numpy.__version__
'1.19.1'
>>> import matplotlib
>>> matplotlib.__version__
'3.3.2'
>>> exit()

(NB_9999) C:\Users\ERTS>
```

Figure 9: Output of version numbers of Python, NumPy, Matplotlib

Welcome to NB theme!

Rulebook

Task 0

(1a) Software Installation (Ubuntu)

(1b) Software Installation (Windows)

(2) Test Setup

(3) Learning Resources / Tutorials

(4) Solving Assignment

Task 1

Task 2

Task 3

Task 4

Task 5

Practice Task

Instructions for Task 6

Task 6 Scene Details

Coding Standard

Git and GitHub

Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog

- **Jupyter Notebook, JupyterLab**
  - Make sure the environment `NB_9999` is activated before you proceed.
  - To ensure our Jupyter Notebook has access to the newly created Virtual Environment, w need to execute the following:

    ```
    python -m ipykernel install --user --name=NB_9999
    ```

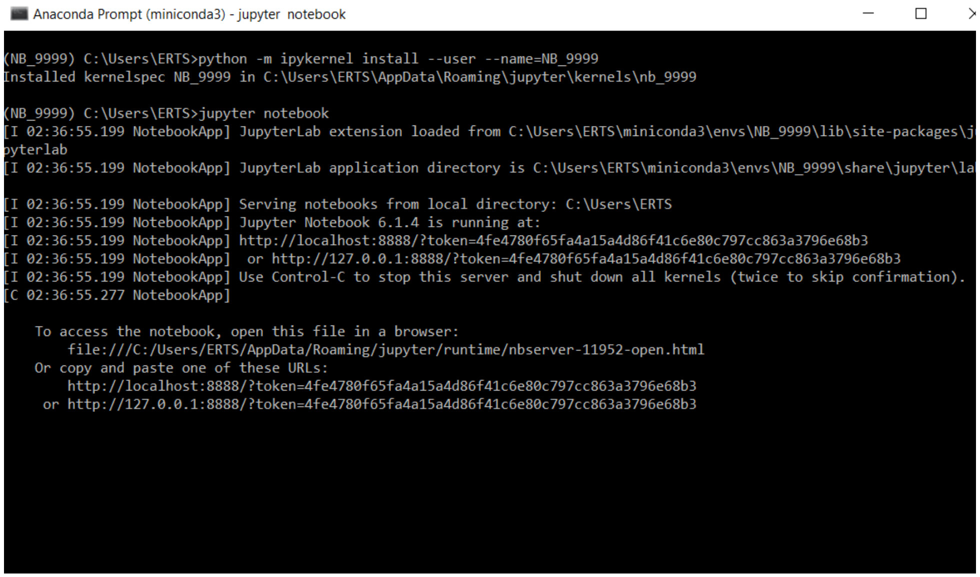
  - Now let's make sure our new environment is accessible through Jupyter Notebook.
  - Launch the **Jupyter Notebook** with the command:

    ```
    jupyter-notebook
    ```

    OR

    ```
    jupyter notebook
    ```

    This will start the Notebook server and open a default web browser to the URL of the we application, which will be: `http://127.0.0.1:8888` OR `http://localhost:8888` as belo in Figure 10.

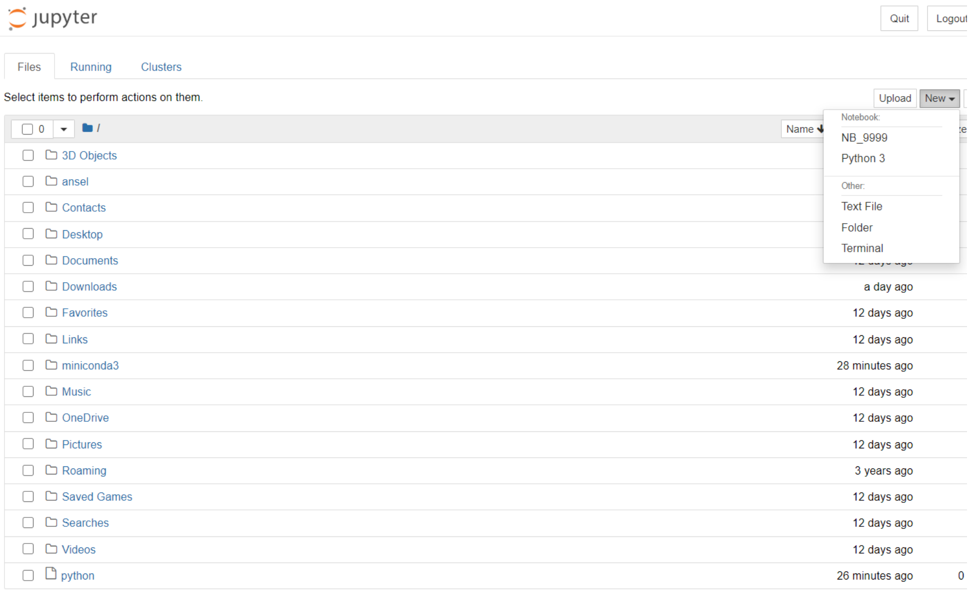


```
(NB_9999) C:\Users\ERTS>python -m ipykernel install --user --name=NB_9999
Installed kernelspec NB_9999 in C:\Users\ERTS\AppData\Roaming\jupyter\kernels\nb_9999

(NB_9999) C:\Users\ERTS>jupyter notebook
[I 02:36:55.199 NotebookApp] JupyterLab extension loaded from C:\Users\ERTS\miniconda3\envs\NB_9999\lib\site-packages\j
pyterlab
[I 02:36:55.199 NotebookApp] JupyterLab application directory is C:\Users\ERTS\miniconda3\envs\NB_9999\share\jupyter\la

[I 02:36:55.199 NotebookApp] Serving notebooks from local directory: C:\Users\ERTS
[I 02:36:55.199 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 02:36:55.199 NotebookApp] http://localhost:8888/?token=4fe4780f65fa4a15a4d86f41c6e80c797cc863a3796e68b3
[I 02:36:55.199 NotebookApp]  or http://127.0.0.1:8888/?token=4fe4780f65fa4a15a4d86f41c6e80c797cc863a3796e68b3
[I 02:36:55.199 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 02:36:55.277 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/ERTS/AppData/Roaming/jupyter/runtime/nbserver-11952-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=4fe4780f65fa4a15a4d86f41c6e80c797cc863a3796e68b3
     or http://127.0.0.1:8888/?token=4fe4780f65fa4a15a4d86f41c6e80c797cc863a3796e68b3
```

Figure 10: Output of jupyter notebook

  - The landing page of the Jupyter Notebook, the **dashboard** shows the **IPython Notebooks (.ipynb)** currently available in the directory from where the notebook server was started.

- In the top-right, under the **New** dropdown, you will be able to see your newly created environment as shown in Figure 11.



Select items to perform actions on them.

□ 3D Objects		
□ ansel		
□ Contacts		
□ Desktop		
□ Documents		
□ Downloads	a day ago	
□ Favorites	12 days ago	
□ Links	12 days ago	
□ miniconda3	28 minutes ago	
□ Music	12 days ago	
□ OneDrive	12 days ago	
□ Pictures	12 days ago	
□ Roaming	3 years ago	
□ Saved Games	12 days ago	
□ Searches	12 days ago	
□ Videos	12 days ago	
□ python	26 minutes ago	0

Figure 11: Jupyter Notebook able to detect the new environment

- Launch the **JupyterLab** using the below command with your conda environment activated.

```
jupyter-lab
```

OR

```
jupyter lab
```

- JupyterLab will open automatically in your browser just like the **Jupyter Notebook**. This will start the Notebook server and open a default web browser to the URL of the web application, which will be: `http://127.0.0.1:8888` OR `http://localhost:8888` as belo in Figure 12a and 12b.

Welcome to NB theme!

Rulebook

Task 0

(1a) Software Installation (Ubuntu)

(1b) Software Installation (Windows)

(2) Test Setup

(3) Learning Resources / Tutorials

(4) Solving Assignment

Task 1

Task 2

Task 3

Task 4

Task 5

Practice Task

Instructions for Task 6

Task 6 Scene Details

Coding Standard

Git and GitHub

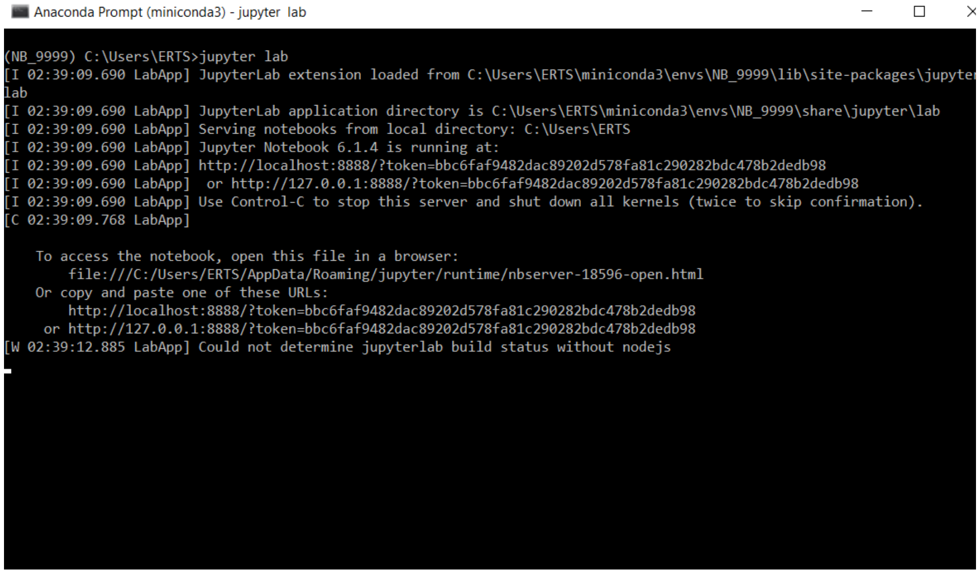
Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

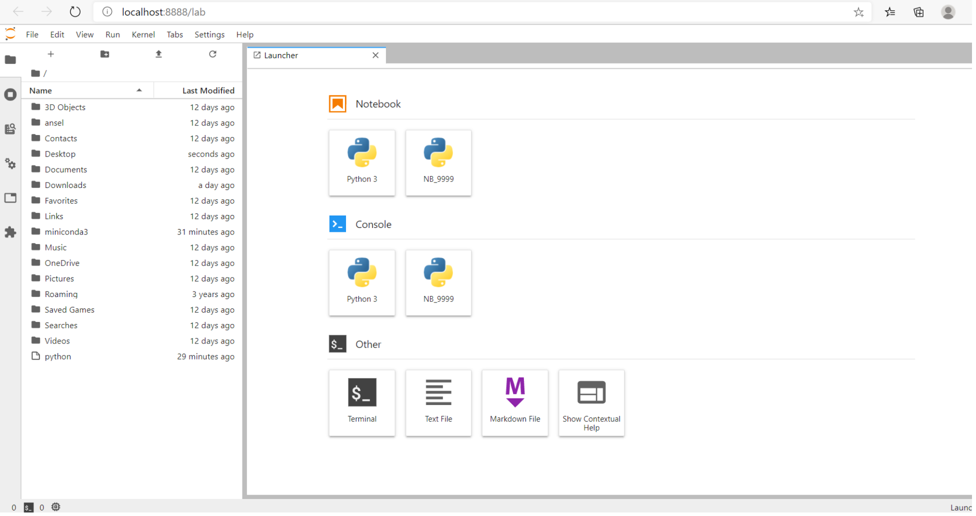
Changelog



```
(NB_9999) C:\Users\ERTS>jupyter lab
[I 02:39:09.690 LabApp] JupyterLab extension loaded from C:\Users\ERTS\miniconda3\envs\NB_9999\lib\site-packages\jupyter
lab
[I 02:39:09.690 LabApp] JupyterLab application directory is C:\Users\ERTS\miniconda3\envs\NB_9999\share\jupyter\lab
[I 02:39:09.690 LabApp] Serving notebooks from local directory: C:\Users\ERTS
[I 02:39:09.690 LabApp] Jupyter Notebook 6.1.4 is running at:
[I 02:39:09.690 LabApp] http://localhost:8888/?token=bbc6faf9482dac89202d578fa81c290282bdc478b2dedb98
[I 02:39:09.690 LabApp]  or http://127.0.0.1:8888/?token=bbc6faf9482dac89202d578fa81c290282bdc478b2dedb98
[I 02:39:09.690 LabApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 02:39:09.768 LabApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/ERTS/AppData/Roaming/jupyter/runtime/nbserver-18596-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=bbc6faf9482dac89202d578fa81c290282bdc478b2dedb98
     or http://127.0.0.1:8888/?token=bbc6faf9482dac89202d578fa81c290282bdc478b2dedb98
[W 02:39:12.885 LabApp] Could not determine jupyterlab build status without nodejs
```

Figure 12a: Output of jupyter lab

The **warning message**: *Could not determine jupyterlab build status without nodejs* can be ignored.



Name	Last Modified
3D Objects	12 days ago
ansel	12 days ago
Contacts	12 days ago
Desktop	seconds ago
Documents	12 days ago
Downloads	a day ago
Favorites	12 days ago
Links	12 days ago
miniconda3	31 minutes ago
Music	12 days ago
OneDrive	12 days ago
Pictures	12 days ago
Roaming	3 years ago
Saved Games	12 days ago
Searches	12 days ago
Videos	12 days ago
python	29 minutes ago

Figure 12b: Jupyter Lab able to detect the new environment

Note: Get acquainted yourself with the UI of Jupyter Notebook dashboard, how to create markdow and code cells and run them, keyboard shortcuts, etc. from the official documentation.

Voila! Your new Virtual Environment is setup successfully without any errors!

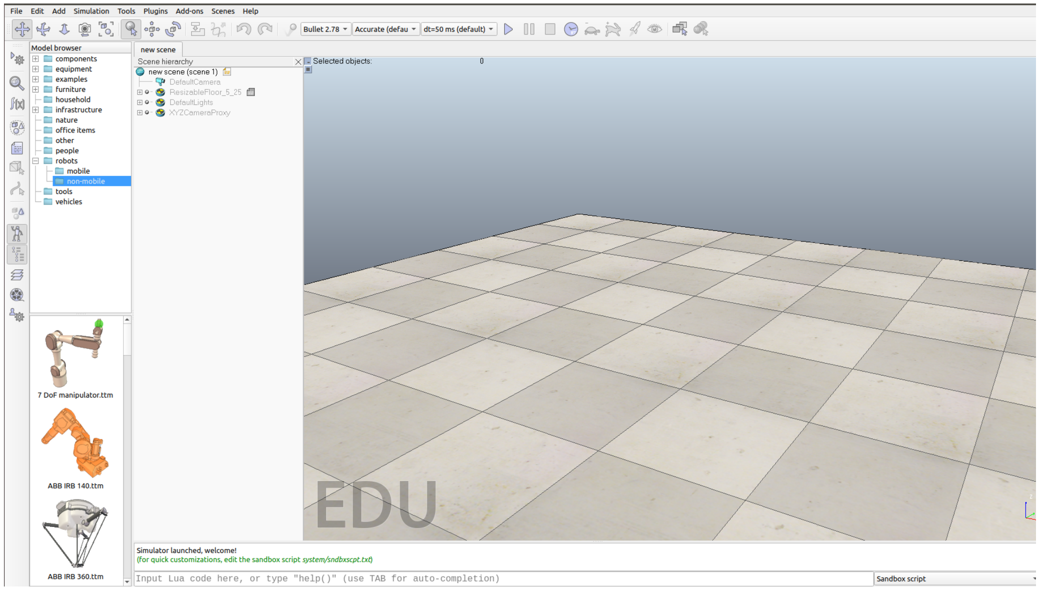
---

3. CoppeliaSim 4.0.0

- In this section, we will install CoppeliaSim Robotics Simulator Platform.
- The robot simulator CoppeliaSim is based on a distributed control architecture: each object/model can be individually controlled via an embedded script, a plugin, a ROS or BlueZero node, a remote API client, or a custom solution. This makes CoppeliaSim very versatile and ideal for multi-robot applications. Controllers can be written in C/C++, Python, Java, Lua, Matlab or Octave.
- CoppeliaSim is used for fast algorithm development, factory automation simulations, fast prototyping and verification, robotics related education, remote monitoring, safety double-checking, as digital twin, and much more. You can find a feature overview here.
- It can be used as a stand-alone application or can easily be embedded into a main client application: its small footprint and elaborate API makes CoppeliaSim an ideal candidate to embed into higher-level applications.

A. Installation Steps

- Download **CoppeliaSim Edu 4.0.0** for **Windows 64-bit OS** from here (*file size - 161MB*).
- Navigate to the downloaded `.exe` file and double-click it to install the software.
- Once the installation is complete, a shortcut icon to launch the software would have been created on your Desktop. Double-click to launch it. CoppeliaSim will open as shown in Figure 1 with the default scene loaded.



```
Simulator launched, welcome!
(for quick customizations, edit the sandbox script system/sndbxscpt.txt)
Input Lua code here, or type "help()" (use TAB for auto-completion)
```

Figure 13: CoppeliaSim software launch

---

4. Visual Studio Code *(optional but hightly recommended)*

- The below steps will increase your programming efficiency.
- Programmers spend a lot of time looking at documentation, re-factoring code, debugging issues, unit testing and styling code.
- What if you get a helping hand at this so that your code looks beautiful and you could debug issues faster? Read further to know more. Don't worry!
- Since using tooling may be a new thing to most of you it may look difficult but none of this is difficult. It's actually rather very very easy and you will become a better programmer the earlie you master these tools.
    - Download a Text Editor. We recommend **Visual Studio Code** (*file size - 61.1MB*).
    - If you write a code in Jupyter Notebook and wish to reuse it written, you will have to mov the code in a **.py** file and then just use the file in **VS Code**.
    - When writing a Python program with IDLE you miss out on a lot of things, like auto-complete suggestions, auto-indentation, inline documentation for module/function, aesthetics, etc.
    - For efficacy reasons, we will use a good text editor like **VS Code**.
    - Set up **VS Code** for **Python**.

    - Install the Python Extension in VS Code: link. This will setup all the tools necessary for easy editing, running and testing Python programs.
    - Select a Python Interpreter: link. This will tell the Python IntelliSense which interpreter to use while you write, run and test your Python programs. It will also detect the virtual environments created by Miniconda and let you choose and switc between them. So, you will select the environment `NB_<Team-ID>` in this case.
  - Other than auto-completion, inline documentation, etc. **VS Code** also lints your code, helps you debug your code very easily (so you can avoid print statements).

---

5. Testing the Setup

In order to verify and test the successful installation of above software and libraries, follow the step provided in Test_Setup document.

References

1. How, What and Why you should use Anaconda Miniconda?
2. Anaconda vs Miniconda - StackOverflow answer
3. Conda Cheat Sheet
4. JupyterLab Documentation
5. Jupyter Notebook Documentation
6. Coppelia Robotics
7. CoppeliaSim User Manual
8. Getting Started with Python in VS Code
