Getting Started with Python

The purpose of this tutorial is to help you get started installing and using Python. Although there are many programming languages available for implementing data mining algorithms, we choose Python for several reasons. First, it has emerged as one of the most popular programming languages for data science in recent years. Second, it has extensive libraries and ecosystems available to support the collection, preprocessing, mining, and visualization of data, which is beneficial for rapid prototyping of code. Third, it is open source, allowing anyone with access to a computer to download and execute the example code given in this tutorial without the need to install expensive software.

In this tutorial, we first provide instructions on how to install and setup Python and Jupyter notebook on your machine. We then illustrate a few simple commands to help you get started executing Python code from a Web browser via the Jupyter notebook.

1. Installing and Setting Up Python

Although there are many Python installations available, one of the easiest way to install Python on your machine is by using a pre-packaged distribution such as Anaconda from Continuum Analytics (http://www.anaconda.com (http://www.anaconda.com)). In this section, we describe the installation procedure for the Anaconda distribution of Python packages.

1.1. Installing Python from Scratch

First, download the Python base installation package from https://www.anaconda.com/download (https://www.anaconda.com/download). All the examples given in this tutorial were tested on Python version 3.6 installation, though many of the examples here will still work with version 2.7. After installing the base package from the link provided, launch the Anaconda command prompt and the following interface will be displayed:
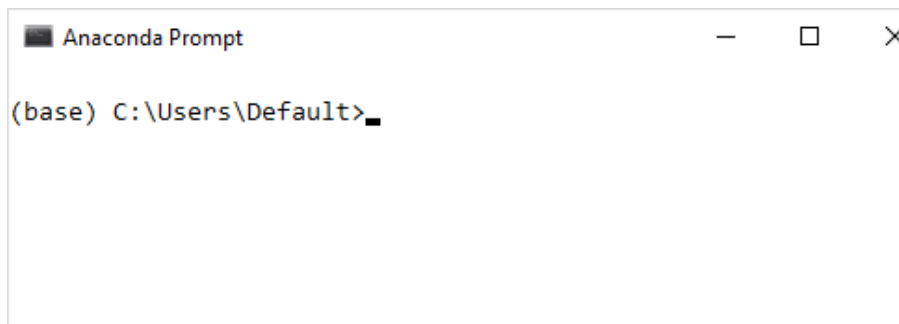
1.2. Upgrade from an Older Version of Python

If you already have an older version of Python (e.g., version 2.7) installed on your machine and would like to upgrade it to a newer version, you can create an environment for installing the new version without overwriting the older one. To do this, you must first launch the the Anaconda command prompt and type the following:

```
C:\Users\Default> conda create -n py36 python=3.6 anaconda
```

where the -n py36 option specifies the name of the environment, python=3.6 is the name and version of the package to be installed, and anaconda is the name of the meta-package that contains all the Python packages of the Anaconda distribution. After creating the environment, you need to activate the new environment from the command prompt before launching Python 3.6 and deactivate it when you're done. The commands for activating and deactivating the new environment are shown below.

```
C:\Users\Default> activate py36       # to launch the new environment
C:\Users\Default> python --version  # check to make sure it is the new version of P
ython
    ...
C:\Users\Default> deactivate         # to exit from the environment
C:\Users\Default> python --version  # this will display the older version of Python
```

Note that any text that follows the '#' character are simply comments to help explain the meaning of each command.

1.3. Updating and Installing Additional Python Library Packages

After installing the base Python distribution packages, you should check to see if any of the installed packages are outdated. If so, you should update them. To search for outdated package, type the following on Anaconda command prompt:

```
C:\Users\Default> conda search --outdated
C:\Users\Default> conda update --all
```

You can also type the conda update command without searching for outdated packages. To check all the packages installed on your machine, type the following command:

```
C:\Users\Default> conda list
```

You can also use 'conda list' to check whether a specific package has been installed. For example, to check whether a specific package name is installed on your machine, you can type the following:

```
C:\Users\Default> conda list package_name
```

To install a specific package, type the following:

```
C:\Users\Default> conda install package_name
```

where package_name is the name of the Python library package you would like to install. Some Python packages may not be part of Anaconda's distribution. In that case, you can still install the package using the 'pip' command:

```
C:\Users\Default> pip install package_name
```

2. How to Write and Execute Your Python Program

This section provides some examples on how to write and execute your Python program. Python is generally considered an interpreted language, which means its program cannot be directly executed by your machine. Instead, it requires another program, called an interpreter, to read and execute the Python program. To invoke the interpreter, type 'python' on the Anaconda command prompt:

```
C:\Users\Default> python
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:25:24) [MSC v.1900
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

The interpreter is now ready to accept your Python commands. For example, to display the message 'Welcome to Data Mining!' on the screen, type:

```
>>> print('Welcome to Data Mining!')
Welcome to Data Mining!
```

To exit the interpreter, type quit() at the interpreter's command prompt. You can also write your Python program using a text editor and save it in a file with a '.py' extension. The following example shows a simple Python program for displaying the message 'Welcome to Data Mining' The program is stored in a file named dm.py. To execute the program using a Python interpreter on a Windows machine, type the following on Anaconda command prompt:

```
C:\Users\Default> type dm.py
#!C:\Users\default\Anaconda2\python
print("Welcome to Data Mining!")

C:\Users\Default> python dm.py
Welcome to Data Mining!
```
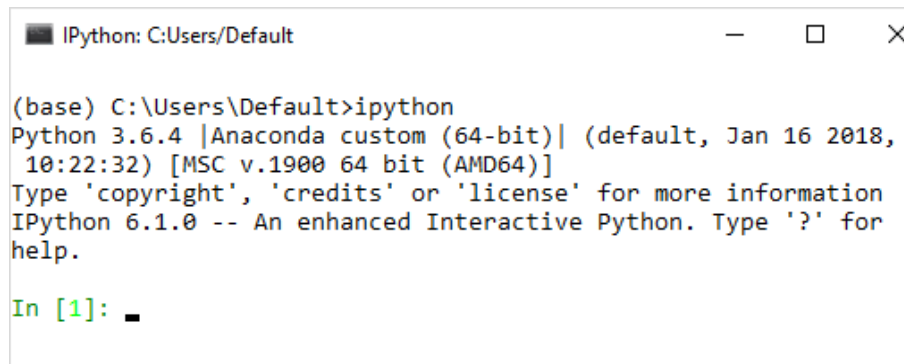
where you need to replace C:\Users\default\Anaconda2\python on the first line of the code with the path to Python's binary executable file on your machine. To run the program on a Linux machine:

```
\users\Default> cat dm.py
#!\usr\bin\python
print("Welcome to Data Mining!")

\users\Default> python dm.py
Welcome to Data Mining!
```

2.1. IPython

IPython provides an interactive shell that allows users to execute their Python code in a user-friendly environment. To launch IPython, type the following on the Anaconda command prompt.

In addition to the different input prompt displayed compared to the standard Python interpreter, the IPython shell is more powerful as it allows you to display graphics, performs tab-completion, object introspection, recall the history of input commands and previous results, etc. For more information, go to https://ipython.org/ipython-doc/3/interactive/tutorial.html (https://ipython.org/ipython-doc/3/interactive/tutorial.html).

```
In [1]: x = [1,2,3]

In [2]: x
Out[2]: [1 2 3]

In [3]: x.<TAB>
x.append x.count x.insert x.reverse
x.clear x.extend x.pop x.pop
x.copy x.index x.remove

In [3]: x.append(4)

In [4]: y = Out[2]

In [5]: y
Out[5]: [1 2 3 4]

In [6]: y?
Type: list
String form: [1, 2, 3, 4]
Length: 4
Docstring:
list() -> new empty list
list(iterable) -> new list initialized from iterable's items
```

The following example shows how to inspect a function along with its source code using ? and ??:

```
In [7]: def negate(x):
   ...: """
   ...: Negate an input list
   ...: """
   ...: return [i*-1 for i in x]
   ...:

In [8]: y = negate(x)

In [9]: y
Out[9]: [-1, -2, -3, -4]

In [10]: negate?
Signature: negate(x)
Docstring: Negate an input list
File: c:\users\default\<ipython-input-7-ca0db2e7ebb3>
Type: function
In [11]: negate??
Signature: negate(x)
Source:
def negate(x)
    """
    Negate an input list
    """
    return [i*-1 for i in x]
File: c:\users\default\<ipython-input-7-ca0db2e7ebb3>
Type: function
```

IPython also has many predefined commands known as magic functions. These commands are prefixed by the symbol %. One of the useful magic commands is to determine the execution time of a Python statement.

```
In [12]: import numpy as np

In [13]: x = np.random.randn(1000000)

In [14]: %time y = np.sort(x)
Wall time: 78.1 ms

In [15]: %timeit y = np.sort(x)
10 loops, best of 3: 67 ms per loop
```

In the above example, the %time magic function will report the wall clock time for executing the statement. Since the reported time may vary from one execution to another, it will be useful to repeat the statement execution multiple times and report their average execution time. This can be done using the %timeit magic function. In the above example, %timeit function executes the statement 10 times in a loop, repeat the loop 3 times, and report

the best average execution time per statement in a loop among the 3 loop repetitions. You can modify the number of times the statement is executed in a loop using the -N option and number of times to repeat the loop using the -r option.

2.2. Jupyter Notebook

YOu can also use a Web-based user-friendly environment called Jupyter notebook to write and execute your Python program. To launch the notebook, type the following command on the Anaconda prompt:

```
C:\Users\Default > jupyter notebook
```

The preceding command will open the Web browser and display a Web page that shows the directory structure of the default folder set for Jupyter Notebook:
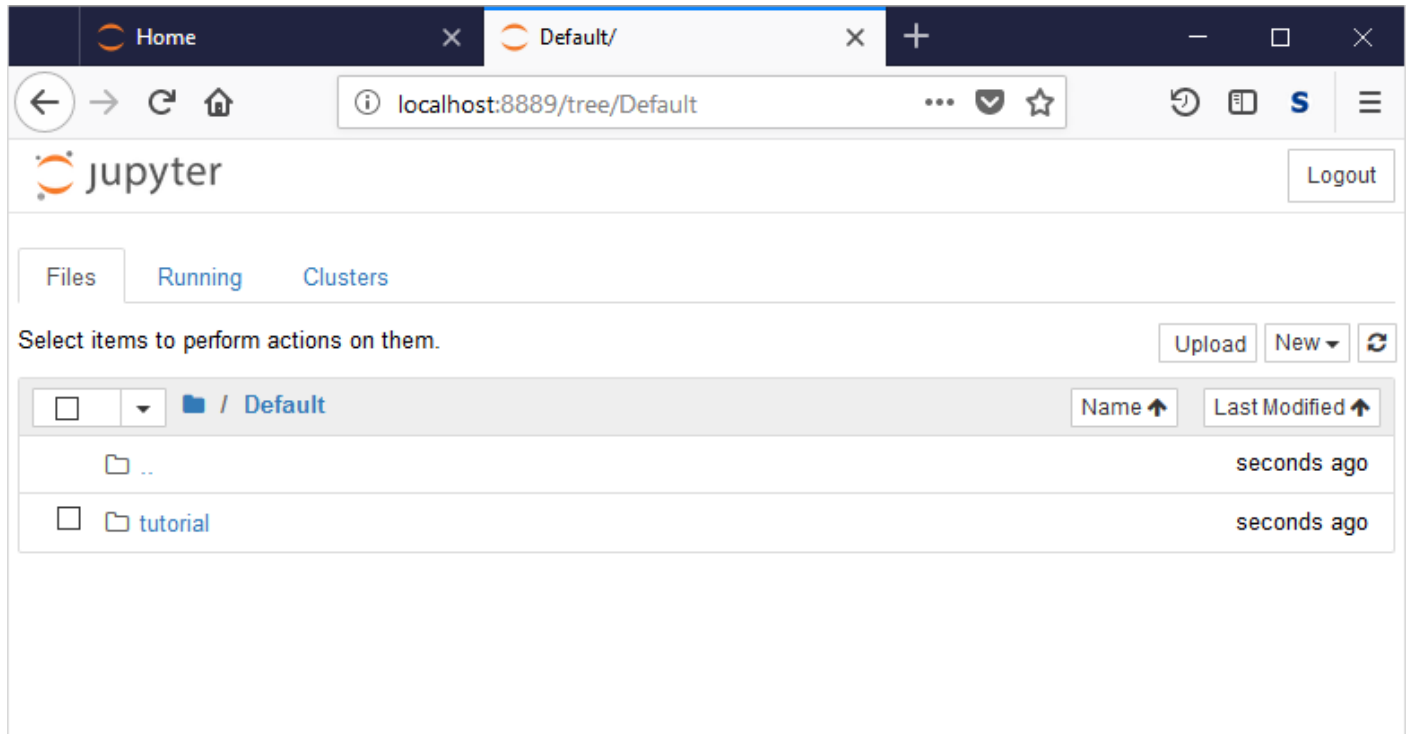

You can traverse the directory structure to locate the folder that contains the Python code you would like to execute, the data files you would like to load, or the working directory for saving your Python notebook.

To create a new notebook, click on the New button on the top right hand corner of the web page and select Python 3 notebook.
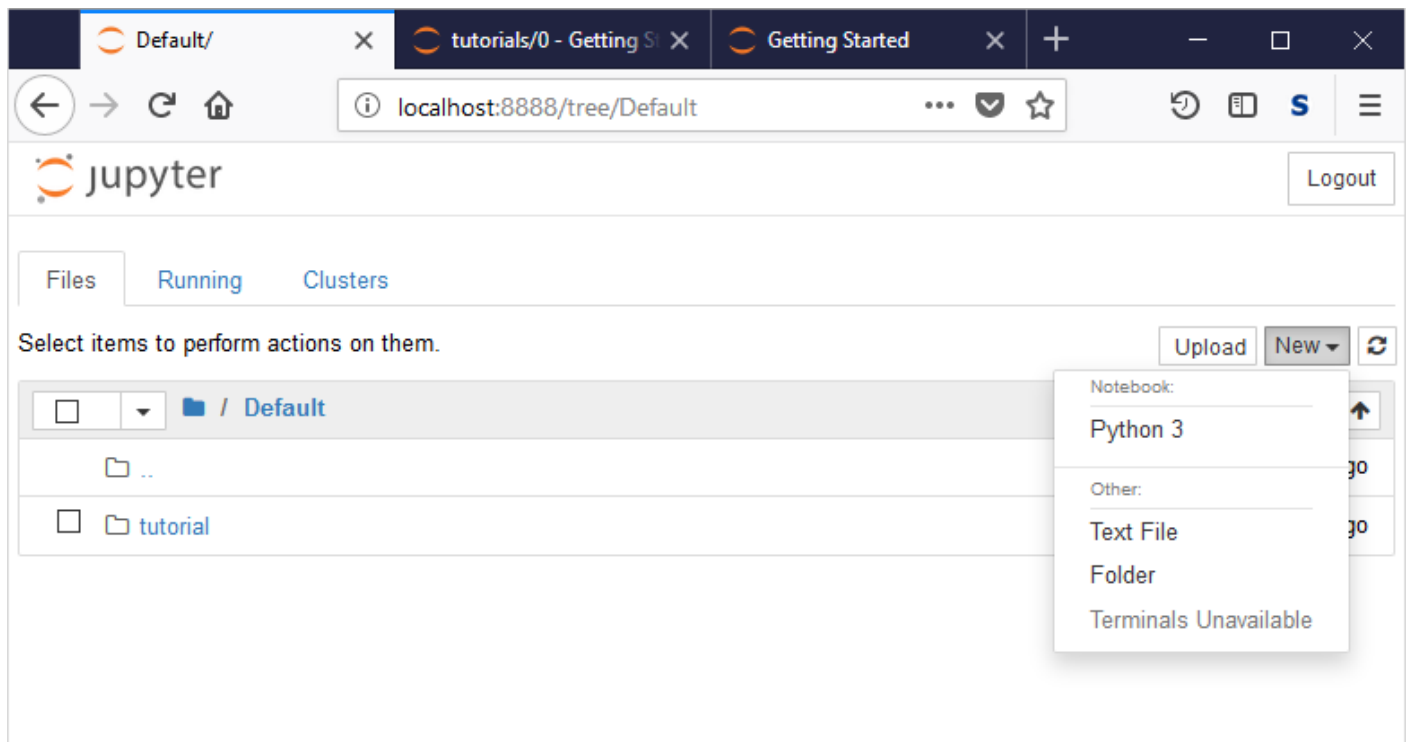
The following web page will be rendered and the notebook is ready to accept your Python commands.
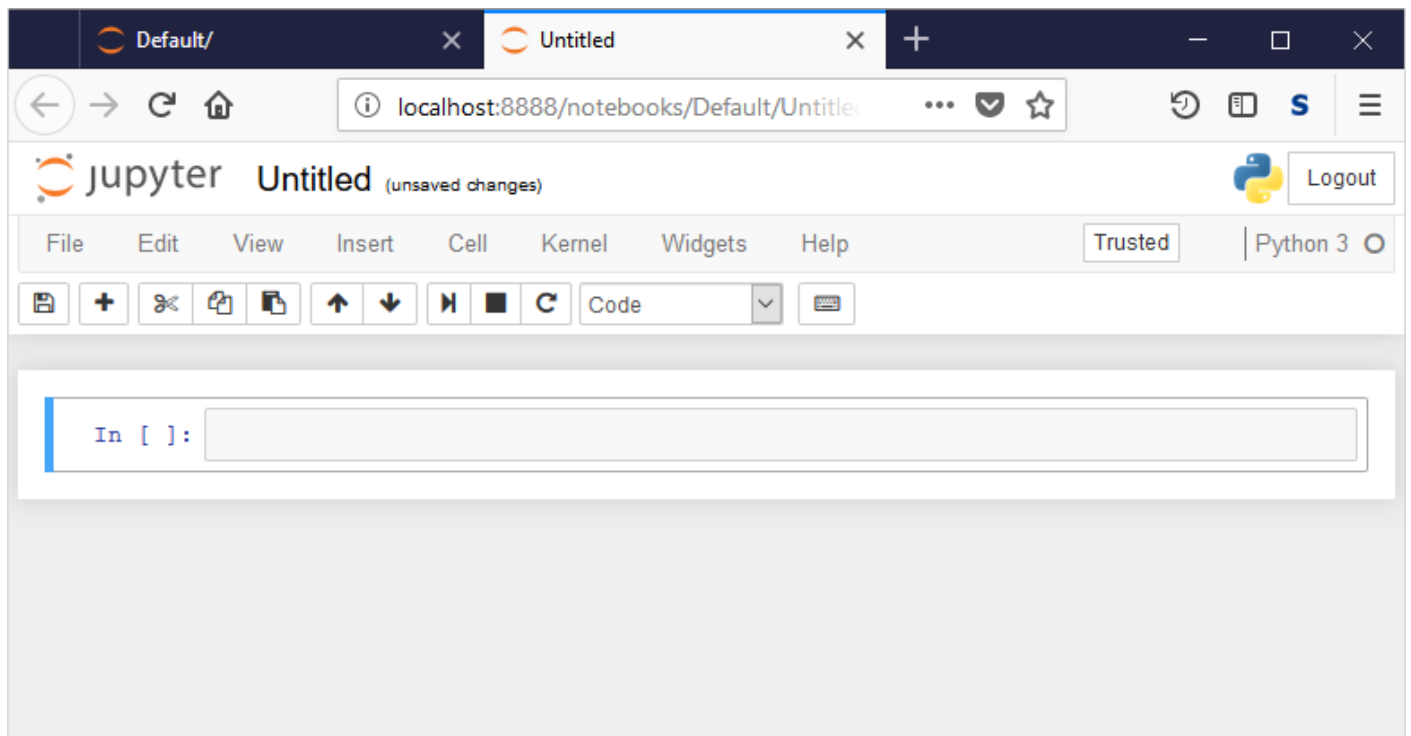
The Python statements are entered in each cell. To execute the Python statements within each cell, press both the SHIFT and ENTER keys simultaneously. The result will be displayed right below the cell, as shown in the diagram below.
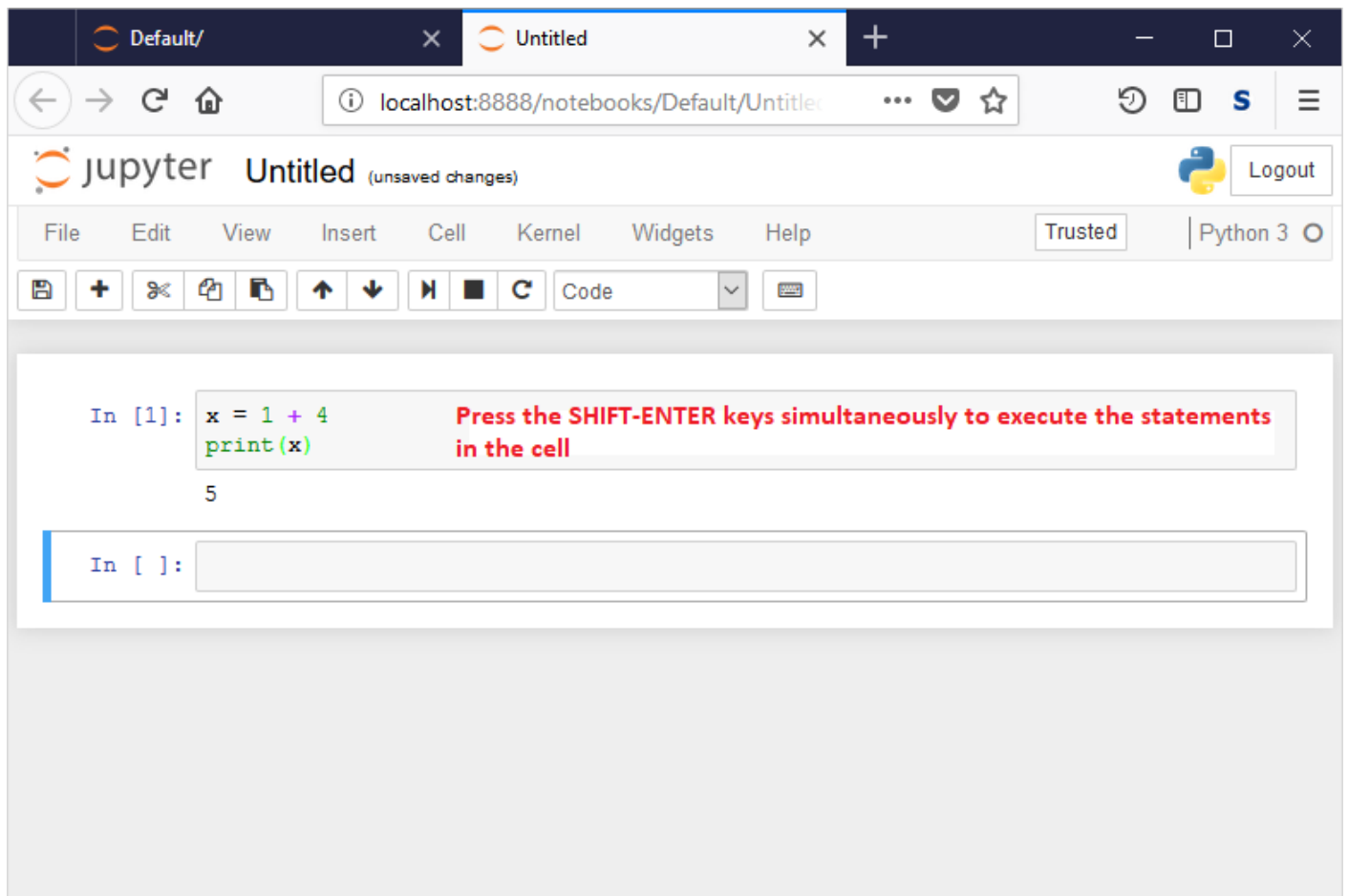
By default, the new notebook will be stored in a file named Untitled.ipynb. You can rename the file by clicking on File and Rename menu option at the top, as shown in the diagram below.
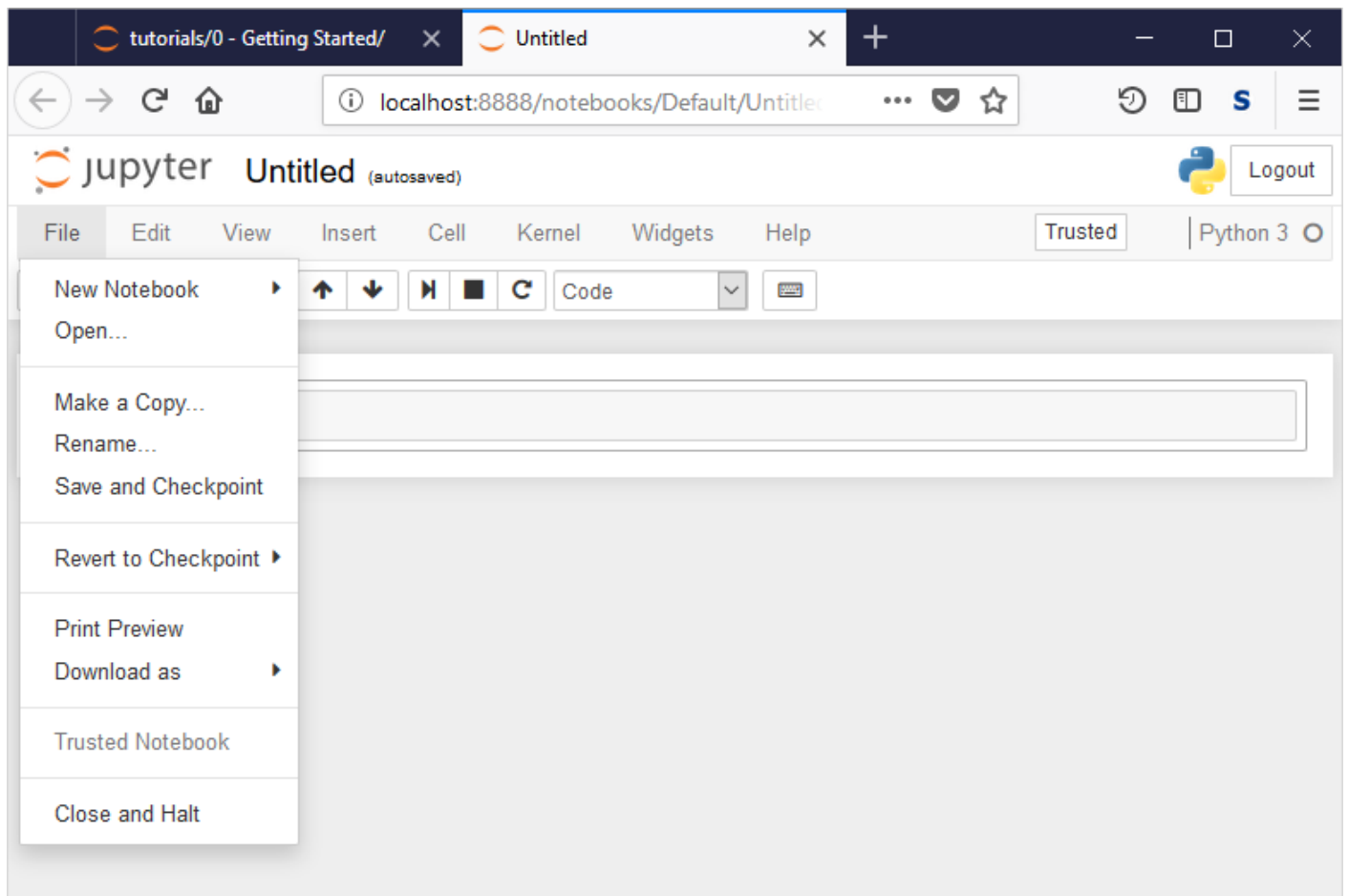
You can save the notebook by clicking on the 'File' and 'Save and Checkpoint' menu options. The notebook will

3. Summary

This tutorial provides instructions on how to install and setup Python on your machine. It also describes the different ways to execute your Python code, e.g., using the IPython interpreter or a Jupyter notebook. You should familiarize yourself with the various functionalities provided by the Jupyter Notebook as it will be used it in the rest of our tutorials.