



Python for Physics

Lab -1

What is Python

Python is a widely used general-purpose high level programming language, designed by Guido van Rossum in 1991.

Why Python

Python has several advantages over other programming languages. They are:

- i. Code readability
- ii. Easy syntax
- iii. English like keywords
- iv. Concise programs
- v. Automatic Memory management
- vi. Free and open source software



Python comparing with C

C

```
#include "stdio.h"
int main() {
    printf("Hello\n");
}
```

Java

```
public class Hi {
    public static void main (String [] args) {
        System.out.println("Hello");
    }
}
```

python

```
print("hello")
```

Notice: no ;



NumPy

localhost:8888/notebooks/Python%20for%20Applied%20Physics%20.ipynb#

jupyter Python for Applied Physics Last Checkpoint: Last Tuesday at 3:14 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Python for Applied Physics

Mechanics

Grivational Force

This program calculates and displays the Gravitational Force between two masses:

$$F = \frac{Gm_1m_2}{r^2}$$

```
In [1]: # define the input
mass_1 = float( input('Enter the value of mass 1 (in kilogram) '))
mass_2 = float( input('Enter the value of mass 1 (in kilogram) '))
distance = float( input('Enter the value of distance '))
G = 6.67*10**-11      # Gravitational constant in the units of Nm^2/kg^2:

# Calculation
# computes the value of Gravitational Force
Grv_force = (G * mass_1*mass_2)/(distance**2)
print ('Force between the masses is = ',Grv_force,' Newtons.')
```

Introduction to Jupiter Notebook



The Jupyter Notebook is an open-source web application that allows us to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

A single document that combines explanations with executable code and its output — an ideal way to provide:

- ▶ reproducible research results
- ▶ documentation of processes
- ▶ instructions
- ▶ tutorials and training materials of all shapes and sizes

Installation

The easiest way for a beginner to get started with Jupyter Notebook is by installing [Anaconda](#).

Link for installing Anaconda is:

<https://www.anaconda.com/products/individual>

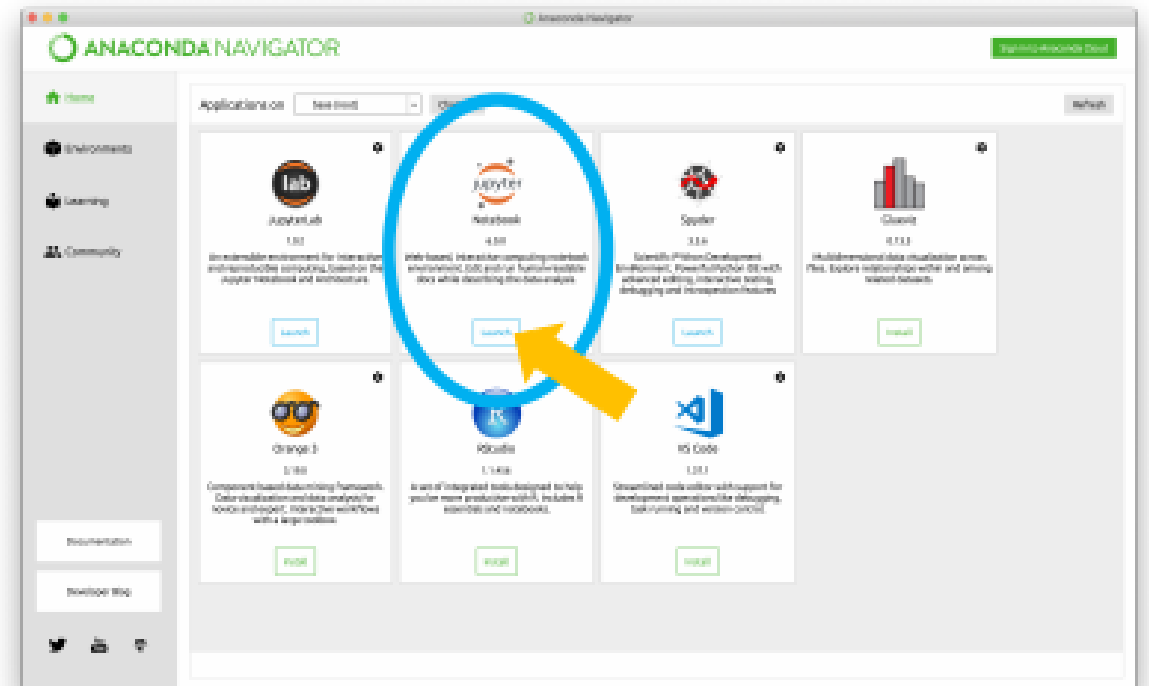
Anaconda is the most widely used Python distribution for data science and comes pre-loaded with all the most popular libraries and tools.

Some of the biggest Python libraries included in Anaconda include [NumPy](#), [pandas](#), and [Matplotlib](#), though the [full 1000+ list](#) is exhaustive.

How to Launch Jupyter Notebook

There are 3 ways to launch Jupyter Notebook:

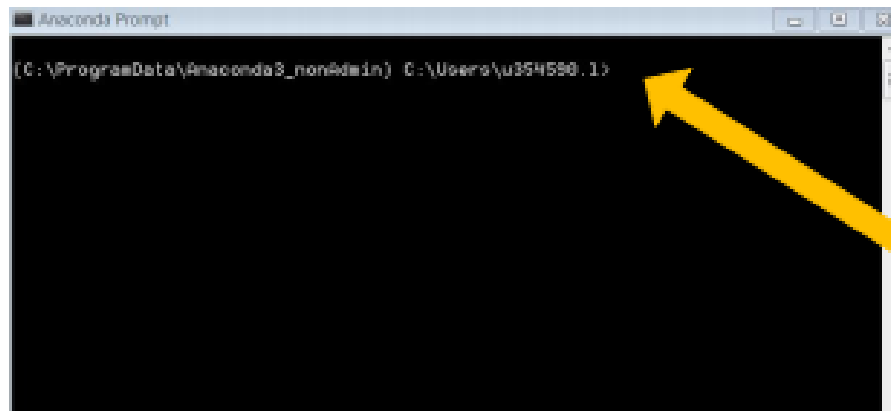
- 1) Using Anaconda Navigator  **ANACONDA NAVIGATOR**
 - a) Open the application called Anaconda Navigator (this may take a couple of minutes)
 - b) Click on “Launch” in the Jupyter Notebook box



How to Launch Jupyter Notebook

2) Using Anaconda Prompt

- a) Open the application called Anaconda Prompt
- b) Type "jupyter notebook" (without quotes) and hit the return key

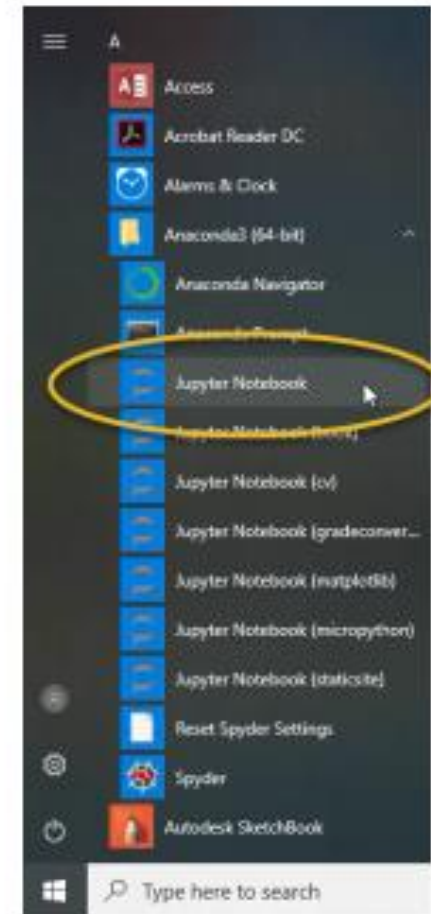


Type
"jupyter notebook"
here

Note: your Anaconda Prompt window will show a different pathname than in this image, but it will look similar!

How to Launch Jupyter Notebook

3) Clicking on the Jupyter Notebook App in the Start Menu (I just learned about this method, and it will probably be the fastest!)



How to Launch Jupyter Notebook

You will know that Jupyter Notebook opened correctly if you see a page similar to this one open in your browser!

A screenshot of the Jupyter Notebook web interface. The top bar shows the 'jupyter' logo and 'Quit' and 'Logout' buttons. Below the logo are tabs for 'Files', 'Running', and 'Clusters'. The 'Files' tab is active, showing a file browser. At the top of the file browser, there's a prompt 'Select items to perform actions on them.' and buttons for 'Upload', 'New', and a refresh icon. A table lists files and folders with columns for 'Name', 'Last Modified', and 'File size'. The table includes folders like '3D Objects', 'Contacts', 'Desktop', 'Documents', 'Downloads', 'Evernote', 'Favorites', 'Gpredict', 'Links', and 'movielens'. A blue arrow points from the text on the left to the file browser interface.

jupyter

Quit Logout

Files Running Clusters

Select items to perform actions on them.

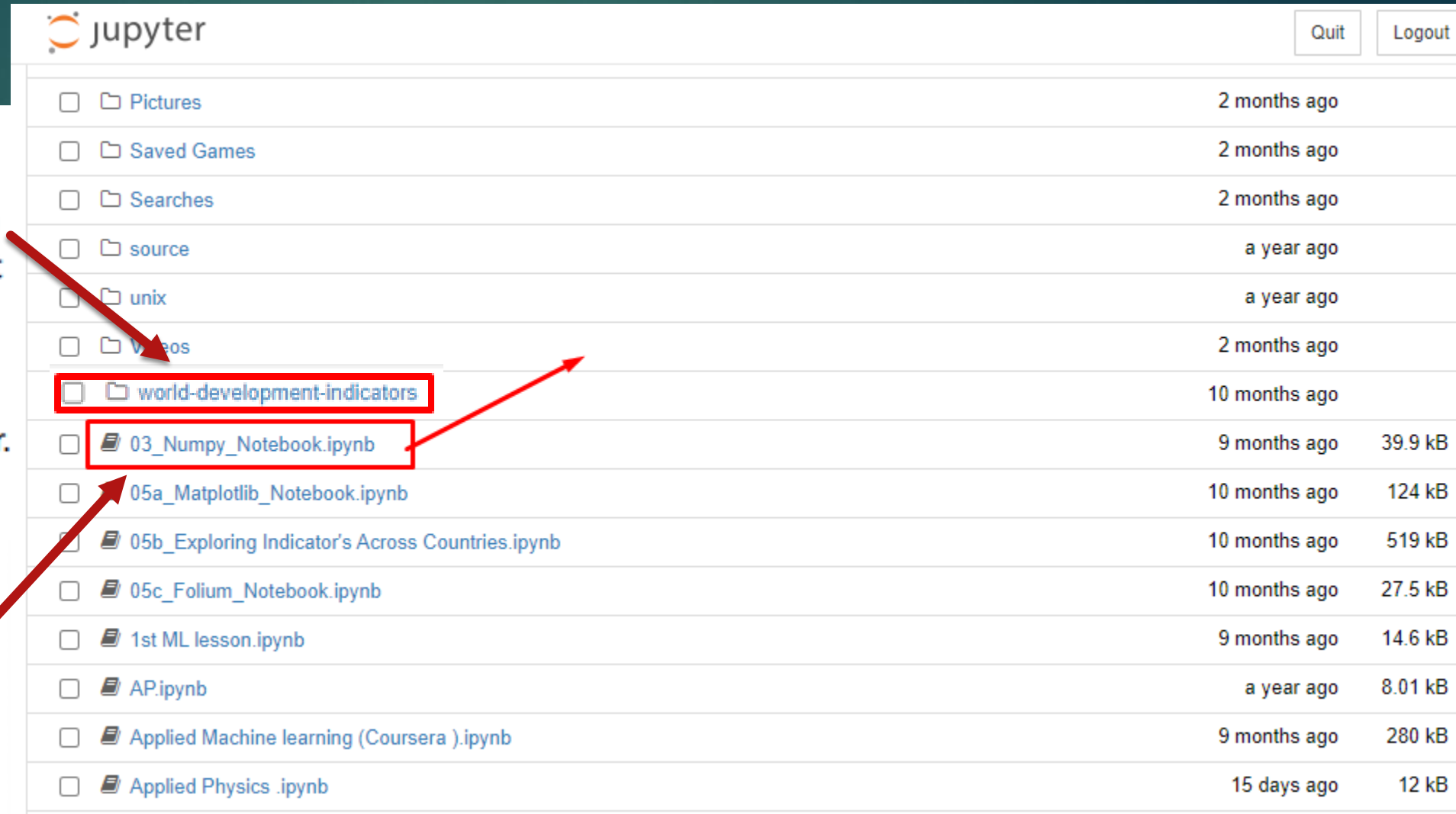
Upload New ↕ ↻

<input type="checkbox"/> 0 ▾	📁 /	Name ▾	Last Modified	File size
<input type="checkbox"/>	📁 3D Objects		2 months ago	
<input type="checkbox"/>	📁 Contacts		2 months ago	
<input type="checkbox"/>	📁 Desktop		13 days ago	
<input type="checkbox"/>	📁 Documents		12 days ago	
<input type="checkbox"/>	📁 Downloads		19 minutes ago	
<input type="checkbox"/>	📁 Evernote		a year ago	
<input type="checkbox"/>	📁 Favorites		2 months ago	
<input type="checkbox"/>	📁 Gpredict		a year ago	
<input type="checkbox"/>	📁 Links		2 months ago	
<input type="checkbox"/>	📁 movielens		10 months ago	

How to open a Notebook file

Navigate through your folders until you get to the directory you want to save your scripts in. You can navigate through by clicking on the name of the Folder.

- **Open a previously saved Notebook file** by clicking on the name of the file
- The extension for a Jupyter Notebook file is “.ipynb”, which is short for “interactive python notebook”



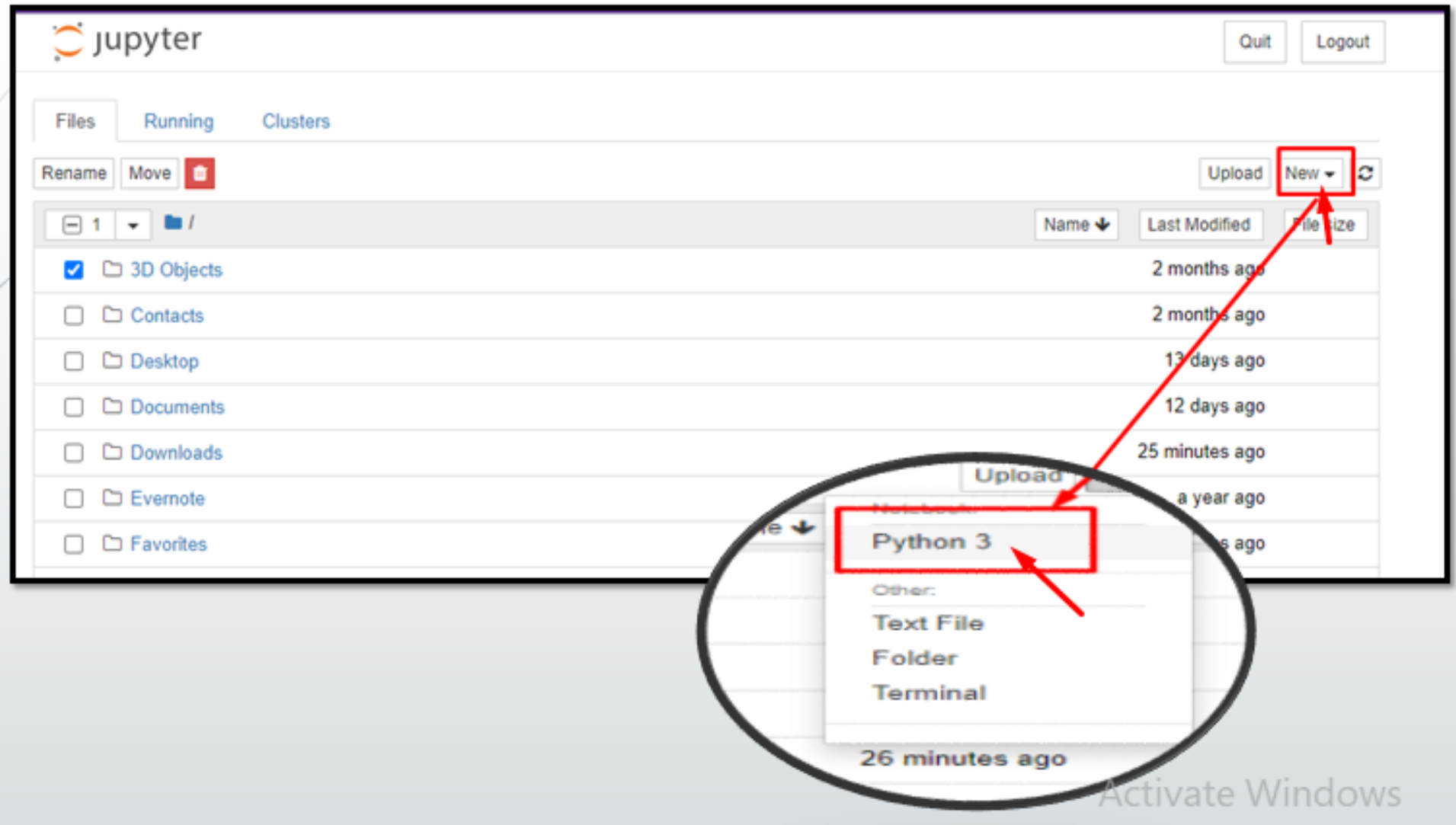
jupyter

Quit Logout

<input type="checkbox"/>	Folder	Pictures	2 months ago	
<input type="checkbox"/>	Folder	Saved Games	2 months ago	
<input type="checkbox"/>	Folder	Searches	2 months ago	
<input type="checkbox"/>	Folder	source	a year ago	
<input type="checkbox"/>	Folder	unix	a year ago	
<input type="checkbox"/>	Folder	Videos	2 months ago	
<input type="checkbox"/>	Folder	world-development-indicators	10 months ago	
<input type="checkbox"/>	File	03_Numpy_Notebook.ipynb	9 months ago	39.9 kB
<input type="checkbox"/>	File	05a_Matplotlib_Notebook.ipynb	10 months ago	124 kB
<input type="checkbox"/>	File	05b_Exploring Indicator's Across Countries.ipynb	10 months ago	519 kB
<input type="checkbox"/>	File	05c_Folium_Notebook.ipynb	10 months ago	27.5 kB
<input type="checkbox"/>	File	1st ML lesson.ipynb	9 months ago	14.6 kB
<input type="checkbox"/>	File	AP.ipynb	a year ago	8.01 kB
<input type="checkbox"/>	File	Applied Machine learning (Coursera).ipynb	9 months ago	280 kB
<input type="checkbox"/>	File	Applied Physics .ipynb	15 days ago	12 kB

How to open a Notebook file

Open a new Notebook file by clicking on the “New” menu on the upper right



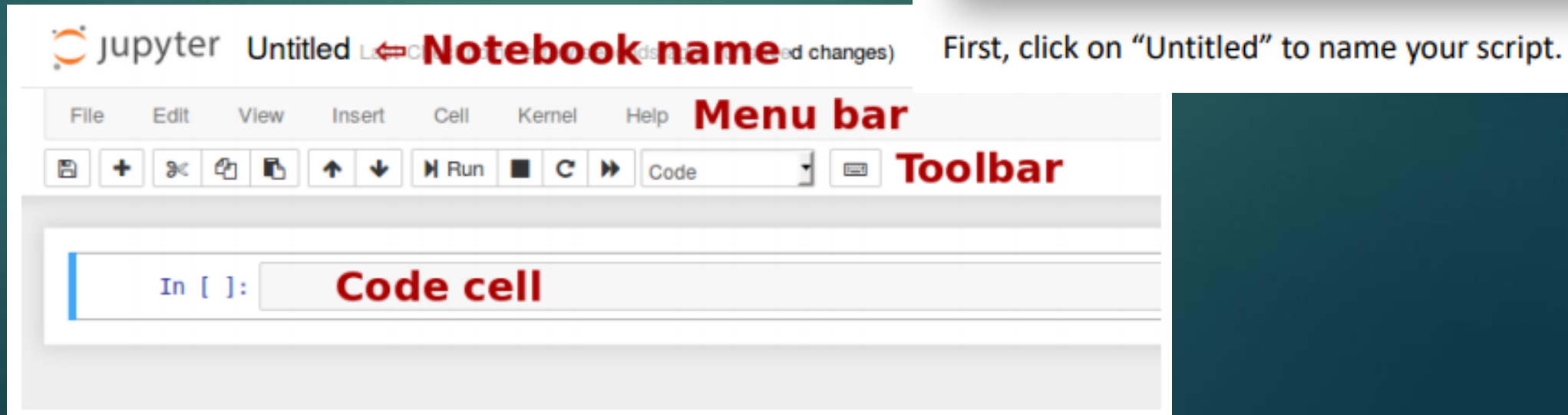
How to start writing a Jupyter Notebook

Notebook name: The name displayed at the top of the page, next to the Jupyter logo, reflects the name of the .ipynb file. Clicking on the notebook name brings up a dialog which allows you to rename it. Thus, renaming a notebook from “Untitled0” to “My first notebook” in the browser, renames the Untitled0.ipynb file to My first notebook.ipynb.

Menu bar: The menu bar presents different options that may be used to manipulate the way the notebook functions.

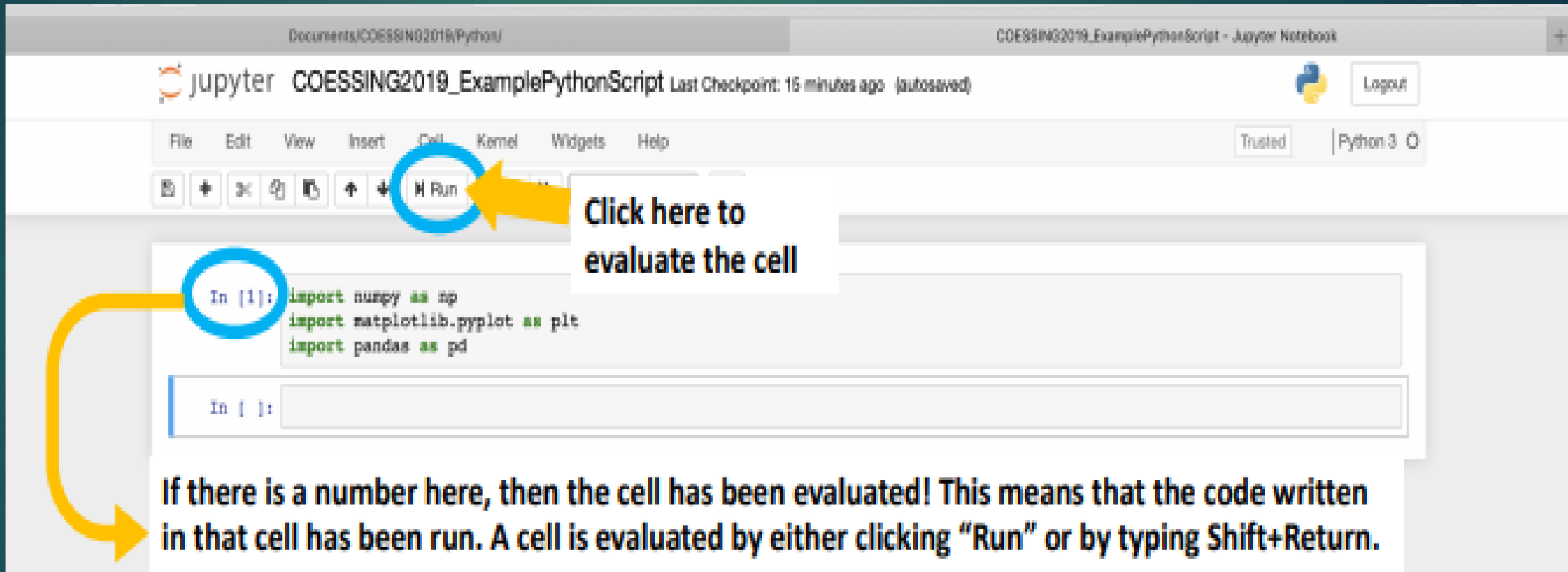
Toolbar: The tool bar gives a quick way of performing the most-used operations within the notebook, by clicking on an icon.

Code cell: the default type of cell; read on for an explanation of cells



How to start writing a Jupyter Notebook

It's good practice to start your script by importing libraries you will need. Below are three libraries that are often used, but you may need different ones.



The screenshot shows a Jupyter Notebook window titled "COESSING2019_ExamplePythonScript". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for creating new cells, saving, and running. A yellow arrow points to the "Run" button in the toolbar, with a text box stating "Click here to evaluate the cell". Another yellow arrow points to the "In [1]:" prompt in the first code cell, which contains the following code:

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Below this cell is an empty code cell with the prompt "In []:". A yellow arrow points from the "In [1]:" prompt to a text box at the bottom of the notebook that reads: "If there is a number here, then the cell has been evaluated! This means that the code written in that cell has been run. A cell is evaluated by either clicking 'Run' or by typing Shift+Return."

Creating a Variable

- ▶ To create a variable in Python, all you need to do is specify the variable name, and then assign a value to it.
- ▶ **<variable name> = <value>**
- ▶ Python uses = to assign values to variables. There's no need to declare a variable in advance (or to assign a data type to it), assigning a value to a variable itself declares and initializes the variable with that value. There's no way to declare a variable without assigning it an initial value.

Examples of different variables of different data types

```
a = 4
b = 3.5
c = 'Physics'
list = [1,2,3,4]
print (a, ' ', b , ' ', c , ' ', list)
```

```
4 , 3.5 , Physics , [1, 2, 3, 4]
```

```
print (type(a), type(b), type(c), type(list))
```

```
<class 'int'> <class 'float'> <class 'str'> <class 'list'>
```

```
print ("Hello World")
```

```
print(a)
```

```
print(" The course name: ", c)
```

```
Hello World
```

```
4
```

```
The course name: Physics
```

Point to remember for variable assignment

- ▶ Variable assignment works from left to right. So the following will give you a syntax error.

```
0=x
```

```
File "<ipython-input-83-e50bc5b3cdd5>", line 1
```

```
0=x
```

```
^
```

```
SyntaxError: can't assign to literal
```

Rules for variable naming

1. Variables names must start with a letter or an underscore

```
x = True # valid
_y = True # valid
9x = False # starts with numeral
```

```
File "<ipython-input-85-798efde345de>", line 3
  9x = False # starts with numeral
   ^
SyntaxError: invalid syntax
```

```
$y = False # starts with symbol
```

```
File "<ipython-input-86-50101988d602>", line 1
  $y = False # starts with symbol
   ^
SyntaxError: invalid syntax
```

2. The remainder of your variable name may consist of letters, numbers and underscores.

3. Names are case sensitive.

```
x = 9  
y = X*5
```

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-90-2c3b2d56c6c5> in <module>()  
      1 x = 9  
----> 2 y = X*5  
  
NameError: name 'X' is not defined
```

4. You cannot use python's keywords as a valid variable name. You can see the list of keyword by:

```
import keyword  
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else',  
'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise',  
'return', 'try', 'while', 'with', 'yield']
```

Python as calculator(1/2)

```
In [1]: 2+5
```

```
Out[1]: 7
```

```
In [2]: 5-3
```

```
Out[2]: 2
```

```
In [3]: 7*6
```

```
Out[3]: 42
```

```
In [4]: 90/4
```

```
Out[4]: 22.5
```

Python as Calculator(2/2)

```
In [1]: 5//2    # rounds down the answer to the nearest whole number
```

```
Out[1]: 2
```

```
In [2]: 9%2     # Modulus : gives the remainder when 9 is divided by 2
```

```
Out[2]: 1
```

```
In [3]: 10%2    # Modulus: gives the remainder when 10 is divided by 2
```

```
Out[3]: 0
```

```
In [6]: 6**2    # Exponent: 6 to the power of 2
```

```
Out[6]: 36
```