

# MSAS Tutorial Sequence -Module Two-

---

WRITTEN BY ADVAY MUCHOOR

A solid yellow horizontal bar at the bottom of the page.

**THANKS  
FOR  
COMING  
BACK**



# Goals of Today

---

1. Understand Some Basics Of Python
2. Learn to navigate Jupyter Notebook

Python



# Background and Info

---

- Python is a popular programming language that was released in 1991
- Designed for readability and simplicity
- Has wide variety of uses and packages
- We will be using Python for our data manipulation
- Python vs. C++
  - Similar syntaxes
  - No typecasting in Python
  - No semicolons in Python
  - Uses white space to define scope (not brackets)

# Variables in Python

---

- Variables are containers for storing data values.
- Unlike other programming languages, Python has no command for declaring a variable.
  - You don't have to typecast your variables at declaration
  - A variable is created the moment you first assign a value to it.
- Initialize a variable using the following format: name = value
- Examples:
  - `x = 5` (x is now an integer)
  - `y = "Advay"` (y is now a string)
  - `x = "Michigan"` (x is now a string – the types can change after declaration)
  - `average_points = 23.4` (average\_points is now a "float"/decimal)

# Notes about variables in Python

---

- Integers and Decimals can be positive or negative.
- Strings can be surrounded by single or double quotes (but not a mixture)
- Any variable can be outputted using the print command
  - `x = 5`
  - `print(x)` This will output 5
- You can use other variables to create new variables
  - `x = 5`
  - `y = 10`
  - `z = x + y`
  - `print(z)` This will output 15

# Variable types

---

- You can get the data type of any object by using the `type()` function
- Useful when we get into more advanced variable types
- Example
  - `x = 5`
  - `print(type(x))` <class 'int'>
- The type of a variable can be changed really simply through casting
  - `x = 5`
  - `y = str(x)`
  - `print(type(y))` <class 'str'>
  - Be careful, as casting in the opposite direction will cause an error unless the string only contains numeric characters



# Literals vs Variables

---

- A literal is the literal value of something
- Literals are useful when we don't want to store a piece of data in a variable
- Examples
  - `x = 3`
  - `y = x + 5`
  - `print(y)` 8
  - `print (3 + 5)` 8
  - `print("hello world")` hello world

# Operators

---

- Basic shared operator for integers and floats
  - Addition (+)
  - Subtraction (-)
  - Multiplication (\*)
  - Exponents (\*\*)
- Basic operators for strings
  - Addition (+)
  - `x = "Python is "`
  - `y = "cool"`
  - `print (x + y)` □ `"Python is cool"`
- CANNOT USE "+" OPERATOR ON OPERANDS WITH DIFFERENT TYPES
  - `print("Advay" + 5)` □ Error
  - `print("Advay" + str(5))` □ `Advay5`

Jupyter  
Notebooks



Select items to perform actions on them.

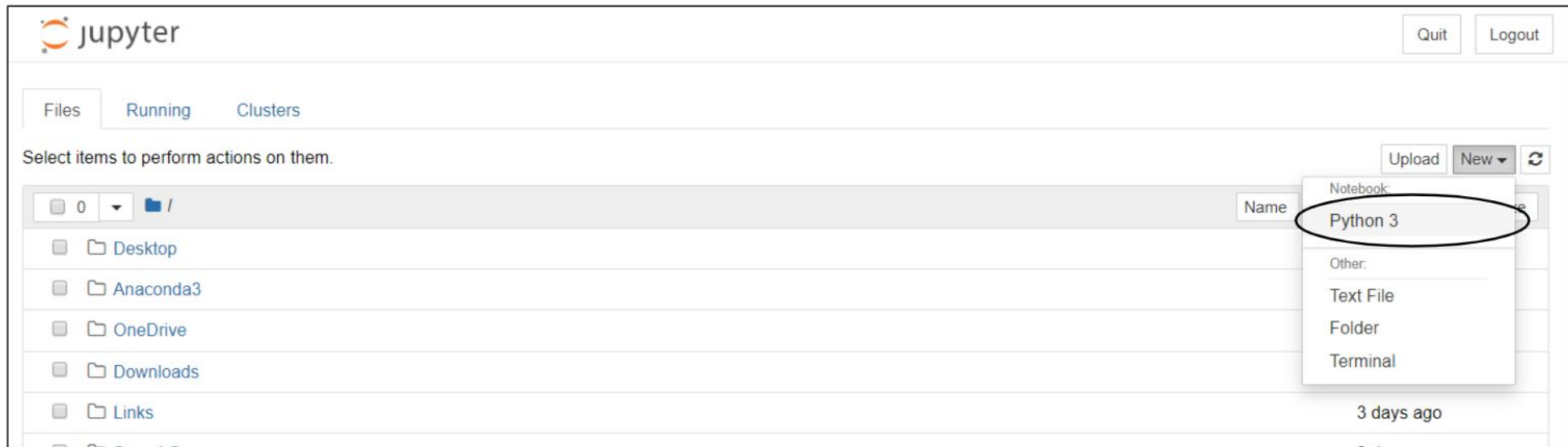
Upload

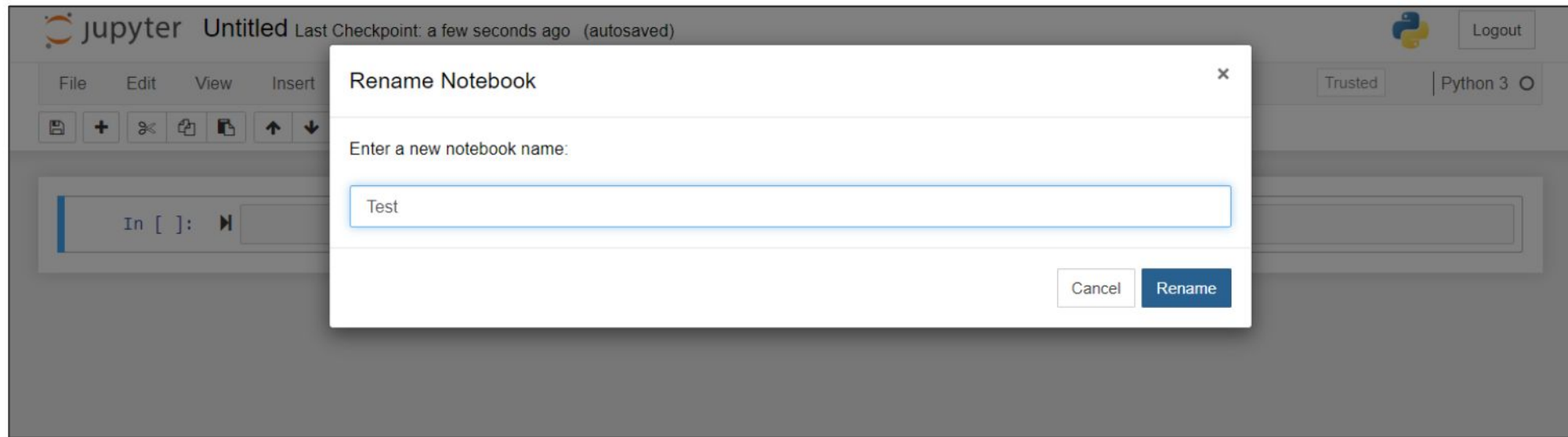
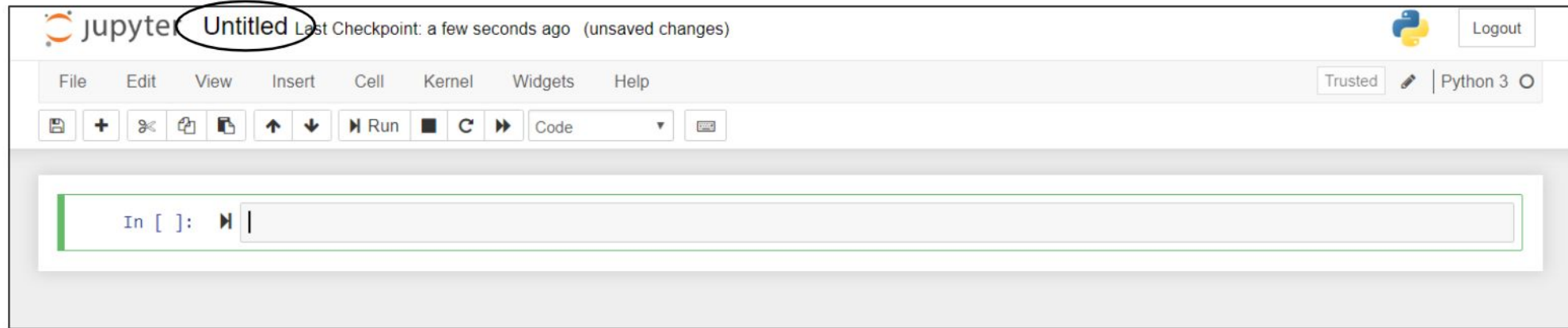
New ▾



0 ▾ /		Name ▾	Last Modified	File size
<input type="checkbox"/>	3D Objects		3 days ago	
<input type="checkbox"/>	Anaconda3		7 hours ago	
<input type="checkbox"/>	CMakeBuilds		9 months ago	
<input type="checkbox"/>	Contacts		3 days ago	
<input type="checkbox"/>	Desktop		a minute ago	
<input type="checkbox"/>	Documents		3 days ago	
<input type="checkbox"/>	Downloads		5 hours ago	
<input type="checkbox"/>	Favorites		3 days ago	
<input type="checkbox"/>	Links		3 days ago	
<input type="checkbox"/>	Music		3 days ago	
<input type="checkbox"/>	OneDrive		7 hours ago	
<input type="checkbox"/>	Pictures		3 days ago	
<input type="checkbox"/>	Saved Games		3 days ago	
<input type="checkbox"/>	Searches		3 days ago	
<input type="checkbox"/>	source		a year ago	
<input type="checkbox"/>	sqlite3		2 months ago	
<input type="checkbox"/>	Videos		3 days ago	
<input type="checkbox"/>	VirtualBox VMs		5 months ago	

# Dashboard





# Understanding The Kernel

---

- Each cell contains code in it
- The kernel is a collection of cells
  - The cells in the kernel are connected to each other
  - When the entire kernel is run, cells run top to bottom
- Cells can “talk” to each other
  - A cell can use variables that were defined in another cell
  - Variable MUST be defined before being used
- Cells can be run individually, or as a whole

# Cell

Blue cell ☐ Command mode (controlling the kernel)  
Green cell ☐ Active mode (writing code)

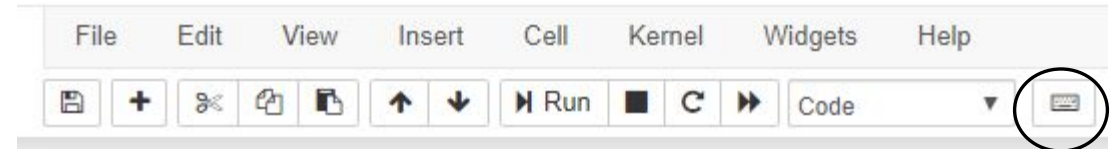
# Kernel



# Basic Commands

---

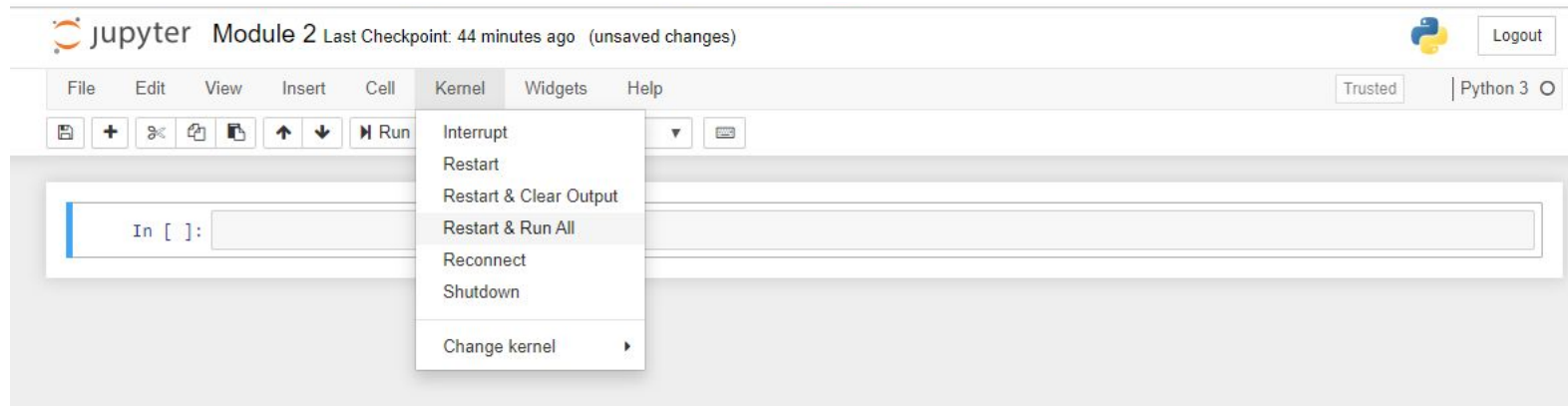
- Running a cell (Active and Command)
  - CTRL + ENTER / COMMAND + ENTER
- Inserting a cell Below (Command)
  - b key
- Inserting a cell Above (Command)
  - a key
- Deleting your current cell (Command)
  - d, d (d key twice)
- Save the kernel
  - CTRL + s / COMMAND + s



# Ordering of the cells

---

- Pay close attention to the order in which cells are run
- Every time you edit a cell, you should run it
- If you edit a cell and do not run it, the other cells will think that nothing has changed
- It is often useful to just restart the kernel and re-run everything (will restart cell numbers at 1)
  - “Restart and clear output” will delete all old variables (otherwise, old variables will never die)



In [1]: `x = 5`

In [2]: `y = x + 5`

In [3]: `print(y)`

10

In [4]: `z = x + y`  
`print (z)`

15

```
In [5]: x = 1000
```

```
In [2]: y = x + 5
```

```
In [3]: print(y)
```

10

```
In [4]: z = x + y  
print (z)
```

15

# Additional Information

---

- “Out [ ]” is the output for cell that was run
- Print statements do not generate “Out [ ]” blocks, but the behavior is still the same
- In jupyter notebooks, you don’t have to use print statements to get the output of a variable

```
In [1]: x = 5  
x
```

```
Out[1]: 5
```

```
In [2]: y = 10  
        print(y)  
10
```

```
In [1]: str1 = "Harbaugh"  
str1
```

```
Out[1]: 'Harbaugh'
```

```
In [2]: str3 = str1 + str2
```

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-2-0f2c49d30d2d> in <module>  
----> 1 str3 = str1 + str2  
  
NameError: name 'str2' is not defined
```

```
In [3]: str2 = " is cool"  
str2
```

```
Out[3]: ' is cool'
```

```
In [1]: str1 = "Harbaugh"  
str1
```

```
Out[1]: 'Harbaugh'
```

```
In [4]: str3 = str1 + str2  
str3
```

```
Out[4]: 'Harbaugh is cool'
```

```
In [3]: str2 = " is cool"  
str2
```

```
Out[3]: ' is cool'
```

# Take some time to play around with the environment

---

ASK QUESTIONS