LAB 1: INTRODUCTION

University of Washington

ECE 241

Winter 2022

Author: Jimin Kim (jk55@uw.edu)

Version: v1.5.0

OUTLINE

Part 1: Getting started with Python

- Setting up Python environment
- Jupyter Notebook
- Online Jupyter platform: Google Colab
- Configuring Jupyter cells
- Package maintenance

Part 2: Python's data types and variables

- Numbers, Booleans, Strings
- Lists

Part 3: Logical expressions and Operators

- Arithmetic operators
- Comparison operators
- Assignment operators
- Logical operators

Part 4: Lab Assignment

- Exercise 1 - 5

GETTING STARTED WITH PYTHON

OPTIONS FOR PYTHON ENVIRONMENT

Anaconda 3

Google Colaboratory

Offline

Online

SETTING UP PYTHON ENVIRONMENT (Anaconda 3)

What is Anaconda?

Anaconda is a distribution of Python and R for scientific computing

- Comes with >250 packages automatically installed
- >7500 additional open-source packages available in conda website
- Equipped with Jupyter Notebook
- Conda environment manager for easy maintenance of packages

SETTING UP PYTHON ENVIRONMENT (Anaconda 3)

Installing Anaconda 3

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

Anaconda Installers

Windows

Python 3.8

64-Bit Graphical Installer (457 MB)

32-Bit Graphical Installer (403 MB)

MacOS

Python 3.8

64-Bit Graphical Installer (435 MB)

64-Bit Command Line Installer (428 MB)

Linux

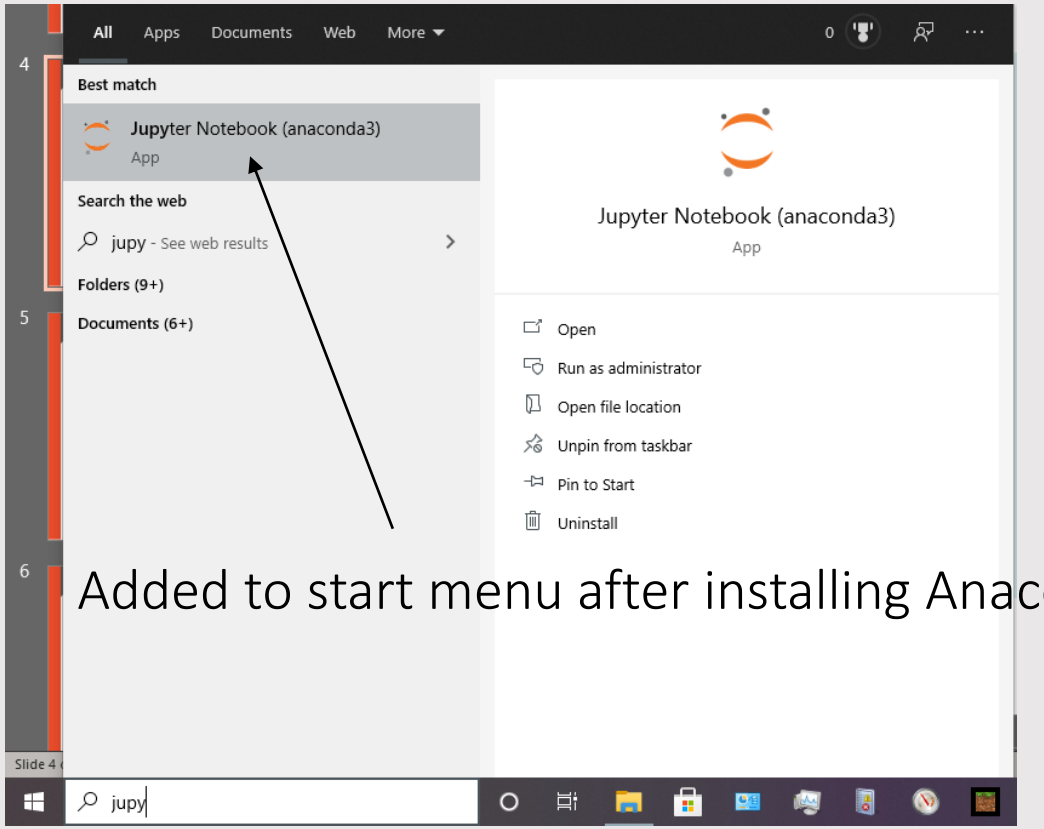
Python 3.8

64-Bit (x86) Installer (529 MB)

64-Bit (Power8 and Power9) Installer (279 MB)

STARTING UP JUPYTER NOTEBOOK (Anaconda3)

Windows



Added to start menu after installing Anaconda 3

Mac/Linux

Start terminal

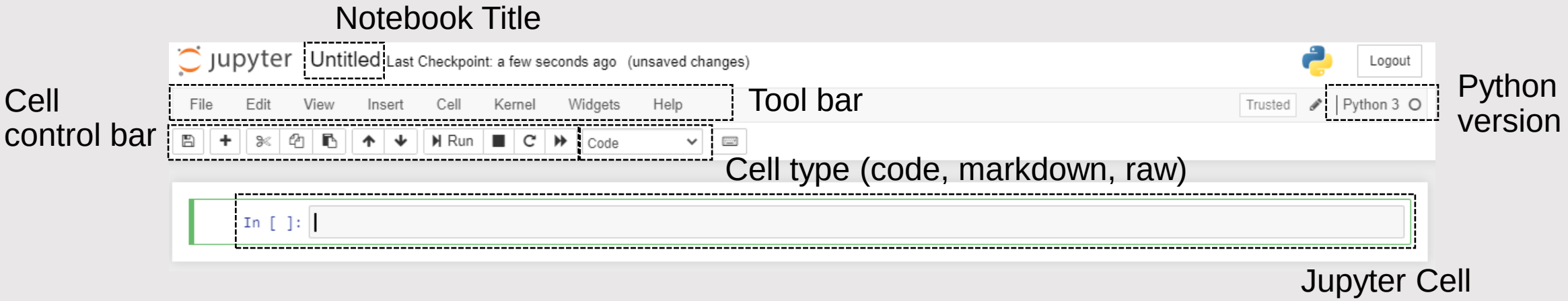
Type “jupyter notebook”

STARTING UP JUPYTER NOTEBOOK (Anaconda3)

Create a new notebook

You can also use Jupyter Navigator to load .ipynb notebook files

STARTING UP JUPYTER NOTEBOOK (Anaconda3)



See https://www.dataquest.io/blog/jupyter-notebook-tutorial to familiarize yourself with basic controls

GOOGLE COLABORATOTY

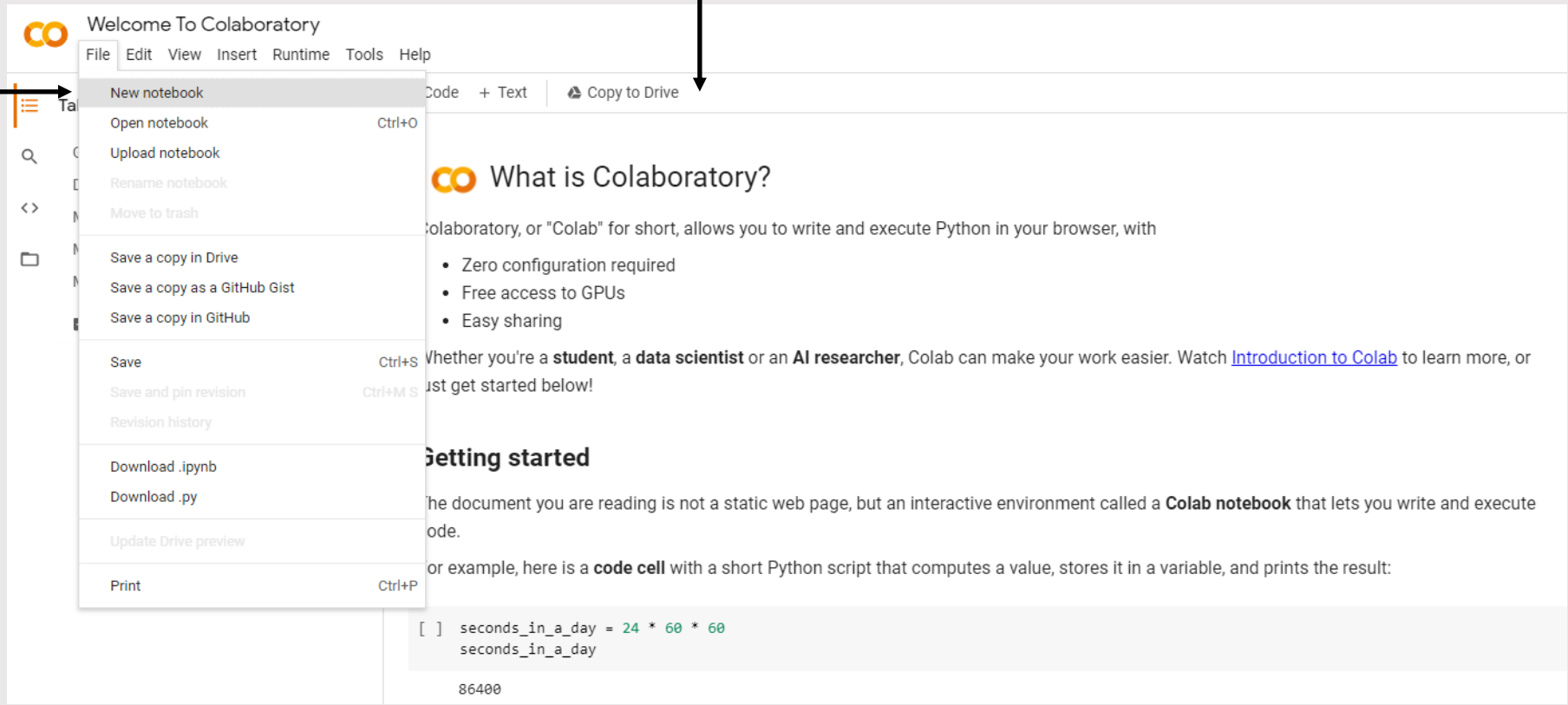
A free Jupyter notebook environment that runs in the cloud

- Saves in Google drive
- Github commit style code sharing with others
- Maximum runtime of 12hrs (Free version)
- Pre-equipped with latest scientific packages (Numpy, Scipy, etc)

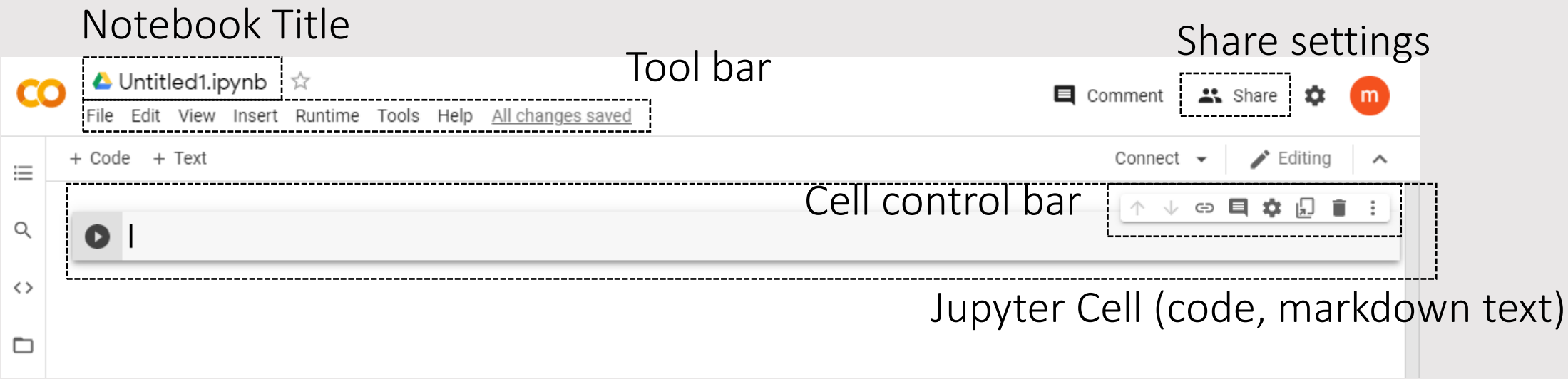
GOOGLE COLABORATOTY: GETTING STARTED

Tutorial to Colab https://colab.research.google.com/notebooks/intro.ipynb

Create new Notebook



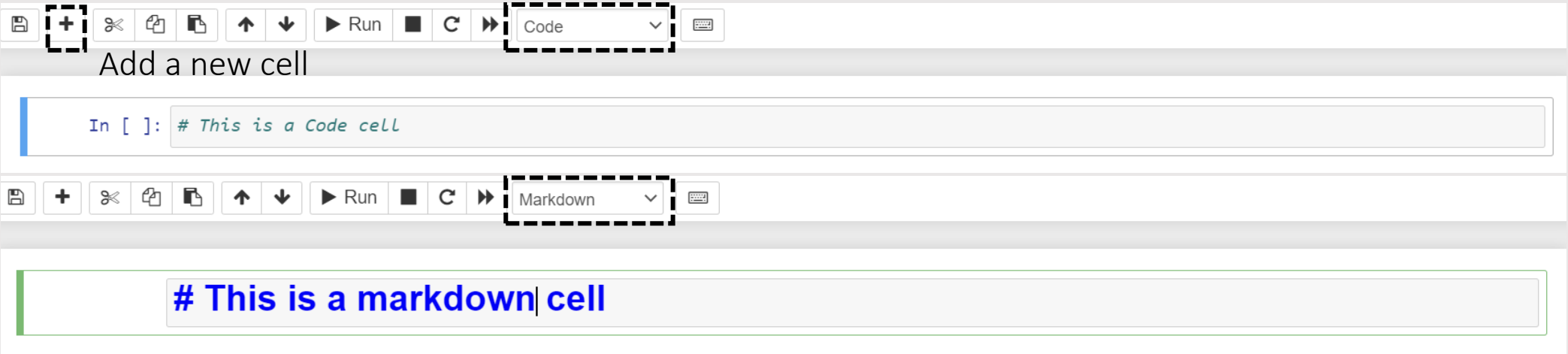
GOOGLE COLABORATOTY: GETTING STARTED



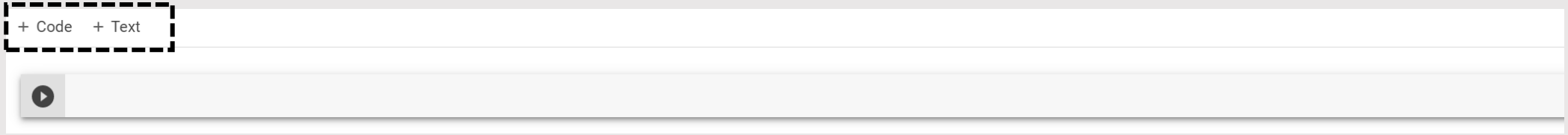
See **Getting Started** part of https://colab.research.google.com/notebooks/intro.ipynb to familiarize yourself with basic controls

CONFIGURING JUPYTER CELL: CODE vs MARKDOWN

Jupyter Notebook



Google Colab



PYTHON PACKAGES

Jupyter Notebook

```
import package_1
import package_2
import package_3
```

Installed packages

Package 1

Package 2

Package 3

⋮

Internet

Package x

Package y

Package z

Importing Packages

Installing Packages

PACKAGE MAINTENANCE: ANACONDA

Installing a Conda package

Mac or Linux : Use Terminal

Anaconda Prompt (anaconda3)

```
(base) C:\Users\Jimin>conda install package-name
```

Installing specific version of Conda package

Anaconda Prompt (anaconda3)

```
(base) C:\Users\Jimin>conda install package-name=2.3.4
```

Note: For Windows – Make sure to use **Anaconda Prompt** NOT Windows Command Prompt
Note: Only use pip-install when the package is not available in conda
Note: Search package name in Anaconda.org search bar to find more install versions

For more information: https://docs.anaconda.com/anaconda/user-guide/tasks/install-packages/

PACKAGE MAINTENANCE: GOOGLE COLAB

Package that is not default in Google Colab -> Use !pip install or !apt-get install

```
[ ]  !pip install matplotlib-venn
```

```
[ ]  !apt-get -qq install -y libfluidsynth1
```

Note: Some hardware (e.g sound card) dependent packages such as simpleaudio might not work with Google Colab

For more information: https://colab.research.google.com/notebooks/snippets/importing_libraries.ipynb

PYTHON’S DATA TYPES AND VARIABLES

WHAT ARE DATA TYPES AND VARIABLES?

Jupyter Notebook Code

```
In [1]: x = 1
        print(x)

        1

In [2]: y = 2.5
        print(y)

        2.5

In [3]: z = True
        print(z)

        True

In [4]: s = 'hello'
        print(s)

        hello
```

Variable	Data Type	Value
x	int	1
y	float	2.5
z	bool	True
s	str	‘hello’

PRINTING VARIABLES WITH print()

Print single variable

```
var1 = 2021
var2 = 'Fall'

print(var1)
```

```
2021
```

Print multiple variable

```
print(var1, var2)
```

```
2021 Fall
```

Variables called in a cell can be displayed without print function, as 'outputs'

```
var1
```

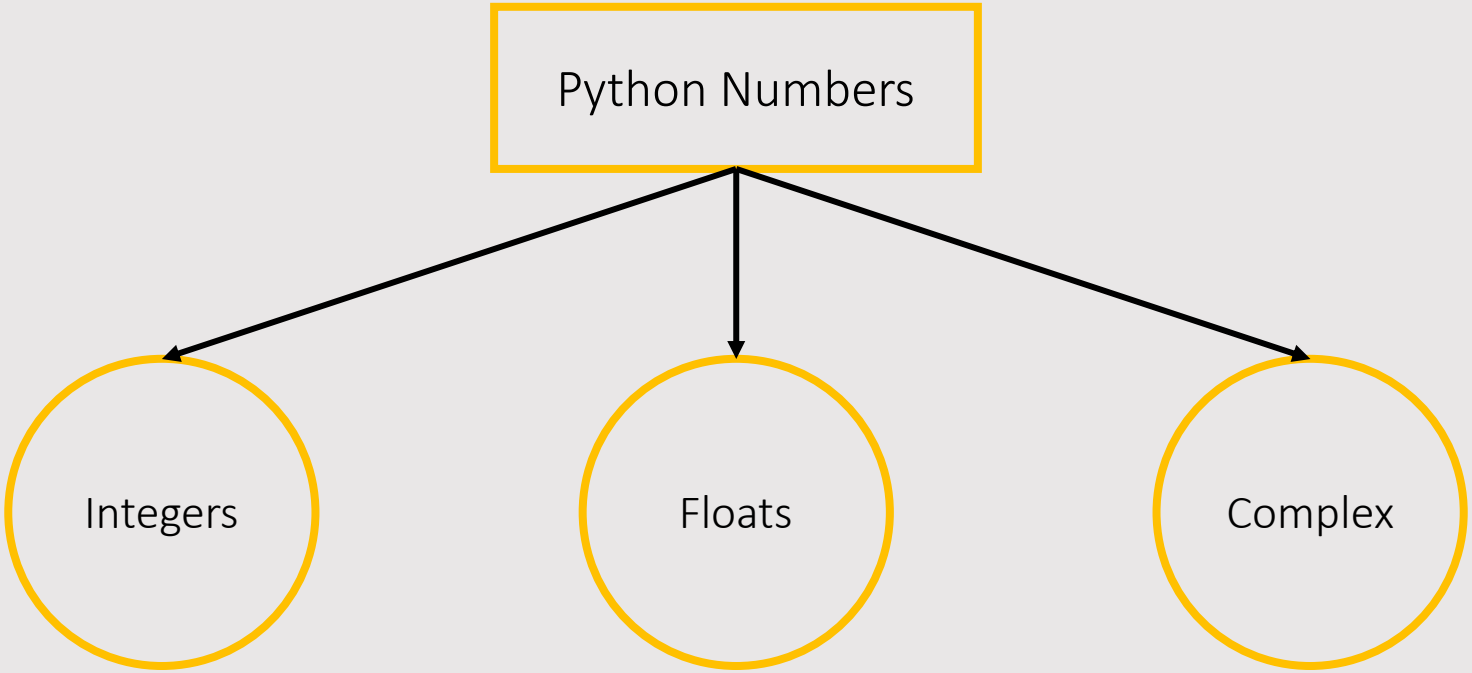
```
2021
```

```
var1, var2
```

```
(2021, 'Fall')
```

DATA TYPES: NUMBERS

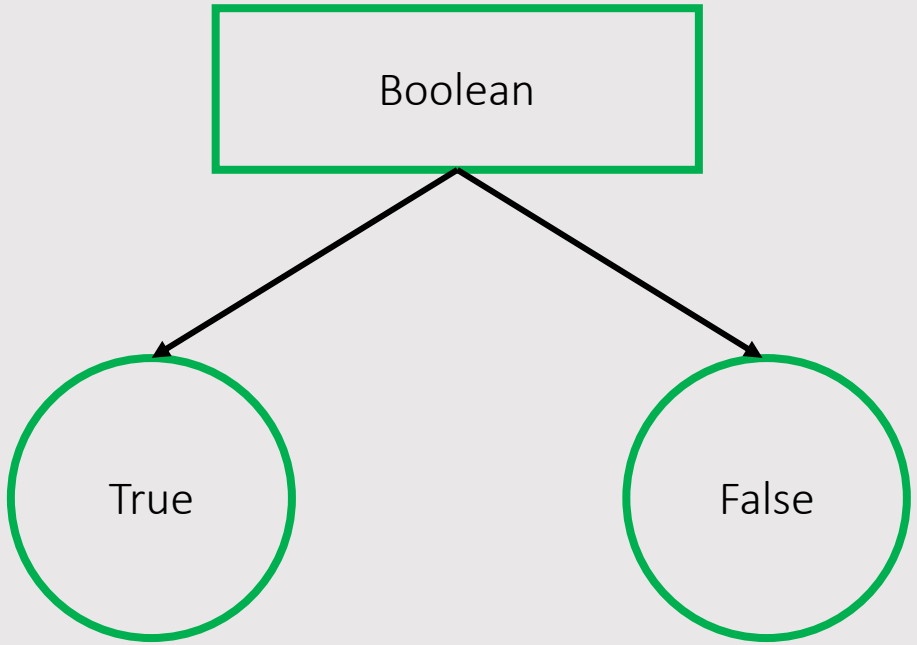
```
x = 1        # Integer
y = 1.6      # Float
z = 2 + 6j   # Complex
```



Examples	1, 300, -50	0.001, -15.7, 6.5, 1.0	6.4 – 3.6j, 8j, -10-2j

DATA TYPES: BOOLEAN

```
x = True        # True

y = False       # False
```



First letter should be capitalized

DATA TYPES: STRINGS

```
x = 'Hello World'
```

	H	e	l	l	o		W	o	r	l	d
Index	0	1	2	3	4	5	6	7	8	9	10

Length of string = 11

GROUPING DATA WITH LISTS

```
In [1]: list_1 = [1, 2, 3]
        list_1

Out[1]: [1, 2, 3]
```

List of numbers

```
In [2]: list_2 = ['Hello', 'World']
        list_2

Out[2]: ['Hello', 'World']
```

List of strings

```
In [3]: list_3 = [1, 2, 3, 'Apple', 'orange']
        list_3

Out[3]: [1, 2, 3, 'Apple', 'orange']
```

List of numbers + strings

```
In [4]: list_4 = [list_1, list_2]
        list_4

Out[4]: [[1, 2, 3], ['Hello', 'World']]
```

List of lists

INDEXING LISTS

```
In [3]: list_3 = [1, 2, 3, 'Apple', 'orange']
        list_3

Out[3]: [1, 2, 3, 'Apple', 'orange']

In [5]: list_3[2]

Out[5]: 3

In [6]: list_3[:3]

Out[6]: [1, 2, 3]

In [7]: list_3[-1]

Out[7]: 'orange'

In [8]: list_3[-3:]

Out[8]: [3, 'Apple', 'orange']
```

	1	2	3	‘Apple’	‘orange’
Index	0	1	2	3	4

More information on indexing:
https://railsware.com/blog/python-for-machine-learning-indexing-and-slicing-for-lists-tuples-strings-and-other-sequential-types/

APPEND, INSERT, DELETE ELEMENTS TO LISTS

```
In [10]: list_3.append(4)
         list_3

Out[10]: [1, 2, 3, 'Apple', 'orange', 4]
```

Appending a new value

```
In [12]: list_3.insert(2,'pineapple')
         list_3

Out[12]: [1, 2, 'pineapple', 3, 'Apple', 'orange']
```

Inserting a new value into an index

2: Index to insert,

'pineapple': Value to insert

```
In [14]: del list_3[2]
         list_3

Out[14]: [1, 2, 'Apple', 'orange']
```

Deleting an existing value

2: Index to delete

EMPTY LIST AND ELEMENT CHECK

```
In [15]: empty_list = []
         empty_list.append(5)
         empty_list

Out[15]: [5]
```

Appending a value to an empty list []

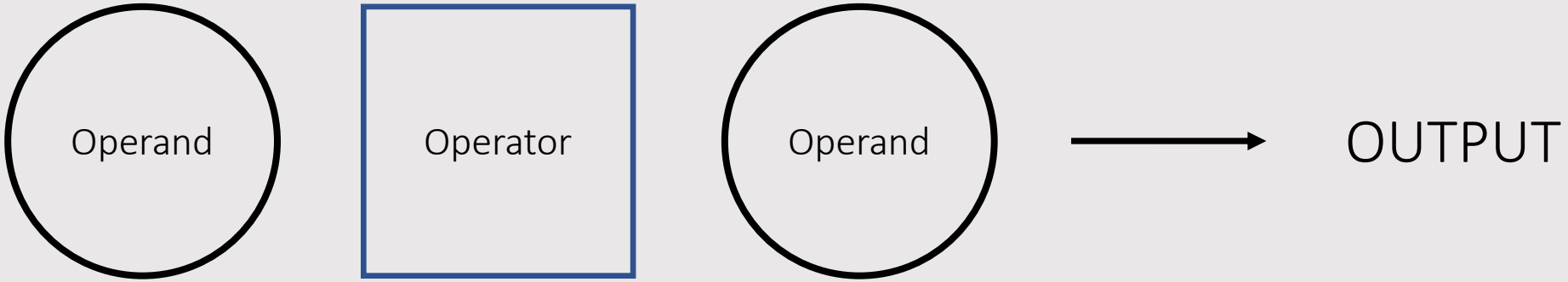
```
In [16]: 5 in empty_list

Out[16]: True
```

Checking if an element is in the list

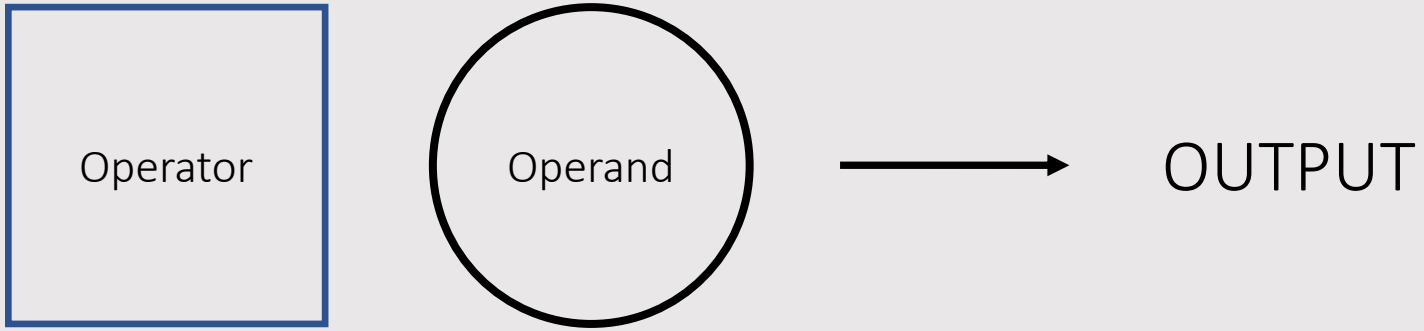
LOGICAL EXPRESSIONS AND OPERATORS

OPERATORS IN PYTHON

1.



2.



ARITHMATIC OPERATORS

	Operator	Example
Addition	+	

```
float1, float2 = 5.4, 8.9
print(float1 + float2)
```

```
14.3
```

Subtraction	-	

```
print(float1 - float2)
```

```
-3.5
```

Multiplication	*	

```
print(float1 * float2)
```

```
48.06
```

Exponent	**	

```
print(float1**2)
```

```
29.160000000000004
```

Division	/	

```
print(float1 / float2)
```

```
0.6067415730337079
```

Modulo	%	

```
float1, float2 = 10., 3.
print(float1 % float2)
```

```
1.0
```

COMPARISON OPERATORS

	Operator	Example
Greater Than	<	`5 < 3` → `False`
Less Than	>	`5 > 3` → `True`
Greater Than or Equal to	>=	`5 >= 3` → `True`
Less Than or Equal to	<=	`5 <= 3` → `False`
Equivalent to	==	`5 == 3` → `False`
Not Equivalent to	!=	`5 != 3` → `True`

ASSIGNMENT OPERATORS

	Operator	Example

Add and Assign	+=	`var1 = 3`
		`var1 += 1`
		`print(var1)`
		
		`4`

Subtract and Assign	-=	`var1 -= 1`
		`print(var1)`
		
		`3`

Multiply and Assign	*=	`var1 *= 1.5`
		`print(var1)`
		
		`4.5`

Divide and Assign	/=	`var1 /= 2`
		`print(var1)`
		
		`2.25`

LOGICAL OPERATORS

	Operator	Example
OR	or	
AND	and	
NOT	not	

```
bool1, bool2 = True, False
print(bool1 or bool2)
```

```
True
```

```
print(bool1 and bool2)
```

```
False
```

```
print(not bool1)
```

```
False
```

MATH OPERATORS WITH math. PACKAGE

	Operator
Sine	math.sin(x)
Cosine	math.cos(x)
Tangent	math.tan(x)
Pi	math.pi
Square Root	math.sqrt(x)
Exponential	math.exp(x)

x in radians

Example

```
import math

print(math.sin(1))
```

```
0.8414709848078965
```

```
print(math.cos(1))
```

```
-0.8390715290764524
```

```
print(math.tan(1))
```

```
0.6483608274590866
```

```
print(math.pi)
```

```
3.141592653589793
```

```
print(math.sqrt(3))
```

```
1.7320508075688772
```

```
print(math.exp(3))
```

```
20.085536923187668
```

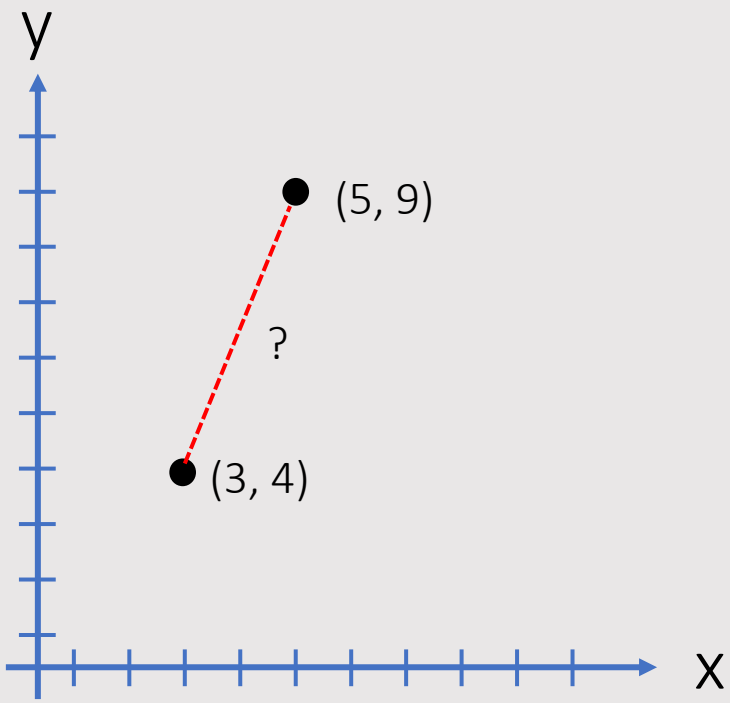
More functions: https://docs.python.org/3/library/math.html#

LAB ASSIGNMENTS

Download ipynb template in Canvas page:
Assignments/Lab 1 report -> click “Lab 1 Report Templates”

EXERCISE 1: Euclidean Distance

- Use Python operators and math package to determine the distance between two points (3, 4) and (5, 9).
- Assign the distance to a variable ‘dist’ and use print() command to output your result.



$$dist = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

EXERCISE 2: Sine and Cosine Values

- Create two lists – sine_list and cosine_list that contain sine and cosine values for the following angles –

$$180°, 90°, 45°, 30°$$

- Output two lists using print() command

Hint: Convert the degree units to radians first

EXERCISE 3: Implement sinh(x) from scratch

The hyperbolic sin or sinh is defined as:

$$\sinh(x) = \frac{\exp(x) - \exp(-x)}{2}$$

- Manually implement sinh(x) using exponential – math.exp() and appropriate operators. Compute sinh(x) when x = 2. Assign the output as a variable 'sinh_manual'

- Verify your implementation by comparing 'sinh_manual' with math.sinh(2). Use appropriate comparison operator and print the Boolean output.

EXERCISE 4: Implement XOR Gate from scratch

The XOR gate has following truth table

p	q	XOR
False	False	False
True	False	True
False	True	True
True	True	False

- Construct the logical expression equivalent to XOR gate by combining AND, OR and NOT operators.
- Validate your expression by printing its Boolean output for all 4 combinations of (p, q) values in the truth table.
- Note: You must use the same logical expression for each truth table row with only p and q values changing.
- Hint: XOR = (OR) AND (NAND) where NAND = NOT(AND)

EXERCISE 5: List indexing

The lab1_template.ipynb contains a sample list which contains integers from 0 – 100.

Use the appropriate list indexing commands to retrieve and print the following data from the list:

- Last 25 values of the list
- Values that fall between 1/4 to 3/4 of the list's length
- Values that correspond to every even index (0, 2, 4…)
- Value that correspond to every odd index (1, 3, 5…)