



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



Offline

Google Colaboratory



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)

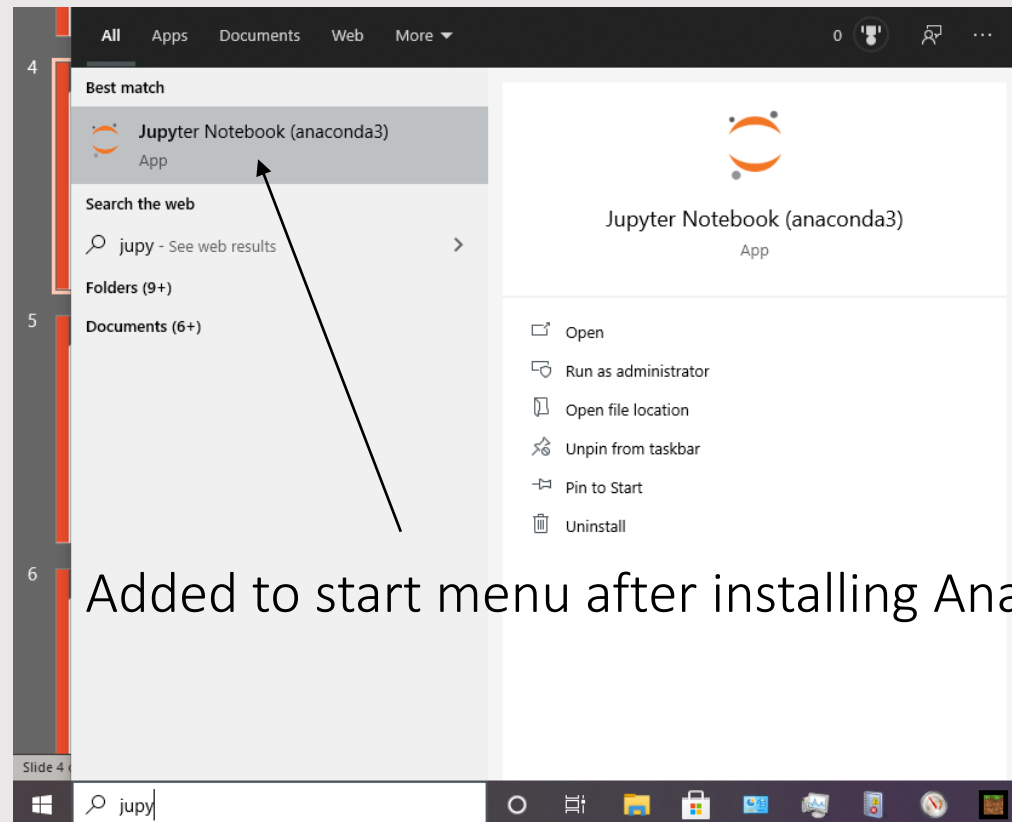
Installing Anaconda 3 <https://www.anaconda.com/products/individual>

Anaconda Installers

Windows 	MacOS 	Linux 
Python 3.8	Python 3.8	Python 3.8
64-Bit Graphical Installer (457 MB)	64-Bit Graphical Installer (435 MB)	64-Bit (x86) Installer (529 MB)
32-Bit Graphical Installer (403 MB)	64-Bit Command Line Installer (428 MB)	64-Bit (Power8 and Power9) Installer (279 MB)

STARTING UP JUPYTER NOTEBOOK (Anaconda3)

Windows

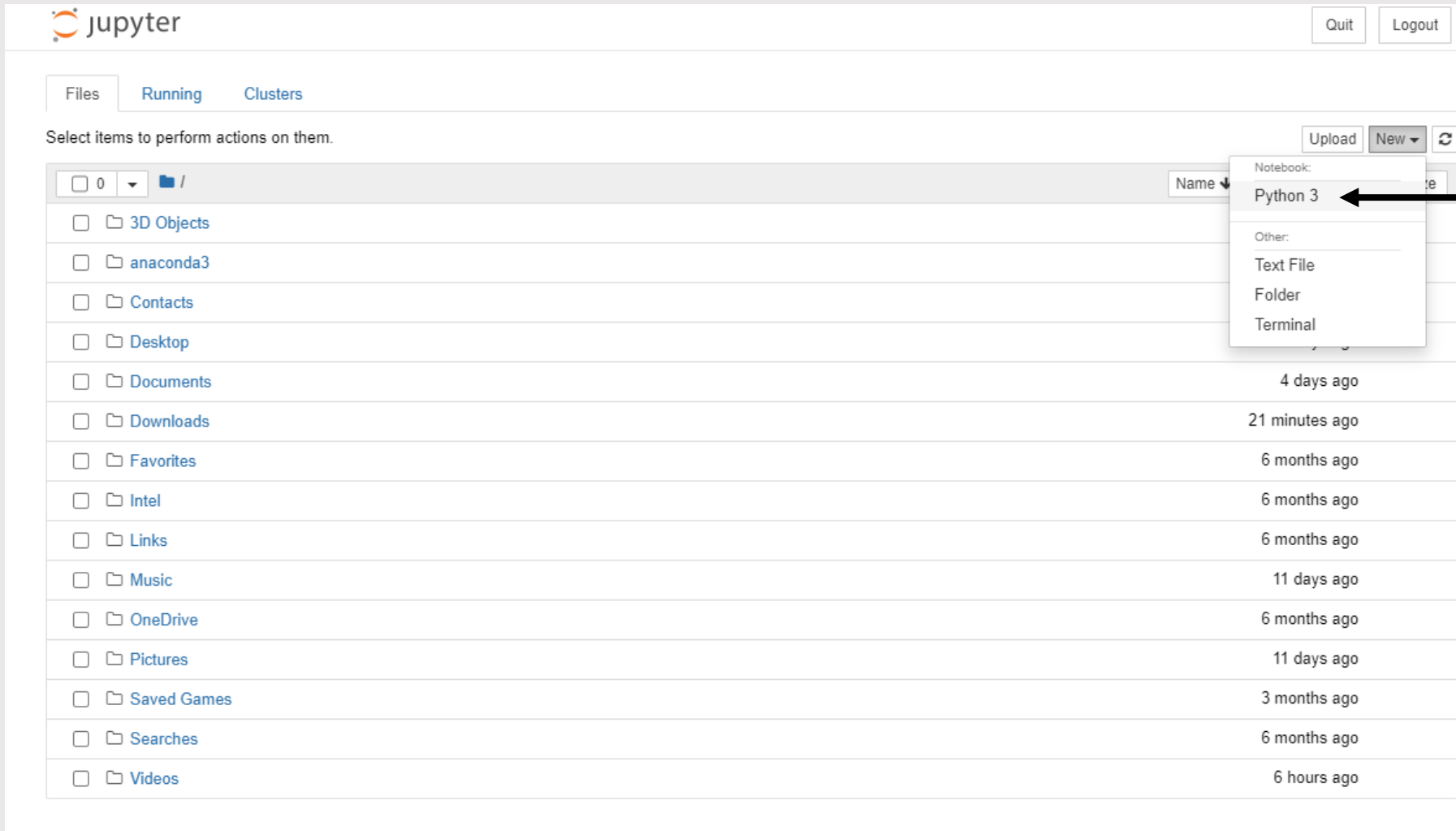


Added to start menu after installing Anaconda 3

Mac/Linux

Start terminal
Type "jupyter notebook"

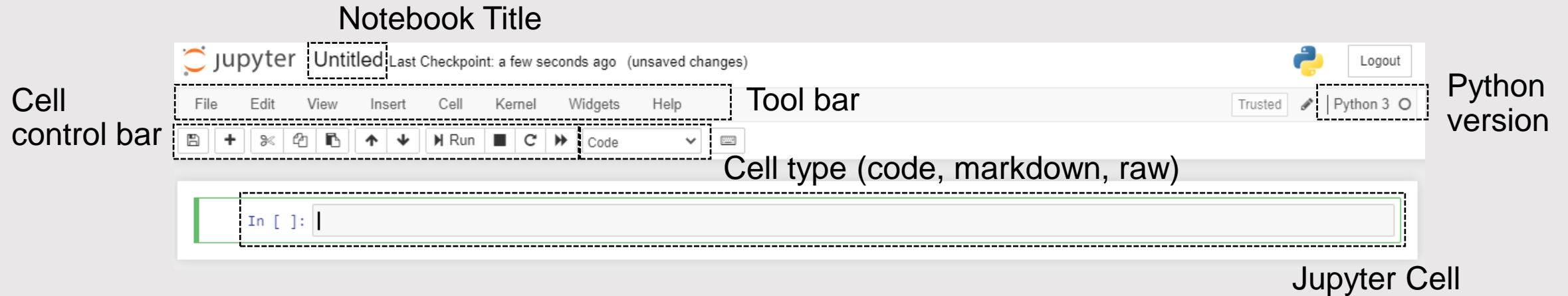
STARTING UP JUPYTER NOTEBOOK (Anaconda3)



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 COLABORATORY

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)

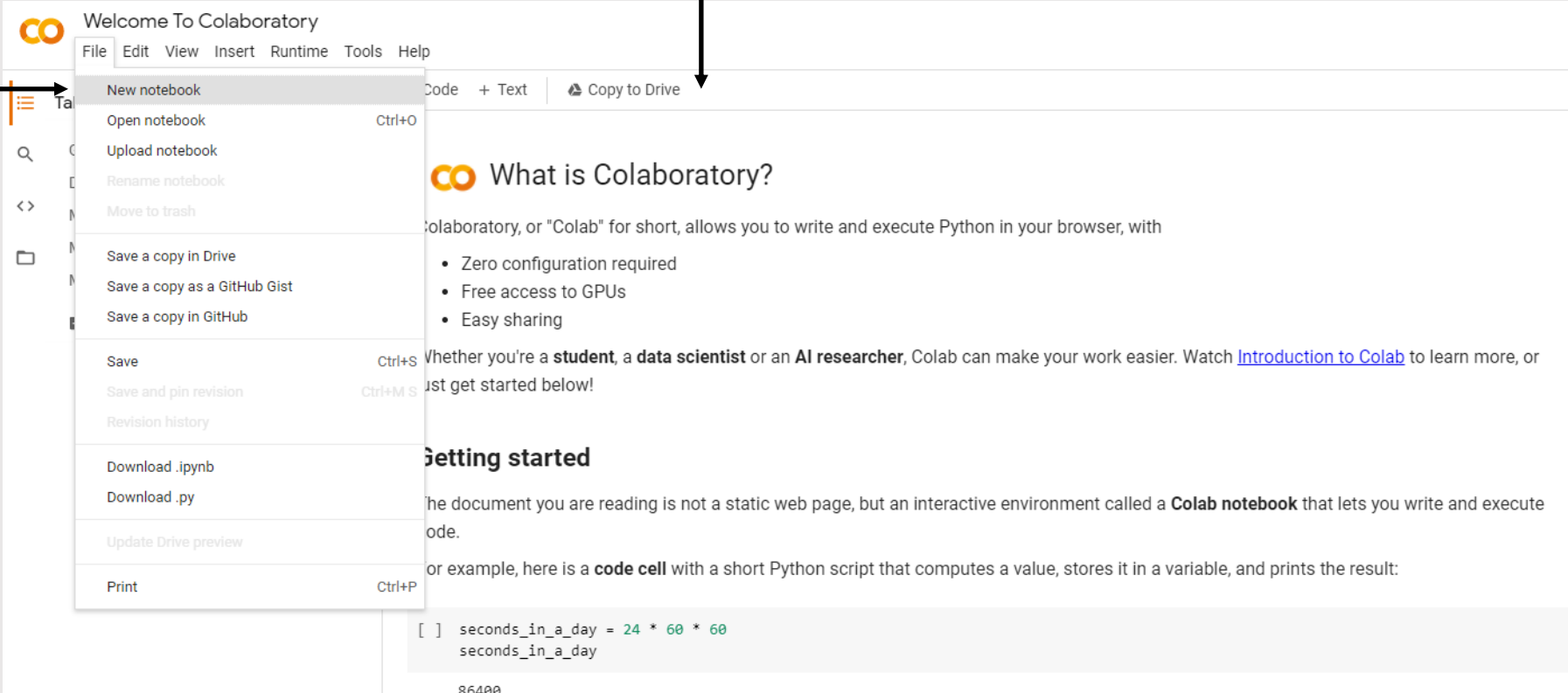


GOOGLE COLABORATORY: GETTING STARTED

Tutorial to Colab

<https://colab.research.google.com/notebooks/intro.ipynb>

Create new
Notebook



The screenshot shows the Google Colaboratory web interface. The 'File' menu is open, and 'New notebook' is highlighted. An arrow points from the text 'Create new Notebook' to this menu item. Another arrow points from the URL 'https://colab.research.google.com/notebooks/intro.ipynb' to the 'Copy to Drive' button in the top right of the interface. The main content area displays the 'What is Colaboratory?' page, which includes a list of features: Zero configuration required, Free access to GPUs, and Easy sharing. Below this, it says 'Whether you're a student, a data scientist or an AI researcher, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!'. The 'Getting started' section begins with 'The document you are reading is not a static web page, but an interactive environment called a Colab notebook that lets you write and execute code.' and provides an example of a code cell with a Python script that calculates the number of seconds in a day (24 * 60 * 60), resulting in 86400.

Welcome To Colaboratory

File Edit View Insert Runtime Tools Help

New notebook
Open notebook Ctrl+O
Upload notebook
Rename notebook
Move to trash
Save a copy in Drive
Save a copy as a GitHub Gist
Save a copy in GitHub
Save Ctrl+S
Save and pin revision Ctrl+M S
Revision history
Download .ipynb
Download .py
Update Drive preview
Print Ctrl+P

Code + Text Copy to Drive

What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!

Getting started

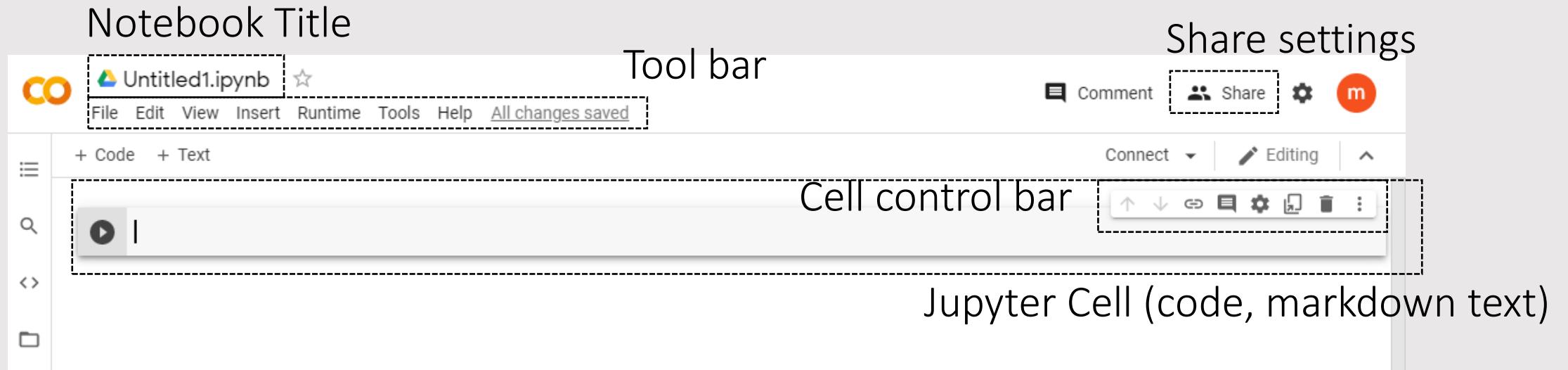
The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ] seconds_in_a_day = 24 * 60 * 60
    seconds_in_a_day

86400
```

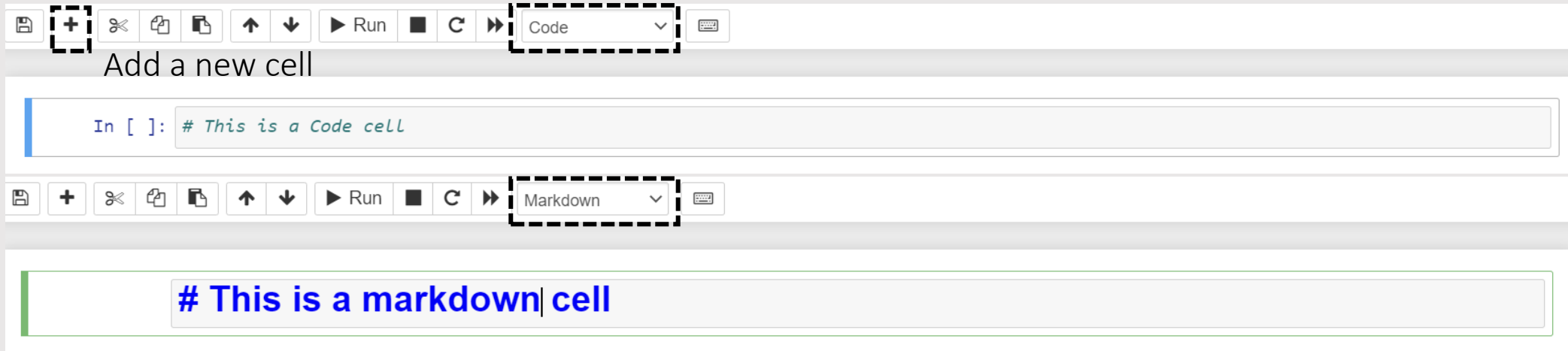
GOOGLE COLABORATORY: GETTING STARTED



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

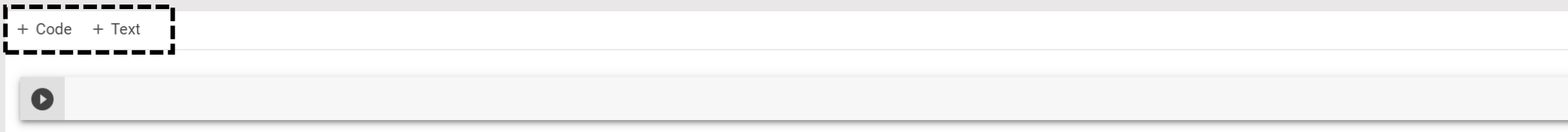
CONFIGURING JUPYTER CELL: CODE vs MARKDOWN

Jupyter Notebook



The screenshot displays the Jupyter Notebook interface. At the top, a toolbar contains icons for saving, adding a new cell, cutting, copying, pasting, moving up/down, running, and refreshing. The 'Add a new cell' icon (a plus sign) is highlighted with a dashed box. Below the toolbar, the text 'Add a new cell' is visible. The main area shows two cells. The first cell is a Code cell, indicated by the 'Code' dropdown in the toolbar, containing the text 'In []: # This is a Code cell'. The second cell is a Markdown cell, indicated by the 'Markdown' dropdown in the toolbar, containing the text '# This is a markdown cell'. Both dropdowns in the toolbars are highlighted with dashed boxes.

Google Colab



The screenshot displays the Google Colab interface. At the top, a toolbar contains icons for saving, adding a new cell, cutting, copying, pasting, moving up/down, running, and refreshing. The 'Add a new cell' icon (a plus sign) is highlighted with a dashed box. Below the toolbar, the text '+ Code + Text' is visible. The main area shows a single cell with a play button icon on the left.

PYTHON PACKAGES

Jupyter Notebook

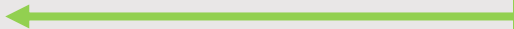
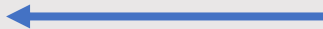
Installed packages

Internet

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

- Package 1
- Package 2
- Package 3

- Package x
- Package y
- Package z



⋮



Importing Packages



Installing Packages



PACKAGE MAINTENANCE: ANACONDA

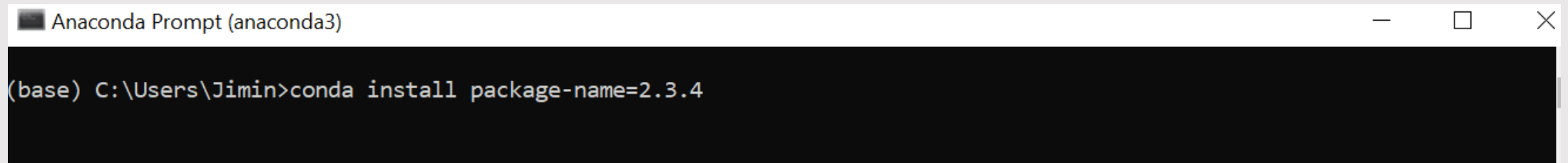
Installing a Conda package

Mac or Linux : Use Terminal

A screenshot of the Anaconda Prompt window. The title bar reads "Anaconda Prompt (anaconda3)". The command prompt shows the command `(base) C:\Users\Jimin>conda install package-name` entered at the prompt.

```
Anaconda Prompt (anaconda3)
(base) C:\Users\Jimin>conda install package-name
```

Installing specific version of Conda package

A screenshot of the Anaconda Prompt window. The title bar reads "Anaconda Prompt (anaconda3)". The command prompt shows the command `(base) C:\Users\Jimin>conda install package-name=2.3.4` entered at the prompt.

```
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
```

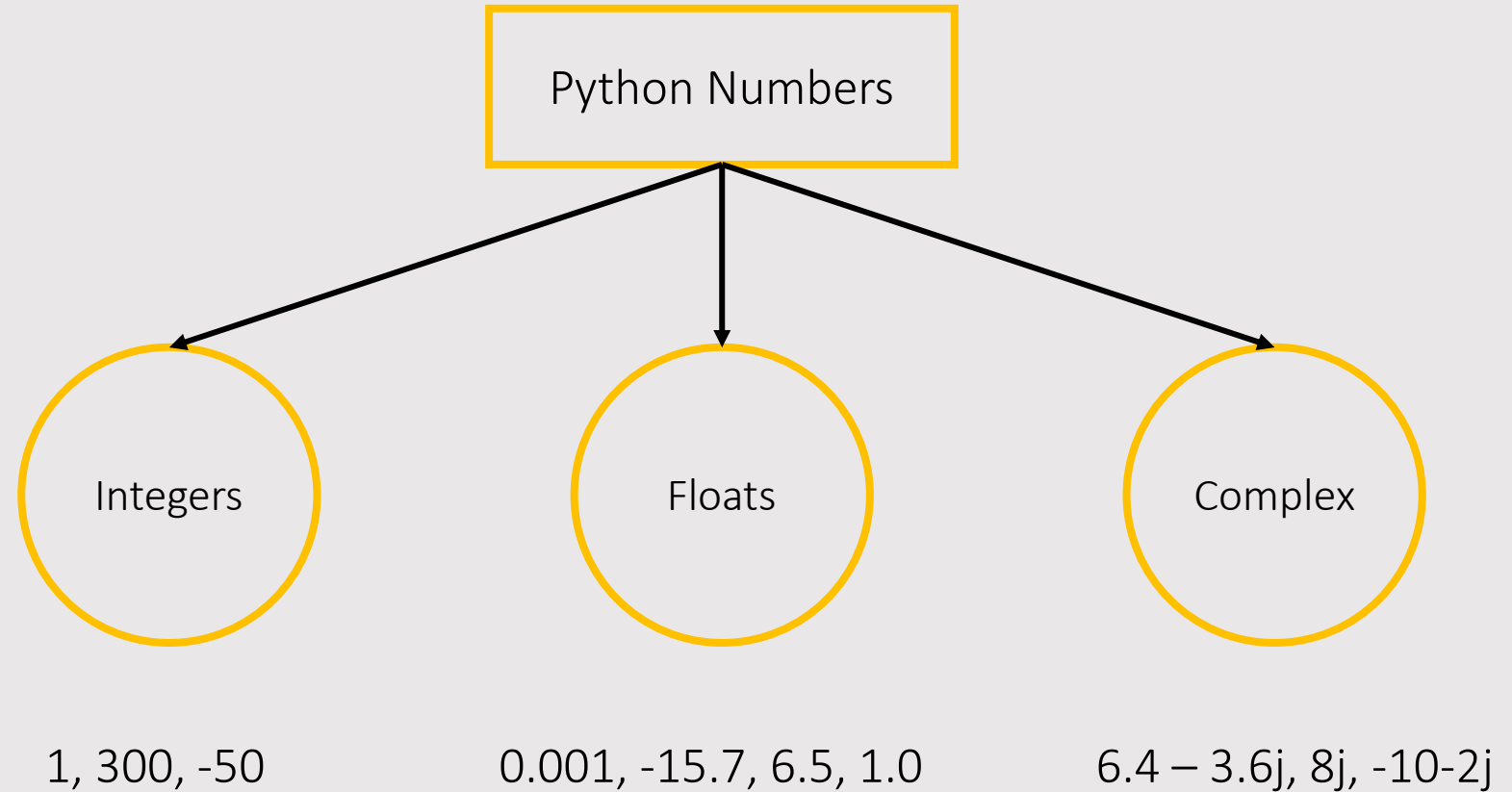
(2021, 'Fall')

DATA TYPES: NUMBERS

```
x = 1      # Integer
```

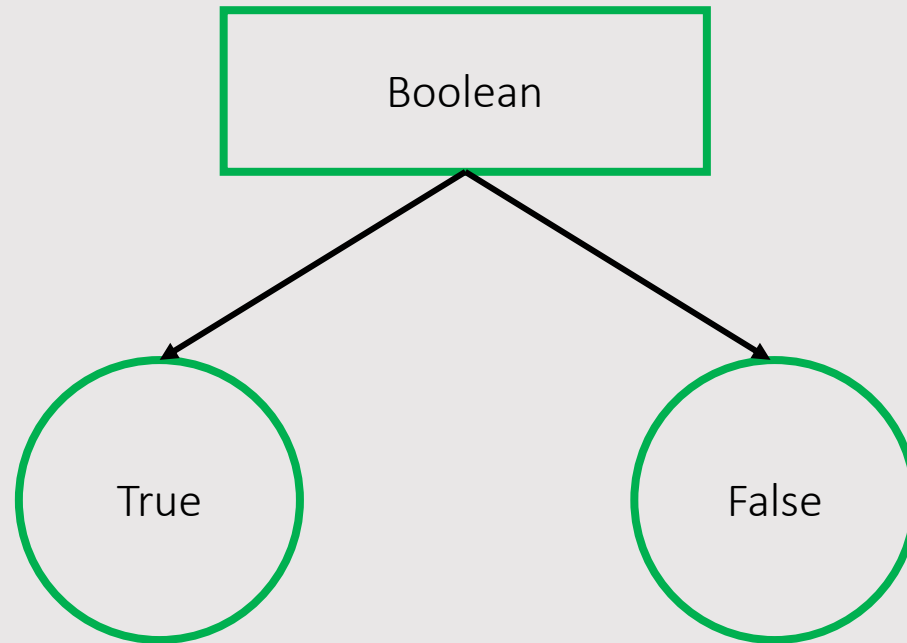
```
y = 1.6    # Float
```

```
z = 2 + 6j  # Complex
```



DATA TYPES: BOOLEAN

```
x = True      # True  
y = False     # False
```



First letter should be capitalized

DATA TYPES: STRINGS

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



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]
```

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

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

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

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

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

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

List of numbers

List of strings

List of numbers + strings

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]
```

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

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

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

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

Appending a new value

Inserting a new value into an index

2: Index to insert,
'pineapple': Value to insert

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]
```

```
In [16]: 5 in empty_list
```

```
Out[16]: True
```

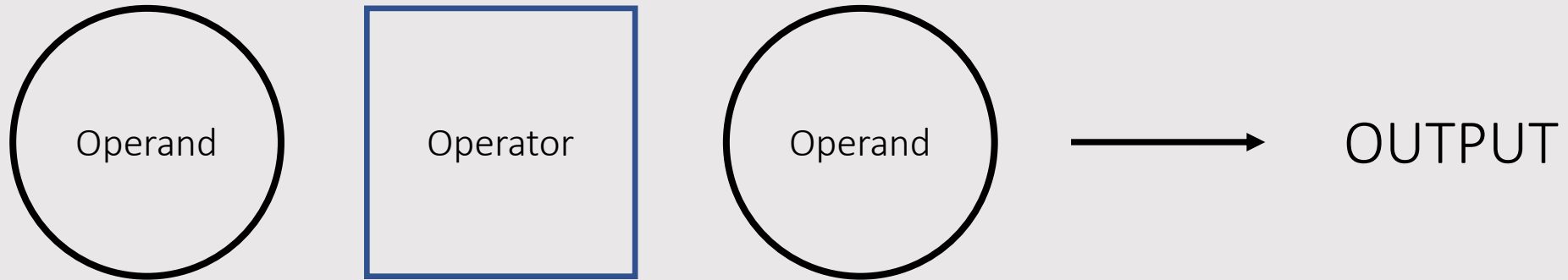
Appending a value to an empty list []

Checking if an element is in the list

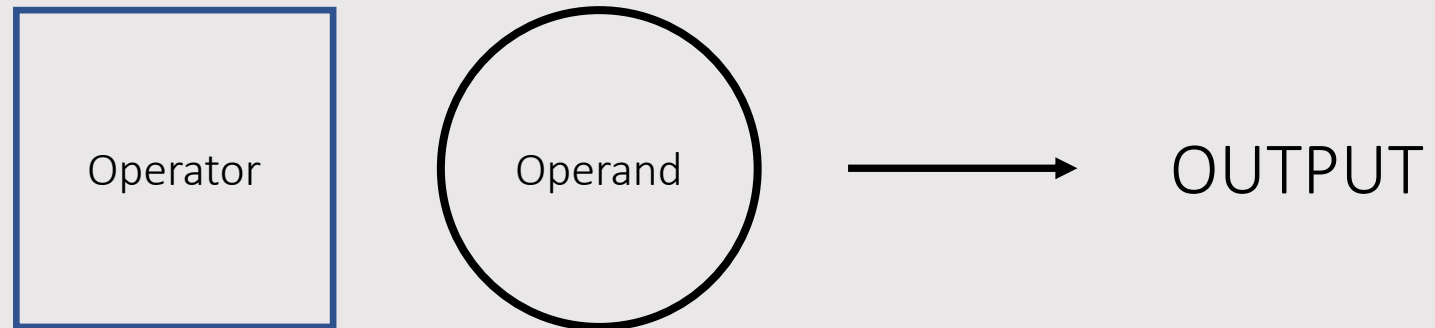
LOGICAL EXPRESSIONS AND OPERATORS

OPERATORS IN PYTHON

1.



2.



ARITHMETIC OPERATORS

	Operator	Example
Addition	+	<pre>float1, float2 = 5.4, 8.9 print(float1 + float2)</pre> 14.3
Subtraction	-	<pre>print(float1 - float2)</pre> -3.5
Multiplication	*	<pre>print(float1 * float2)</pre> 48.06
Exponent	**	<pre>print(float1**2)</pre> 29.160000000000004
Division	/	<pre>print(float1 / float2)</pre> 0.6067415730337079
Modulo	%	<pre>float1, float2 = 10., 3. print(float1 % float2)</pre> 1.0

COMPARISON OPERATORS

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

ASSIGNMENT OPERATORS

	Operator	Example
Add and Assign	<code>+=</code>	<pre>var1 = 3 var1 += 1 print(var1)</pre> <p>4</p>
Subtract and Assign	<code>-=</code>	<pre>var1 -= 1 print(var1)</pre> <p>3</p>
Multiply and Assign	<code>*=</code>	<pre>var1 *= 1.5 print(var1)</pre> <p>4.5</p>
Divide and Assign	<code>/=</code>	<pre>var1 /= 2 print(var1)</pre> <p>2.25</p>

LOGICAL OPERATORS

Operator

Example

OR

or

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

True

AND

and

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

False

NOT

not

```
print(not bool1)
```

False

MATH OPERATORS WITH math. PACKAGE

	Operator	Example
Sine	$\text{math.sin}(x)$	<pre>import math print(math.sin(1))</pre> <p>0.8414709848078965</p> <pre>print(math.cos(1))</pre> <p>-0.8390715290764524</p> <pre>print(math.tan(1))</pre> <p>0.6483608274590866</p>
Cosine	$\text{math.cos}(x)$	
Tangent	$\text{math.tan}(x)$	
Pi	math.pi	<pre>print(math.pi)</pre> <p>3.141592653589793</p>
Square Root	$\text{math.sqrt}(x)$	<pre>print(math.sqrt(3))</pre> <p>1.7320508075688772</p>
Exponential	$\text{math.exp}(x)$	<pre>print(math.exp(3))</pre> <p>20.085536923187668</p>

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

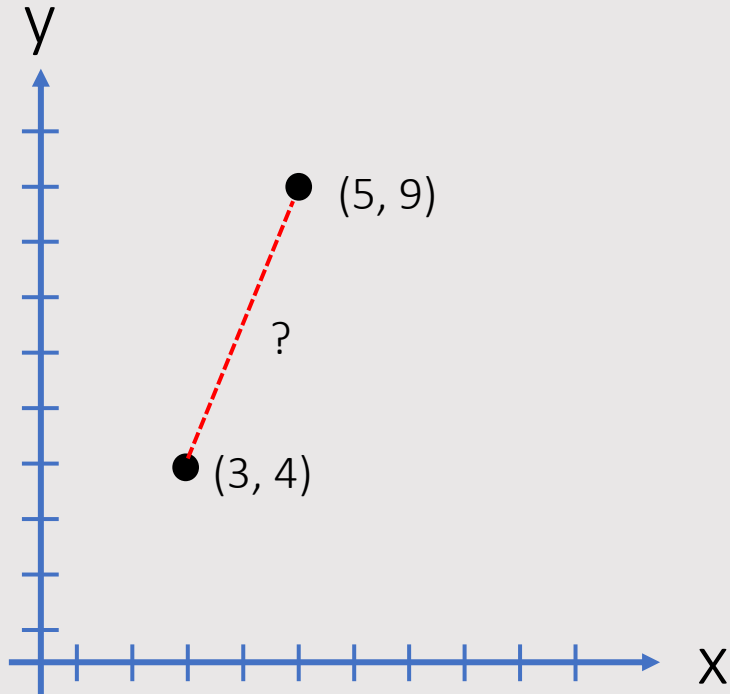
LAB ASSIGNMENTS

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 –

$180^\circ, 90^\circ, 45^\circ, 30^\circ$

- 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

- 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: $XOR = (OR) \text{ AND } (NAND)$ where $NAND = NOT(AND)$

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

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...)