

Lecture 1

Introduction & Python Fundamentals Part 1

Jan 2024

WARNING

This material has been reproduced and communicated to you by or on behalf of the University of New South Wales in accordance with section 113P(1) of the Copyright Act 1968 (Act).

The material in this communication may be subject to copyright under the Act. Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act.

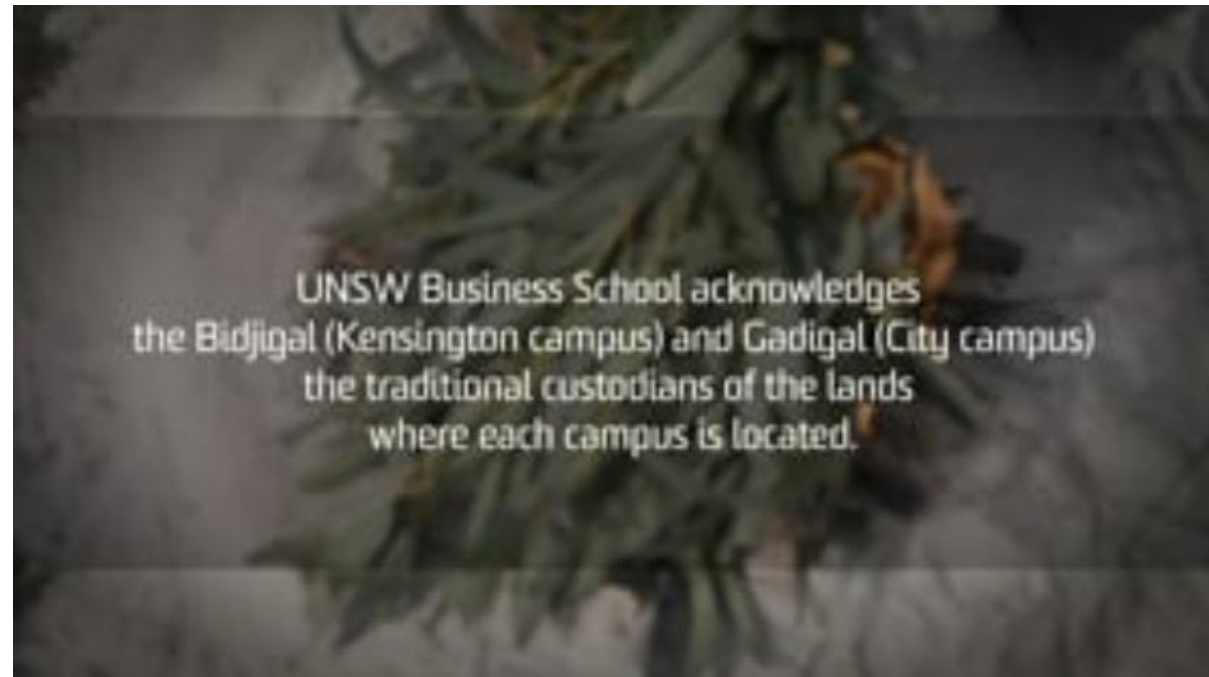
Do not remove this notice

Copyright

- We have seen file-sharing websites that specialise in buying and selling academic work to and from university students.
- If you upload your original work to these websites, and if another student downloads and presents it as their own either wholly or partially, you might be found guilty of collusion — even years after graduation.
- These file-sharing websites may also accept purchase of course materials, such as copies of lecture slides and tutorial handouts. By law, the copyright on course materials, developed by UNSW staff in the course of their employment, belongs to UNSW. It constitutes copyright infringement, if not academic misconduct, to trade these materials.

Country

- UNSW Business School acknowledges the Bidjigal (Kensington campus) and Gadigal (City campus) the traditional custodians of the lands where each campus is located.
- We acknowledge all Aboriginal and Torres Strait Islander Elders, past and present and their communities who have shared and practiced their teachings over thousands of years including business practices.
- We recognize Aboriginal and Torres Strait Islander people's ongoing leadership and contributions, including to business, education and industry.



UNSW Business School. (2023, August 18). *Acknowledgement of Country* [online video]. Retrieved from <https://vimeo.com/369229957/d995d8087f>

Outline

- **Introduction to the course**
- Python: A Programming Language
- Python Fundamentals Part 1

Course Structure

Week 1.1



Python
fundamentals 1

Week 2.1



Python
fundamentals 2

Week 2.2



Python
fundamentals 3

Week 3.1



Python
fundamentals 4

Week 3.2

Data Exploration
and Manipulation



Week 4.1

Data Insights



Week 4.2

File Operations

Week 5.1

Machine Learning
Basics with Python



Week 5.2

Advanced Topics
Course Review

How To Learn Python?

"the best way to learn a language
is to speak to natives"
the guy learning Python:



Image source: Cudoo

Learning Philosophy

- Learning by doing
- Not just a “Python course”— learning how to use programming to address problems
- Support/commentary being constructive
- Open-access materials provided

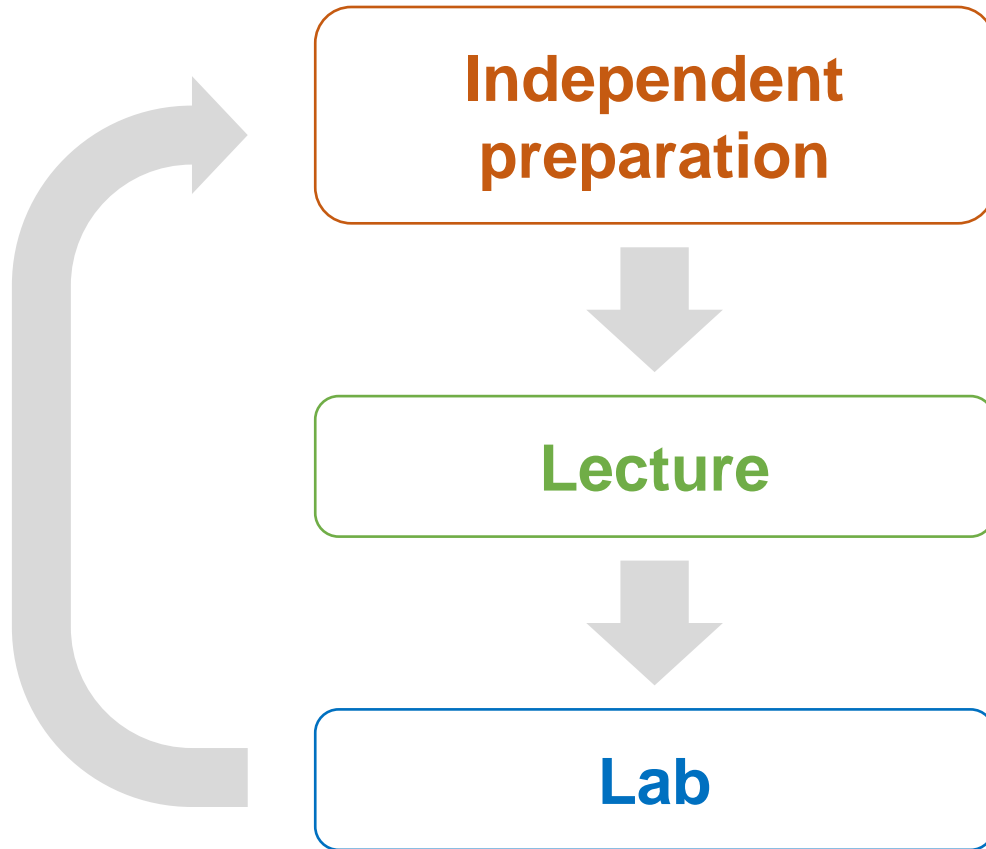


technê – ‘how to’

epistêmê – theory

phronêsis – practical wisdom / ethics

Course Structure



epistêmê

Including readings, online tutorials, etc..

phronêsis

Guide you through concepts visually and verbally.

technê

Hands-on practical skills.

Assessment components

■ 45% final examination

- 2 hours; Date TBC (Feb 3-5)
- **In person on campus**
- 80% will be multiple choice questions
- 20% writing questions, e.g., pinpoint errors given snippets of code, or explain real-world application of coding.
- Closed book with One (1) double-sided A4-sized help sheet, hand-written or printed

■ 25% Skills Challenges: Individual Assessment

- 10%: One (1) individual assignment
 - Turnitin submission of .docx file
- 15%: Two (2) quizzes
 - Conducted during lab 3 and lab 5 **in person on campus**
 - Multiple choice questions
 - Open book (not open laptop)

■ 30% team assignment

- Lab 8: Team Assignment Presentation **in person on campus (hence Mon tutorials in week 5 are in person)**
- Final submission: code, PPT

Overview of schedule

The Summer offering of COMM5007 will run intensively for 5 weeks in the U1 Summer period.

Instead of having the 1st lecture on Jan 1st, 2024 (public holiday), we will **skip Monday and Tuesday lecture and labs on Jan 1st and Jan 2nd**. The course will **start from Wed, Jan 3rd, 2024**, and have only 1 lecture and 1 lab in week 1. From week 2-5, students can expect to take 2 lectures and 2 labs every week, to complete the 9-week course.

Monday Labs from week 2-5 are online, to provide flexibility. Only Lab 8 LIVE presentation (Monday in week 5) must be in person.

All labs on Tue, Wed, Thurs are in person on campus.

All zoom links are in Left Menu > Zoom Link & Recordings.

Refer to this link for more details.

<https://timetable.unsw.edu.au/2024/COMM5007.html#X1S>

Weekly lesson plan

In each week, the convention goes by the numeric label X.Y, e.g.,

- week 1.1 means in week 1, lecture 1 and lab 1
- week 3.2 means in week 3, lecture 2 and lab 2, which is essentially lecture 5 and lab 5 within the entire course.

Week	Topics	Assignments/Activities
1.1	Introduction + Python Fundamentals Part 1	
2.1	Python Fundamentals Part 2	
2.2	Python Fundamentals Part 3	Lab 3: Quiz 1 (7.5%) In person on campus Individual Assignment release (10%)
3.1	Python Fundamentals Part 4	
3.2	Data Exploration and Manipulation	Lab 5: Quiz 2 (7.5%) In person on campus Individual Assignment submission by Fri, 19 Jan 2024, 14:59 Team Assignment guideline release (30%)
4.1	Data Insights	
4.2	File Operations	
5.1	Machine Learning Basics with Python	Lab 8: Team Assignment Presentation In person on campus Team Assignment submission by Mon, 29 Jan 2024, 12:59
5.2	Advanced Topics; Course Review	
	FINAL EXAM (Closed book with cheat sheet)	45%; 2 hours; Date TBC (Feb 3-5) In person on campus

Academic integrity

- Academic integrity is of utmost importance for all work presented and submitted for grading. As such, work presented must be one's own or one's group with appropriate citation from sources used, including the Internet, etc..
- You should **never copy code from classmates**.
- You should not simply re-use code from online resources. When you use or adapt code from online as part of your work, you must cite the source by including an inline comment in the code as well crediting the author in your report. You can **indicate “Adapted from:” or “Based on” and put in the URL** and the date of retrieval. Your work should still reflect your original approach to the problem.
- All submissions will **go through plagiarism checker**. Flagged cases will be reported to the school and face disciplinary actions.

Logistics

- **Consultation with lecturer:** By appointment, online only
- **Questions:** For generic coding or admin questions, please post your question in Ed Discussion. The teaching team will respond within 2 working days.
- **Marking:** Tutors will swap classes for marking individual assignment, to avoid bias
- **Week 0 pre-lecture task:** available on Moodle; complete before 1st lab
- **Announcements:** Constantly check Email (**“Other” inbox**) and Ed
- **Lab sessions:**
 - Bring Your Own Device
 - ONLY attend the lab that you have enrolled in (attending a different class requires you to make the change in enrollment system, which the teaching team has no admin rights to do)
 - ONLY form groups with students from the same lab session

How to post on Ed Discussion when your question is generic

When it is about code...

- Do research online first
- Include a clear or full description of your issue, including your attempted code
- Copy and format your “code” rather than give a screenshot
- Use a few sentences detailing your **current understanding**, followed by the question

When it is about administrative matters...

- Check Ed announcements first (for exam date, submission deadline etc.)
- Include a clear description of your doubt about the announcement

How to email when your question is not generic

- Use your **UNSW email address**
- Start the subject line with [COMM5007], e.g., [COMM5007] Lecture 1
- Sign your email with your full name, zID, lab session (and group number when you have one), e.g.,
Alice King
z1234567
M18A, or M18AG01

If you use AI tools (e.g., ChatGPT)...



Image source: Forbes

If you use ChatGPT to assist you on assignment, please make sure you **reference** it at the end of your assignment.

Readings

- **Main reading**
 - This course has no required textbook.
 - Lecture notes will be posted on the course website.
- **Recommended reading**
 - Chapters 1 – 9 of *Python for Everybody* by Charles R. Severance, available at <https://www.py4e.com/book.php> (Both the PDF and HTML versions are free.)
 - Chapters 1, 5, 7 and 8 of *Python for Data Analysis* (2nd edition) by Wes Mckinney
- **Other online resources for practice**
 - <https://www.codecademy.com/learn/learn-python>

Outline

- Introduction to the course
- **Python: A Programming Language**
- Python Fundamentals Part 1

Python as a Programming Language

- Python was developed in the late **1980s** by Dutch research programmer **Guido van Rossum**

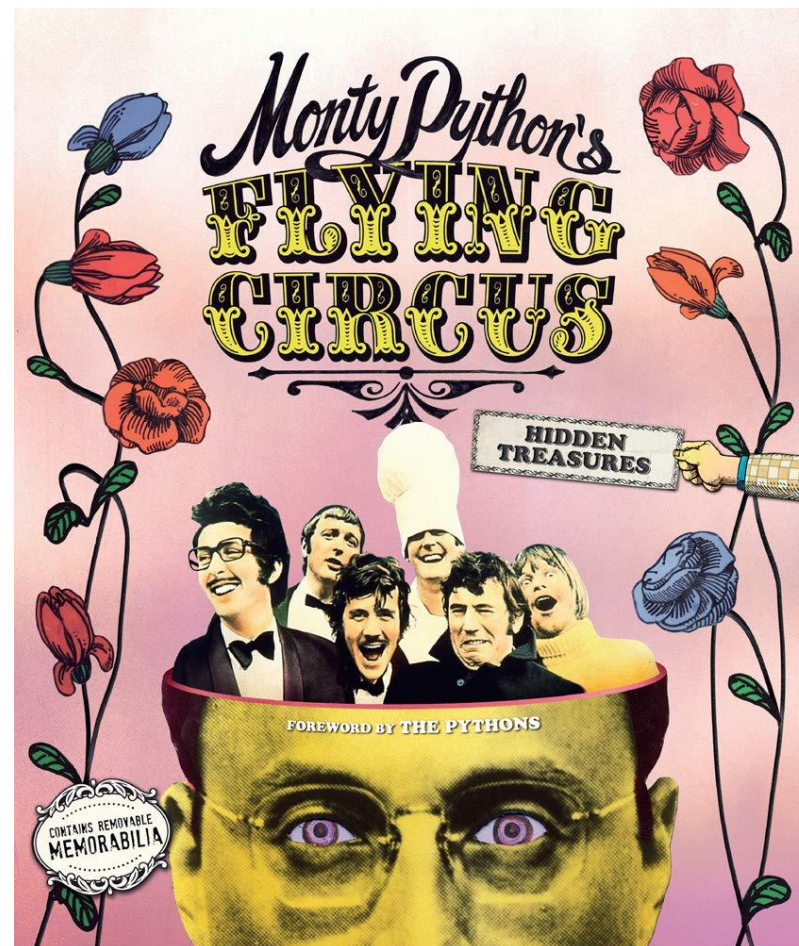
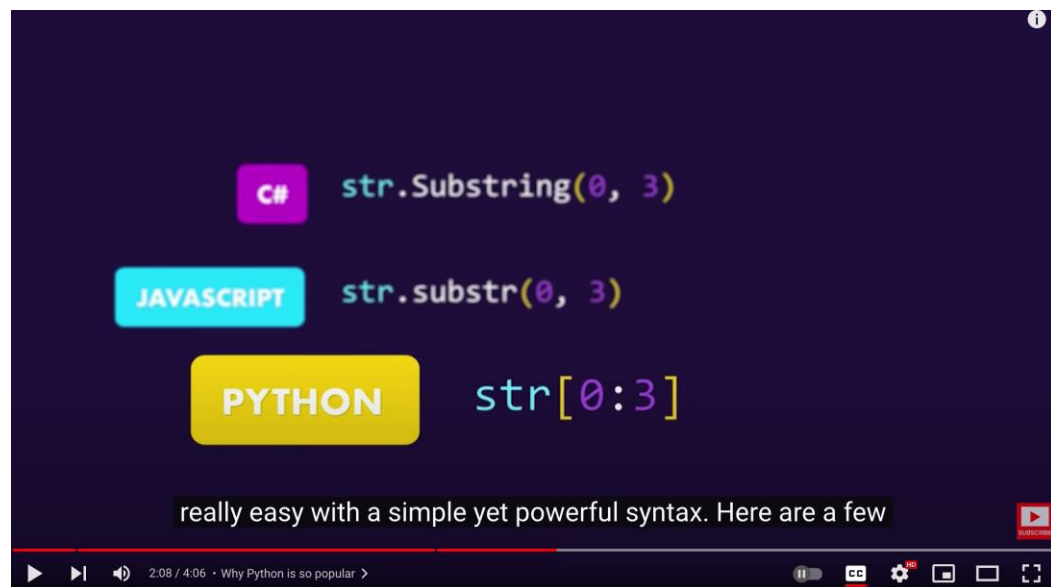


Image Source: <https://www.amazon.com/>

Python as a Programming Language



“What is Python? Why Python is So Popular?”

→ To make your life easier

See: <https://www.youtube.com/watch?v=Y8Tko2YC5hA>

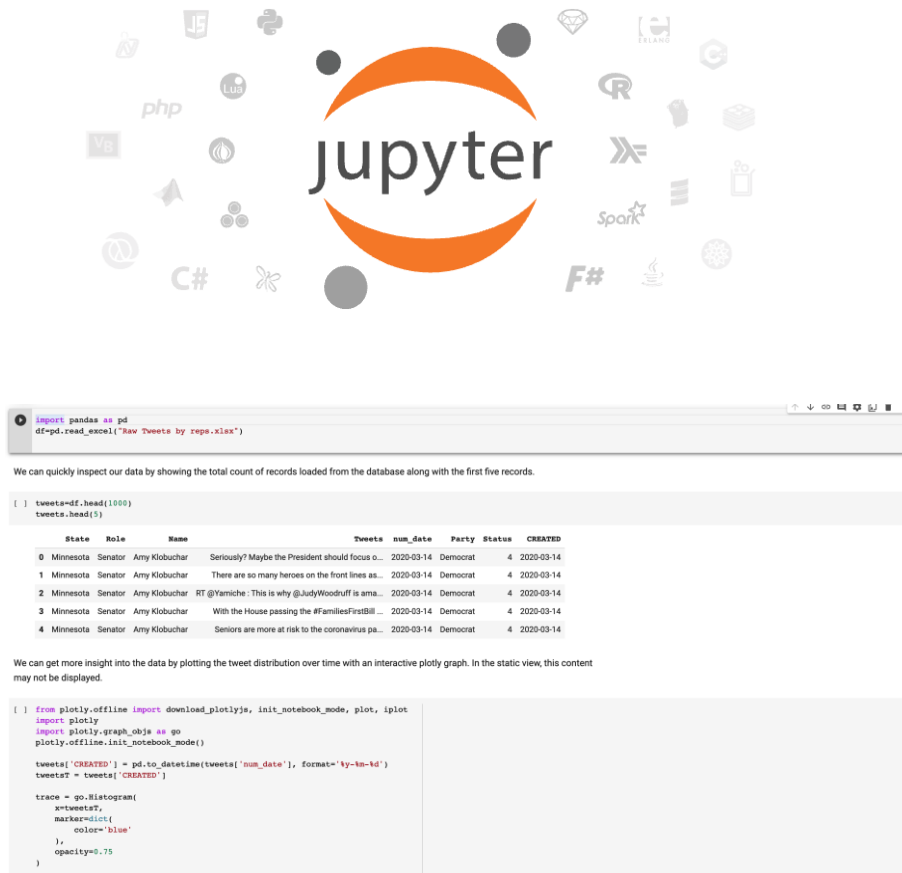
Python as a Programming Language

- Elements of Python
 - Vocabularies/words: Variables and reserved words
 - Sentence structure: Valid syntax patterns
 - Logic structure: Constructing a program to accomplish a certain task
- Case-sensitive programming language
 - **Sum** is different from **sum**

Integrated Development Environment (IDE)

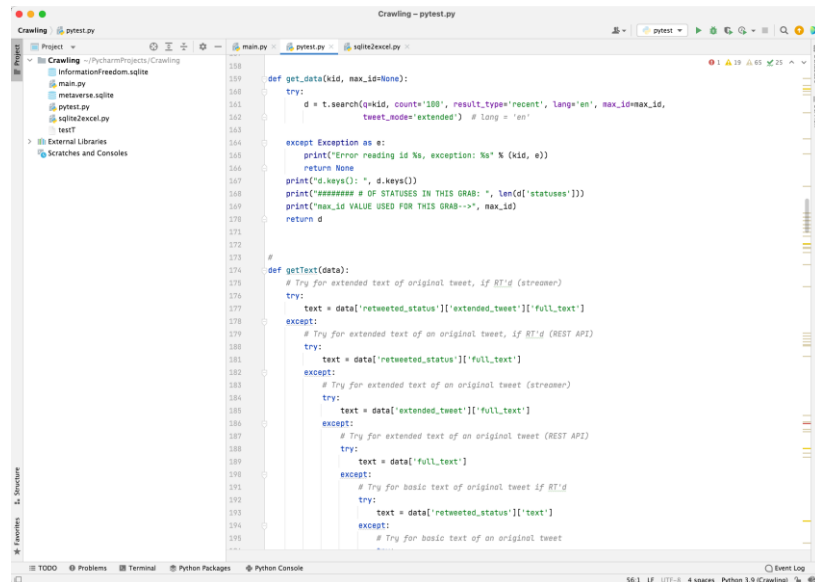
- An environment that consolidates tools that are commonly used in programming
- IDEs are used for developing applications, websites, and other software projects.
- Including code editors, syntax highlighting tools, debugging tools etc..
- Common IDEs
 - Jupyter Notebook
 - PyCharm

Jupyter Notebook



- Open-source web application
- Primarily designed for data science and machine learning workflows
- Helps us create documents that combine live code, equations, visualizations, and narrative text
- Widely used for data analysis, data visualization, scientific research, machine learning, and educational purposes, especially in contexts where real-time execution of code and visualization of results is beneficial

PyCharm



- A wide range of essential tools that are tightly integrated within PyCharm
- A convenient environment for productive Python, web, and data science development
- Ability to handle data science tasks (especially the professional version with its Jupyter integration)
- Primarily for Python software development, including web development, application development etc.

Tools



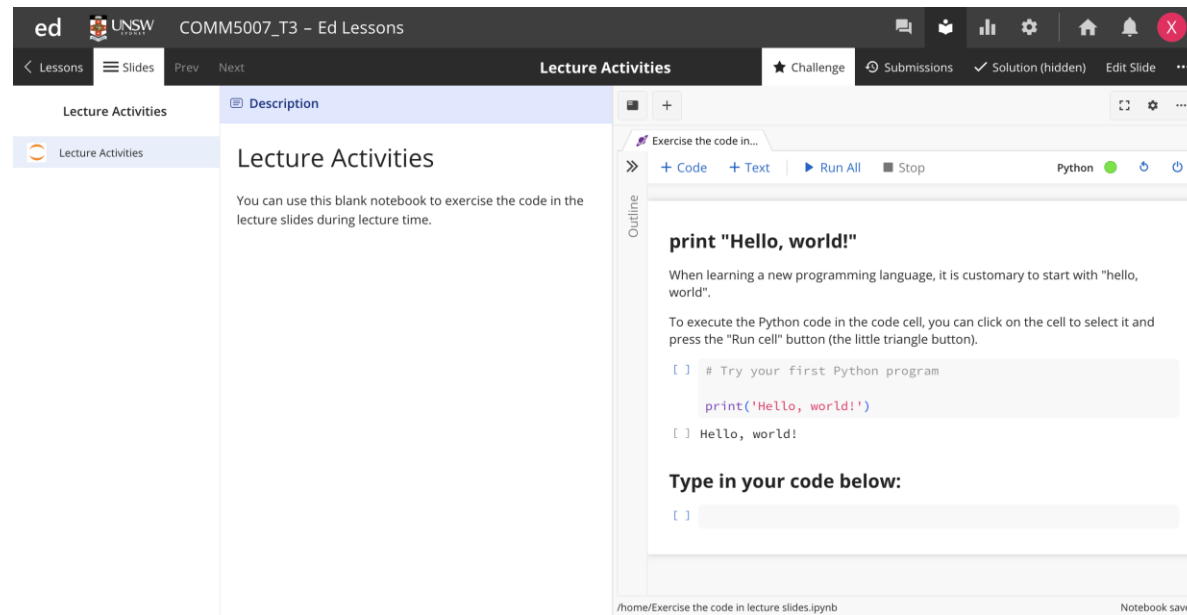
Google Colab

Outline

- Introduction to the course
- Python: A Programming Language
- **Python Fundamentals Part 1**

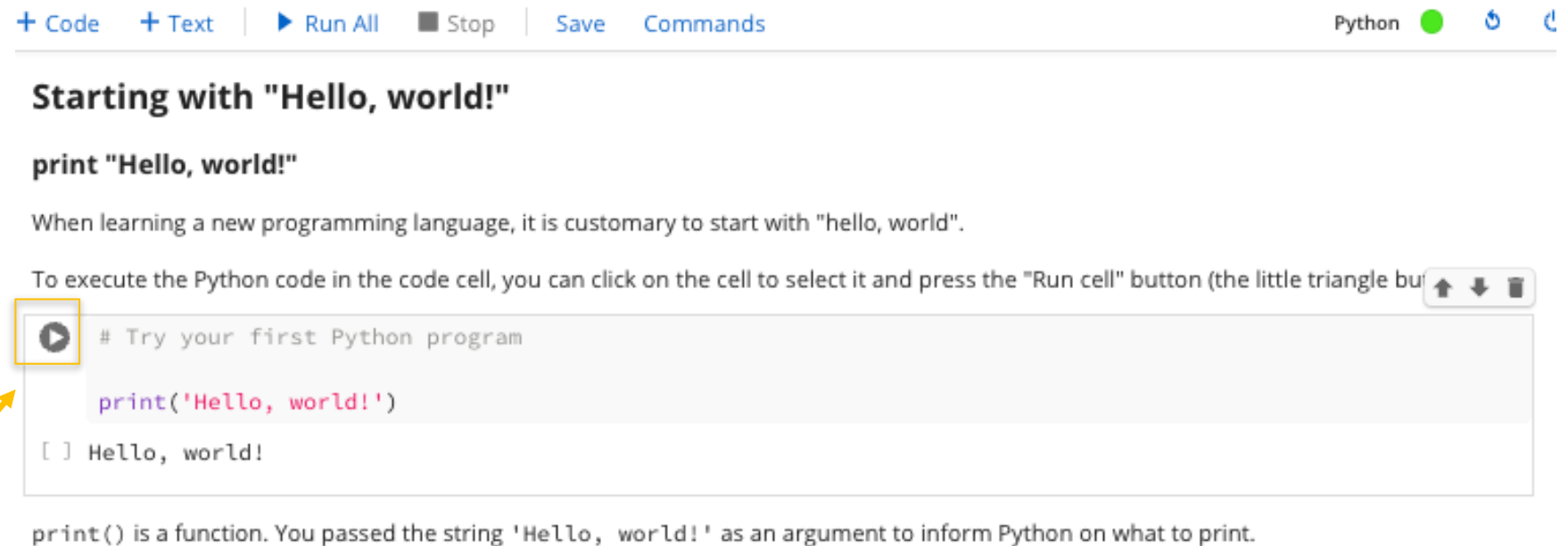
To Start

- Go to Moodle “Week 0 - Pre-course tasks”, and go to Ed Lessons
- Use the blank notebook “Lecture Activities” to practice coding during the lecture



To Start

To run the code



- 1) Clicking on the cell to select it and
- 2) Press Shift + Enter or click the button  to run your code

Hello, world

- When learning a new programming language, it is customary to begin with an “Hello, world!” example
- Say “Hello, world!” in Python
 - **Open** your notebook and **type** in **the code below** into a code cell of your notebook.
 - **Execute** the Python code in the code cell, by 1) **clicking** on the cell to select it and 2) press **Shift + Enter**
 - **Python Code:**

```
# Run your 1st Python code  
print('Hello, world!')
```

Python version

- Make sure your python is updated

```
# Check the Python Version  
import sys  
print(sys.version)
```

```
3.11.3 (main, Jun 5 2023, 09:32:32) [GCC 13.1.1 20230429]
```

Write Comments

- Anything (e.g., words, code) after `#` is considered a **comment** which will be ignored
 - As our code gets bigger and more complicated, they get more difficult to read
 - It is a good idea to add notes in the form of comments to our code
- How many comments are there in the example below?

✓
0s



```
# Practice on writing comments  
print('Hello, world!') # This line prints a string  
# print('Hi!')
```

☞ Hello, world!

Whitespace and Indentation

- **Whitespace:** the spaces or gaps you see in the text to **improve code readability**
 - side-to-side (like spaces between words)
 - Up-to-down (vertical space)
- Forms of whitespace
 - Spaces: Represented by the space bar.
 - Tabs: Generated by the Tab key.
 - Newlines: Generated by the Enter or Return key.

Whitespace and Indentation

- Indentation refers to the **spaces or tabs** used at the beginning of a line of code.
- In Python, unlike many other programming languages, indentation is not just for readability; it defines **the scope of blocks of code**.
- If you don't maintain consistent indentation, Python will raise an `IndentationError`.

```
x = 5
if x > 0:
    print("x is positive")    #this line has indentation
    print("This line is also part of the if block")
print("This line is not part of the if block")
#outside if statement block
```

Data Value

- A data **value** refers to a basic unit of information, such as a number or string, that can be stored and manipulated by a program.
 - Data values belong to **different data types**, e.g., *numeric values and string values*.
 - The **print()** function can print numeric values and string values out.

- **Numeric** value, e.g., 10, 12, 0.3
To print out a number, you can use

```
print(10)
```

- **String** value, e.g., 'good', 'bad'
To print out a string, you can use single quotes (') or double quotes (")

```
print('good')
```

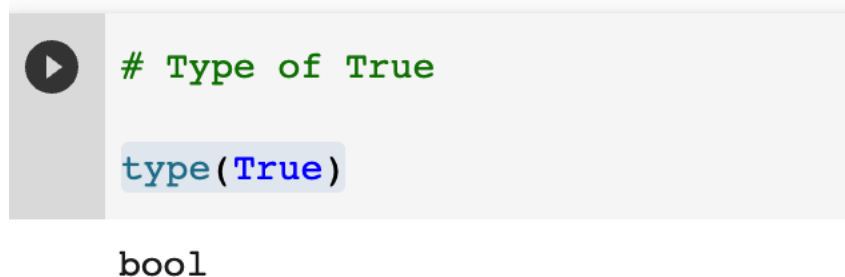
```
print("good")
```

Built-in Data Types: integer, float and string

Data Type	Expression
integer (int)	10, 11, 12, 13
float (float)	0.3, 0.5, 1.33, 1.56
string (str)	'good', 'it is a banana', 'it is a &*'

Built-in Data Type: Boolean

- **Boolean** is another important built-in data type in Python. An object of type Boolean can take on one of two values: **True** or **False**
 - Notice that the value **True** has an uppercase "T"
 - False must use the uppercase "F"
- To display the type of a Boolean object
 - `type(True)` will show bool which stands for Boolean:

A screenshot of a Python REPL (Read-Eval-Print Loop) window. It shows a comment line `# Type of True` in green, followed by the command `type(True)` in blue. The output of the command, `bool`, is displayed below the command line.

```
# Type of True
type(True)
bool
```

Type() function

- To check the data type of a value, you can use **type()** function, e.g.,

`type(10)`

`type(0.3)`

`type('good')`



Type Conversion (1/2)

Type conversion (or typecasting) means transfer of data from one data type to another.

- Numeric value conversion

```
x = 1
print(float(x)) #Cast the integer 1 to a float 1.0
y = 1.1
print(int(y)) #Cast the float 1.1 to 1, we lose some information here.
```

- String conversion

```
z = "123" # z contains three numeric characters. We can convert it to int.
print(int(z))
z = "123z" # z contains a non-numeric character; hence we get an error.
print(int(z))
```

Type Conversion (2/2)

- String conversion

```
# Convert 1 to str 1  
str(1)
```

- Boolean conversion

```
#Convert 1 to Boolean True  
bool(1)
```


Variables

- Variables are containers for storing data values
- A **variable** is a named place in the memory to store a value
- Variables can be used to store data of different types, such as integers, floats, strings, and so on

Naming Convention Summary

- Variable names should be descriptive and meaningful and should indicate the purpose of the variable (e.g., student_name instead of x)
- A variable name should not contain other characters other than letters, numbers, and underscore
 - Variable names should not begin with a number.
 - A variable's name can begin with a letter or an underscore _
 - Case sensitive
 - Variable names shouldn't be the same as Python **reserved keywords or built-in functions**

Statements

- An assignment statement creates new variables and gives them values
 - The programmer chooses **the name** of a variable
 - `x = 15` is an assignment statement, assigning an integer 15 to a new variable called x
 - The programmer may change the value of a variable, e.g.,

```
x = 15 # assigning an integer 15 to a new variable called x
x = 10 # changing the value to 10
print(x)
```
- A **statement** is a unit of code that the Python interpreter can execute. We have seen two kinds of statements:
 - print statement
 - assignment statement

```
i = 0          #This is a simple assignment statement
print(i)       #This is a print statement
```

Assignment Statement

- In most cases, an **assignment statement** consists of an expression on the right side and a variable on the left side

```
y = 5
```

```
y = 2 + 3
```

- One assignment statement can be used to assign values to multiple variables

```
x, y = 1, 2
```

```
print(x)
```

```
print(y)
```

```
print(x, y)
```

Assignment

- If we save a value to an existing variable, the new value will **overwrite** the previous value
- What will be the printed with the code below?

```
x, y = 1, 2  
print(x)  
print(y)  
x = 3  
print(x)  
print(y)  
print(x, y)
```



Exercise

- Type below the code in your Jupyter Notebook

```
Wallet = 20
print(Wallet)
Wallet = 8
print(Wallet)
```
- Define a variable called day and set it to “Monday”

Reserved Words

- Python has reserved words. We can't use a reserved word as a variable name
- Important reserved words:
 - **if, else, and, or, not, del, def**
 - **print, type, len, int, float, str, bool**

```
print('Hello, world!') # print is a reserved word to print
```

```
print = 10 # this statement is problematic as print is a reserved word
```

Errors

- Read error message carefully to correct a mistake in your code

```
[ ] # Print string as error message
```

```
pprint("Hello, world!")
```

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-1-574e064069e8> in <module>()  
      1 # Print string as error message  
      2  
----> 3 pprint("Hello, world!")
```

```
NameError: name 'pprint' is not defined
```