Python workshop

Peiyi Zhou

November 12, 2025

University College London

Table of contents

Getting Started...

Launch Python

We will run our python codes in

- (either) Anaconda Navigator - Jupyter Notebook.

Figure 1: Anaconda logo

- (or) Visual Studio Code (VS Code).

Figure 2: Visual Studio Code logo

Download Anaconda Navigator or VS code

For downloading Anaconda Navigator, follow instructions on *https://www.anaconda.com/download*.

For downloading VS Code, visit *https://code.visualstudio.com/*.

The way for implementing the Notebook file is the same for both, and for the presentation we shall apply VS code as the example.

Python basics

Arithmetic in Python

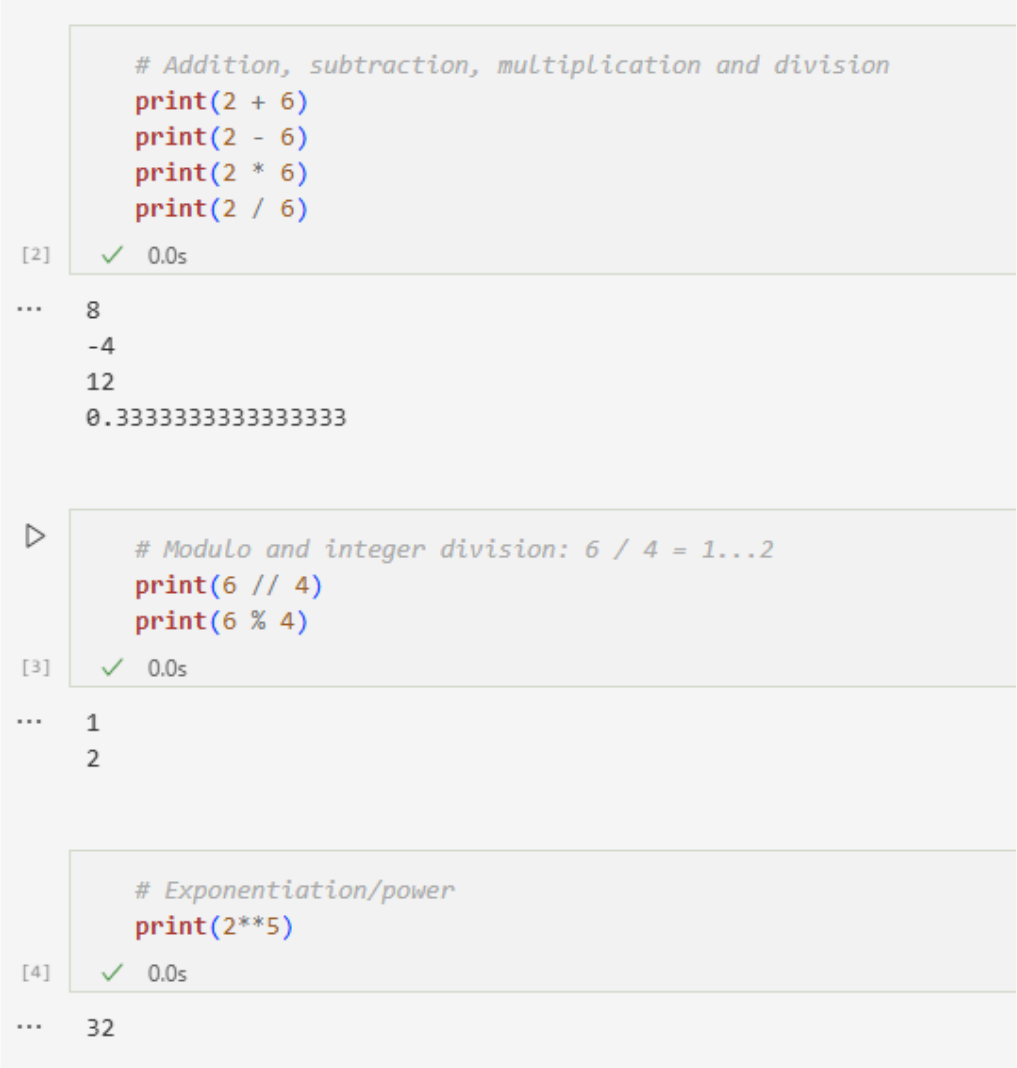

Figure 3: Arithmetic in Python.

Exercise: How do you check whether 17 is a factor of 33677?

[Hint: try to apply the modulo operator %]

Importing modules

Although we simply carry addition, subtraction, etc. in Python without any further steps, we might face errors when we try to run operations like square root...

```
[6]  # Implementing square root directly
     sqrt(3)
     0.4s

...  ---------------------------------------------------------------------------
     NameError                                 Traceback (most recent call last)
     Cell In[6], line 2
           1 # Implementing square root directly
     ----> 2 sqrt(3)

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

However, Python indeed provides a way for carrying such operations directly - the only thing we need is to import related modules containing in-built functions for such operations.

Though there are more than one modules that can do this, here I would focus on *NumPy* module (short for Numerical Python, abbreviated as *np*). This module is very general and commonly used in various of tasks.

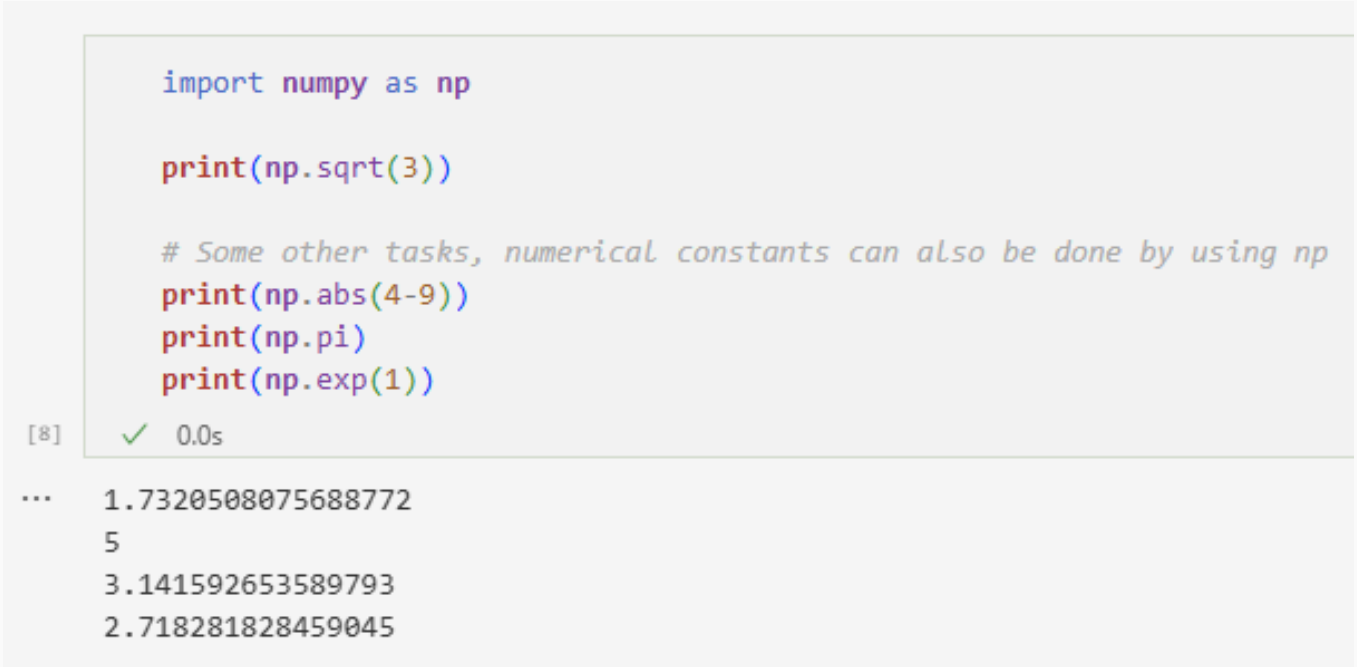

Figure 4: Import *NumPy* Module and use its in-built functions.

Exercise: What happens if you try to compute $\csc(\pi/6)$? $\operatorname{sech}(2)$? What can you do about this?

Python variable data types and data structures

Variable data types

Variables can store data of different types, and different types can do different things.

Common data types for variables in Python include:

- `float`: float numbers as decimals.
- `int`: integer.
- `complex`: complex numbers, eg. $1 + 2j$, where j is the imaginary part used in Python. We can use `real` and `imag` to identify the real and imaginary part respectively.
- `bool`: Boolean: TRUE or FALSE.

The data type for variables can be examined by using *type*.

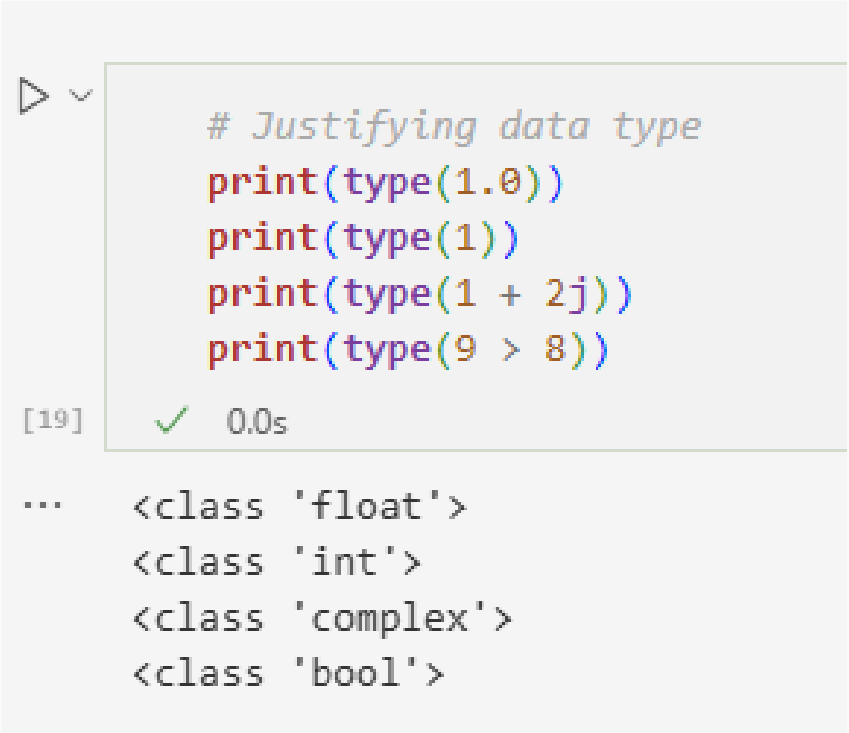

Figure 5: Python data types for variables.

Data Structures

We often treat a number or a Boolean as a single unit of data.

When several such units combine together, such collection of data is called a **data structure**. Examples are strings, lists, tuples, sets, dictionaries, vectors, and matrices. Simply speaking:

- Strings can be seen as an ordered sequence of characters.
- Lists can be seen as an ordered sequence of anything.
- Tuples behave nearly the same with lists, but with some minor differences in some aspects like appending.

Strings

We will see what can Python do on strings:

- Join multiple strings.
- Count the length of the string.
- Indexing the string.
- Split and join the string.
- Replace characters.
- Formatting: use f-string.

Lists

We will discuss the following.

- Define lists.
- Join multiple lists.
- List indexing.
- List appending.
- List removal.

Tuples

Tuples are nearly same as lists but defined using the round brackets instead of square brackets.

Exercise: can you construct a tuple and a list both with entries: 1, 'two', 3, 'vier', 'five'?

The difference between lists and tuples is the way of appending: list appending uses *.append*, while tuple uses '+' directly (meaning to add another tuple - slightly different compared with appending).

Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.

Selection and Iteration in Python

Python selection

The syntax for selection in python is again related of using *`if...else`*. Let's see a toy example first.

```
    mercedes_point = 398
    redbull_point = 366

    if mercedes_point >= redbull_point:
        print('Mercedes is now at the second place of WCC.')
    else:
        print('Redbull racing (Verstappen racing) is now at the second place of WCC. ')
```

[45] ✓ 0.0s

```
Mercedes is now at the second place of WCC.
```

Figure 6: Python selection example related to F1 2025 (up to San Paulo GP).

Exercise: Could you write a Python selection used for giving weather suggestions? [Hint: use f-string format]

- If it is a sunny day, then print 'You might want to hike today as it is (weather).'
- If it is a cloudy day. then print 'Meh - at least a (weather) day is better than a rainy day.'
- Else, print 'Probably it is not a good option do go outdoors in such a (weather) day.'

Python iteration: *for* loop

There are two ways in iteration in Python: *while* loop and *for* loop.

The *for* loop restricts the number of iterations. See the syntax of using *for* loop.

```
    # for loop syntax
    for i in range(10):
        print('We are now in Iteration', i)
[46] ✓ 0.0s

··· We are now in Iteration 0
    We are now in Iteration 1
    We are now in Iteration 2
    We are now in Iteration 3
    We are now in Iteration 4
    We are now in Iteration 5
    We are now in Iteration 6
    We are now in Iteration 7
    We are now in Iteration 8
    We are now in Iteration 9
```

Figure 7: Syntax for applying *for* loop.

Python iteration: `while` loop

We again also introduce the ***while*** loop, which needs a stopping criteria otherwise the loop will never terminate.

```
# while loop syntax
i = 0  # initialise

while i < 10:
    print('We are now in Iteration', i)
    i += 1
```

[48] ✓ 0.0s

```
We are now in Iteration 0
We are now in Iteration 1
We are now in Iteration 2
We are now in Iteration 3
We are now in Iteration 4
We are now in Iteration 5
We are now in Iteration 6
We are now in Iteration 7
We are now in Iteration 8
We are now in Iteration 9
```

Figure 8: Syntax for applying ***while*** loop, achieving same effect as ***for*** loop.

Exercise: Given the famous Newton-Raphson Method for finding the root of $f(x)$ via the recursion of

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

given an appropriate initial guess x_0 to result convergence. Could you solve the real root of $x^3 + 2x - 2 = 0$? (I give you the fact that $f(x) = x^3 + 2x - 2$ has one real root and two conjugate complex roots.)

[Hint: use `for` loop for a large number of iteration, or use `while` loop by setting a certain stopping criteria. You may also wish to compare the efficiency of both methods.]

Python functions

Python function syntax

The syntax for defining a function in python is summarised as the following:

Figure 9: Syntax of defining a Python function.

And we will give a toy example (again related to F1).

```
def F1_WDC(year):
    """
    Check the WDC of F1 for the most recent 10 years.
    """

    if year < 2015:
        print('Our searching year should be no earlier than 2015.')
    elif year in set({2015, 2017, 2018, 2019, 2020}):
        print(f'F1 WDC in {year} is Lewis Hamilton.')
    elif year in set({2021, 2022, 2023, 2024}):
        print(f'F1 WDC in {year} is Max Verstappen.')
    else:
        print(f'F1 WDC in {year} is Nico Rosberg.')

F1_WDC(2016)
```

[22] ✓ 0.0s

```
F1 WDC in 2016 is Nico Rosberg.
```

Figure 10: A toy example of Python function.

Exercise: Write the previous Newton-Raphson Method as a function that can be applied for any function f.

Thank you for your listening!