Week 1: Introduction

2023-08-30

Housekeeping: updates on tutorial presentations

- Preparation
  - Presentations will be on Wednesday for the topic covered in Monday
  - Does not leave you too much time
  - Efforts will be made to upload course materials in advance so that you can refer to them in building your tutorials
- Three presentations in total (approximately 15min)
  - Either two from first half + one from second half
  - Or one from the first half + two from the second half
- Those who are experienced are encouraged to fill up the first couple of weeks
- The sign-up sheet should work now

Using Python

- Python can be downloaded and used in Windows, macOS, Linux, etc.
- Download at https://www.python.org/

Anaconda

- Automatically installs Python and widely-used libraries
- Makes Python library/package management easy
- Used in all operating systems

Programming Environments (or IDEs)

- Visual Studio Code
- Spyder
- PyCharm
- Jupyter Notebook
  - Browser-based
  - Needs to installed locally

Google Colaboratory

- Browser-based
    - No need to be installed
    - Scripts can be accessed from any device with an internet connection
- Free access to cloud-based resources
    - CPU and RAM
    - GPU/TPU (good for deep learning tasks)

Google Colaboratory (cont'd)

- Lots of libraries/packages (including almost everything we will use) pre-installed
  - We can also install new libraries/packages in line
- Integrate code & text
- Progress automatically saved

Google Colaboratory (cont'd)

- However, if a runtime disconnects, you lose what you have on memory
- When conducting time-consuming, heave data work, save what you have on memory
- Let's check this out quickly *here*

Computers

- Perform very fast calculations
- Remember results
- Python is a programming language
- Python interpreter translates that into machine code
- Computer “understands” machine code

Python Programs

- Programming refers to the broader process of creating computer programs
- Programs are sets of instructions that tell a computer how to perform specific tasks
- Programming can be implemented in
    - Text file (.py),
    - Notebook file (.ipynb),
    - Directly in a shell (run codes/scripts in a command-line interface, such as Terminal in MacOS)

Let’s start coding

```python
print('Hello, world!')
```

```
Hello, world!
```

Objects

- Python creates and manipulates **data objects**
- Objects have a **type**, which defines how they can be used
- Objects are:
    - Scalar (cannot be subdivided)
    - Non-scalar (e.g., list, tuple, dictionary, set)

Scalar Objects

- `int` - integers, e.g. `3` or `142`
- `float` - real numbers, e.g. `4.2` or `-3.5`
- `bool` - Boolean, also known as *logical* in some languages, `True` or `False`
- `NoneType` - a special type with one possible value, `None`

Scalar Objects

- Use `type()` to get the type of an object

```
type(3)
```

```
int
```

```
type(4.2)
```

```
float
```

```
type(False)
```

```
bool
```

```
type(None)
```

```
NoneType
```

Is `str` scalar or not?

- A sequences of characters
- Typically used to represent text
- It depends on the context

```
name = "Mia"
print(name)
print(name[0])
print(name[2])
```

```
Mia
M
a
```

Type Conversions (cast)

- Objects of different types can be converted to one another
- Be careful, though

```
int(3.9) # rounds the float to the smaller integer
```

```
3
```

```
float(4.0) # integer 4 becomes float 4.0
```

```
4.0
```

Print

- Use `print()` to show output in the console
- Otherwise, the output will only be available during interaction

```
2 * 2 # will output 4 only during interactive use
```

```
4
```

```
print(2 * 2) # will print 4 to end user in the console
```

```
4
```

Expressions

- **objects** and **operators** are combined to form **expressions**
- The expressions evaluate to a value
- Syntax for a simple expression:

```
<object> <operator> <object>
```

Operators for `int` and `float`

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Example

- Division will result in a float

```python
type(6 / 5)
```

```
float
```

- Addition, subtraction, multiplication may result in integer

```python
type(10 - 22)
```

```
int
```

- Or float if one of the objects is a float

```python
type(10 * 7.6)
```

```
float
```

Arithmetic Operations

```
2 + 3
```

```
5
```

```
4 - 7
```

```
-3
```

```
10 * 4
```

```
40
```

```
100 / 30
```

```
3.3333333333333335
```

Arithmetic Operations

# is used for comments

```
100 // 30 # floor division
```

```
3
```

```
100 % 30 # modulus (remainder)
```

```
10
```

```
7 ** 2 # exponentiation
```

```
49
```

Operator Precedence

Parentheses tell Python to do those operations first

- ()
- **
- *, /, //, %
- +, -

Binding Variables and Values

- Equal sign (=) is used for **assignment** of a value to a variable

```
pi = 3.14159
radius = 5
```

```
print(pi)
print(radius)
```

```
3.14159
5
```

Variables and Values

- Retrieve value by invoking the name of the variable

```
area = pi * (radius ** 2) # parentheses redundant here
                          # they help read code though
print(area)
```

```
78.53975
```

```
circumference = 2 * pi * radius
print(circumference)
```

```
31.4159
```

Variables and Values

- Why assign values to variables?
- Easier to read and debug
- If value changes, just change in one place

```
radius = 10
area = pi * (radius ** 2)
circumference = 2 * pi * radius
print(circumference)
print(area)
```

```
62.8318
314.159
```

Equality Sign in Programming vs. Math

- Remember, a single = sign means assignment, not equality

```
radius = radius + 1 # is totally acceptable
```

```
radius += 1 # is also equivalent (increase radius by 1)
print(radius)
```

```
12
```

Rebinding

- One can rebind a variable name to a different value

```
radius = 10
pi = 3.14
circumference = 2 * pi * radius
print(circumference)
```

```
62.800000000000004
```

Rebinding

```
radius = 15
print(circumference) # still same
```

```
62.800000000000004
```

Rebinding

```
circumference = 2 * pi * radius
print(circumference) # now changed
```

```
94.2
```