Python, the Basics

Kaihang Zhang (khzhang2@cconnect.hku.hk)
208 Haking Wong Building, The University of Hong Kong
December 7, 2022

TABLE OF CONTENTS

PRELIMINARY

Many contents of this tutorial are adopted from the official Python Tutorial (https://docs.python.org/3/tutorial/). In this instructional file, only the most basic elements are mentioned. For more advanced knowledge, please check the official tutorial as well as the documentation (https://docs.python.org/3/index.html).

Many of the scientific computing parts in this course are conducted with Numpy and Pandas, which are two powerful and popular scientific computing packages. Readers can refer to their official documentations (https://numpy.org/doc/stable/ and https://pandas.pydata.org/docs/user_guide/10min.html#min) and tutorial instructions (https://numpy.org/devdocs/user/quickstart.html). The machine learning related problems will be conducted by the package PyTorch (https://pytorch.org/). To install PyTorch, you can find instructions via this link (https://zhuanlan.zhihu.com/p/412838545), or you can use a free cloud computing platform (with PyTorch pre-installed) called Colab (https://colab.research.google.com/). To learn the basics of PyTorch, you can check this link (https://pytorch.org/tutorials/) but leaning all the programming skills by practical examples are always suggested.

This instructional file also contains a supplementary Jupyter Notebook file. Readers can access this notebook via this link:

https://github.com/hku-kejintao/HKU-CIVL7018-Data-Science-for-Civil-Engineering/blob/main/Python%20tutorial/Python%2C%20the%20Basics.ipynb

DATA TYPES

Data

- `int` For storing integers, such as `1`, `2`, `10`.
- `float` For storing floating numbers, appeared as decimal numbers.
- `str` For storing text-like data, such as "`Hello world`".
- `bool` For storing logic values, `True` and `False`, or `1` and `0`.

Data Structures

- `list` A group of many data, each of which is called an element. The data type of each element is not necessarily to be the same. Square parentheses are used to create a list. Examples are shown below.

  - `[1, 2, 3]`
  - `["1", "2", "3"]`
  - `[1, 2, "3"]` ← Elements can have different data types, here they are `int` and `string`
- `dict` Dictionary. Examples are shown below.

```
{
    "key1": value1,
    "key2": value2
}
```

- `array` Matrix data, a type in package Numpy
- `DataFrame` A data type in package Pandas.
- `tensor` A data type in PyTorch, similar to `array`

WRITING A LOOP

There are two types of loops in Python, namely "`for`" loop and "`while`" loop. See the Jupyter Notebook for detailed instructions.

"for" loop

```
max_num_iter = 5  # the maximum number of iteration

# "i" is a temporary variable to use in each iteration
# "range" is a build-in function to generate iterable values
for i in range(max_num_iter):
    print(i)
# Note: the index number in Python starts from 0, not 1!
```

```
0
1
2
3
4
```

"while" loop

```
# in "while" loop, an iterable variable must be pre-specified
c = 0
while c<5:
    print(c)
    c = c + 1

# Personally, "for" loop is more used than "while" loop
```

```
0
1
2
3
4
```

Figure 1. "`for`" loop (left hand side) and "`while`" loop (right hand side)

OPERATIONS: METHODS AND FUNCTIONS

Methods usually apply to objects (e.g., a `list`), without values returned, and no value assignment needed. However, there are returned values from a function. Therefore, one should not forget to assign the output value after applying a function. Typical methods: `append` (`list` operation), `extend` (`list` operation). Typical functions: `print`, `sum`, `shape` (for `array`).

One can also define functions by herself, using "**`def function_name`**`(inputs)`".

Define_function

```
def my_function(input1, input2):
    # write your funciton details here
    # consider both inputs are numerical values (int, float)
    output = input1 + input2
    return output
```

```
my_function(1, 2)
```

```
3
```

Figure 2. Example code for defining a function