04_BasicPython

September 12, 2021

0.1 1. Numbers

0.1.1 1.1 Integer

An integer is a whole number such as:1, 2, -5, 0

In the cell below, try (type):

```
1 + 1
10 - 1
```

Python will run the first line which yields 2 and it will run the second line which yields 9. The result below the cell will be the one from the last line only.

[ ]:
```
10 - 1
```

[ ]:
```
9
```

[ ]:
```
2**3
```

[ ]:
```
8
```

[ ]:
```
a = 555
```

0.2 Basic Python for Data Science - 1

0.2.1 Basic Data Types

In the field of data science, the two most popular programming languages are **R** and **Python**. This course, we will use Python. Python is a very intuitive programming language. Many schools in Thailand start teaching Python in the primary school level. For our course, we assume that the students **have no prior knowledge in Python programming**. A very basic Python programming concepts will be covered so that students can use Python to test/practice with others tools in data science. Students will gain more knowledge about Python along the way throughout the term.

Note: The focus of this course is not about Python programming. Therefore, the code aims to be clean and concise to help us understand basic concepts in data science, not for the best practice in Python programming or for the fastest execution.

Here we will learn the following to pics

1. Number

[ ]:
```
1 + 1
10 - 1
# will see only the last output
```

[ ]:
```
9
```

[ ]:
```
# print each output
print(1+1)
print(10-1)
```

```
2
9
```

0.2.2 1.2 Float

Float is a number with a decimal point such as: 0.5, -2.5, 3.14, 0.0

[ ]:
```
5.0 * 0.5
```

[ ]:
```
2.5
```

When we divide two integer numbers, we will get a float as a result.

[ ]:
```
# int / int
1/1
```

[ ]:
```
1.0
```

[ ]:
```
# int / int
1/2
```

[ ]:
```
0.5
```

0.3 2. Variables & the assignment operator

2.1 Creating a variable
When we have a value and we want to use this value in other places. Typing the value every time

Notice that the = symbol is ***not*** really an equal sign. This is called **an assignment operator**.
This will take value from the right side and then assign to the left (aka, put in the box).

[ ]:
```
my_gpa = 2.015
your_gpa = 3.0918
average_gpa = (my_gpa + your_gpa)/2
```

At this point, you may think nothing happens since you don't see any output. Actually, we just created 3 variables. You can view them (and their values) by clicking the icon **Show variables active in jupyter kernel** (the one that looks like a table)

Naming variables. Python has rules regading the way we can name our varibles. The main rules are:

(1) Start with a character

(2) Contain no space or any special symbol

There are more rules that this. I am trying to be a minimalist here. :)

[ ]:
```
x = 10
Pi = 3.14
my_salary = 18000
mySalary = 24000
```

If we violate Python rules, we will get error messages.

[ ]:
```
my salary = 18000
2months = 36000
```

Comments , we can create a comment by using #. Any code after # to the end of the line will be ignored by Python.

[ ]:
```
# This line begins with #. Thefore, Python will not run this line.
# We can write anything. It won't create any error.
# We normally use this to write COMMENTS.

# We use comments to describe or provide additional context to whomever reading␣
 ↪this code.
# Most likely, that person will be you (in the near future).

x = 5
x = x + 1

# The code above IS NOT a math equation!!!
# It is
#          variable <---  SOMETHING
#
# Python will do that SOMETHING first, then whatever that value is, it will␣
 ↪assign
# that value to the variable on the left
# Let's see the value of x
```

```
x  # Python will run the code before the first # sign. However, these two␣
↪sentences will be ignored.
```

0.4 3. String

A string is a series of characters.

0.4.1 3.1 Creating a string

We can create a string (text) using single quotes or doulbe quotes.

[ ]:
```
# using single quotes
'This is MUIC'
```

[ ]:
```
# using double quotes
"This is also MUIC."
```

The may choose to use ” if we plan to have ’ inside the string, and vice versa.

[ ]:
```
# Having a single quote inside double quotes
"It's me."
```

[ ]:
```
# Having a double quote inside single quotes

'Hello "DUDE" ...'
```

If you want to create a string but forget to put them inside ’ ’ or ” ”, we will get an error message.

[ ]:
```
Let's pretend to forget the quotes.
```

0.4.2 3.2 Mixing string & numbers

[ ]:
```
x = 3
"There are {spot_one} cars!".format(spot_one=x)
```

[ ]:
```
b = 3
c = 4
"I have {spot_one} brothers and {spot_two} sisters.".format(spot_one = b,␣
↪spot_two = c)
```

[ ]:
```
d = 2
unit = "cars"
"He has {spot_one} {spot_two}.".format(spot_one = d, spot_two = unit)
```

0.4.3 Using f-string

f-string as a newer way of formatting strings. This looks a bit strange to have a letter (f or F) in front of our string. This letter is a signal to tell python that during run time, change the placeholder to the real value.

[ ]:
```
"This line is just a string. It will never change {2*3}"
```

Now, we will create an f-string by adding a letter f in front of the string.

[ ]:
```
f"A part inside this string will be changed during runtime. {2*3}"
```

We can also use with variables. Try:

```
x = 3
unit = "cars"
sentence = f"Tom has {x} {unit}."
sentence
```

[ ]:
```
x = 3
unit = "cars"
sentence = f"Tom has {x} {unit}."
sentence
```