Lesson Three

Math 6080 (for the Masters Teaching Program), Summer 2020

Part 1. Variables in Python A variable in python comes into existence when it is set to some value with the equals sign (=). It learns what it is from the type of the thing it is set to. Thus, for example,

$$x = 5$$

generates the variable x, assigns it the value 5 and tells it that it is an integer.

$$y = 5.$$

does the same thing for the variable y, but y is told to be a real number. Thus, after writing the above lines, if you enter x, then Python responds with 5, and if you enter y, then Python responds with 5.0.

A variable can also be set to a string:

$$z = \text{'five'}$$

generates the variable z and tells it that is a string. Thus, for instance, if you type in z, then Python responds with 'five', and if you type in z[0], then Python responds with 'f'.

There are two simple rules for what names you may use for variables:

- a variable must consist only of letters, numbers and underscores (_).
- a variable must **start** with a letter or underscore.

Thus,

variable var_iable, variable1, variable_1, _variable_1

are all legitimate names for a variable, but 1variable is not.

You may assign multiple variables at the same time. Thus:

$$x, y, z = 5, 5., \text{'five'}$$

simultaneously assigns the variables x, y and z the values (and types) $5, 5.0$ and 'five'. This can be very useful for making Python code fit onto a page more easily.

First Exercise. Assign the variables as above, and try various arithmetic and logical operations with variables standing in for the values, just to convince yourself that the variables do inherit the types. A variable can also be assigned a "Boolean" value (and type), i.e.

$$x, y = \text{True, False}$$

assigns x the value True and y the value False. The operations:

$$x \text{ and } y, \quad x \text{ or } y, \quad \text{not } x, \quad \text{not } y$$

make sense for variables that have been told they are Booleans.

Part 2. If then statements in Python. The basic conditional in Python is

if x: blah

where x is either True or False and the colon stands for "then". When x is True, Python executes blah. When x is False, Python skips over blah. Thus, for example,

if 1 == 1: print('Aha!')

results in Python printing: Aha!

Typography Caution. There are two extremely important details to notice here:

- "if x" must be followed by a colon.
- blah can be entered in the following line, but if so it **must be indented,** i.e. it must commence to the right of the (if) above. Thus, entering:

```
if 1 < 2:
print('duh')
```

will earn you an error message, but entering

```
if 1 < 2:
   print('duh')
   print ('really duh')
```

will earn you a Python output of duh, followed on the next line by really duh.

Remark. You can give Python multiple command lines after an "if x:" but they have to all have the **same** indentation. Python is cranky about indentations. We will almost always be entering these compound Python commands via files. When you enter an "if x:" statement manually, Python prompts for the next line with three dots. You need to try this out to get the hang of it.

Variation. Python has an "elif y:" option which can be used after "if x:" to take care of multiple contingencies (elif stands for "else if"), with "else:" taking care of all the remaining contingencies. For example:

```
x = 2
if x < 1:
   print('x is smaller than 1')
elif x < 2:
   print('x is smaller than 2')
elif x < 3:
   print('x is smaller than 3')
else:
   print('x is not smaller than 3')
```

is a perfectly good, if boring, bit of Python code. (What does it respond with?)

A Final Printing Remark. Like the assignment of variables, the print command can print multiple things if they are separated by commas. Thus

```
x = 2
print(x, 'is smaller than 3')
```

outputs: 2 is smaller than 3.

Without a comma, the print command above returns an error message!

Second Exercise. Play around with these "if x:" statements. Make intentional mistakes to start to get the hang of what the Python error messages are trying to tell you. This will come in handy, because everyone forgets colon indentations!

Reference:
https://www.w3schools.com/python/ (python variables and python if...else)