ECMM455 Python Worksheet 3: Python 2.7 vs 3.7

Prof Hywel Williams

September 12, 2019

1 Aims

- Look at some of the most obvious differences between Python 2.7 and Python 3.7
  - How numerical division is handled
  - How the *print* command works
  - The *raw_input()* function in Python 2 is renamed to *input()* in Python 3

2 Python 2 vs Python 3

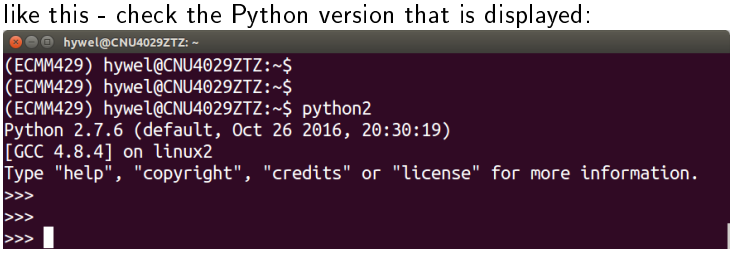
There are two versions of Python in common use: Python 2.7 and Python 3.x . In this course we will use Python 3.7, which ships with the latest version of Anaconda. The programming community is gradually shifting towards Python 3. However, the Code Academy Python tutorial, which we draw on for some of these worksheets, uses Python 2.7 . For this reason, and for general knowledge, it is important to be aware of the main differences you will encounter when switching between Python 2.7 and Python 3. A useful list is given here: *http://sebastianraschka.com/Articles/2014_python_2_3_key_diff.html* . Below are some exercises to help you get used to the differences.

3 Integer division vs float division

Python uses two kinds of variable to store numerical values: *int* and *float* . The *int* data type is used to store integers (whole numbers). The *float* data type is used to store real numbers (i.e. numbers with a fractional part after the decimal place). Most of the time you will not be aware of this distinction and Python will just choose the most appropriate type (primarily based on efficiency - *int* values take up less memory space than *float* values). However, sometimes it will catch you out, especially when switching between Python 2 and Python 3, since the different versions handle division differently - Python 3 corrects a minor oddity in Python 2...

3.1 Exercises

1. From a terminal, open Python 2.7 in interactive mode by typing "python2". You should see something like this - check the Python version that is displayed:



2. Then enter these integer division commands and note the result:

   (a) 1/1

   (b) 2/1

   (c) 1/2

3. Now try these float division commands. (Note that Python treats any value with a decimal point as a float. Also, if any value in a calculation is a float, it treats them all as floats.)

   (a) 1.0/2

   (b) 1/2.0

   (c) 1.0/2.0

4. You can convert from float to int using the int() function. This can be useful if you only want the integer part of a number. Try these commands and note what values are returned:

   (a) int(2.3)

   (b) int(1.0/2)

5. You can convert from int to float using the float() function. This can be useful to force float calculations. Try these commands and note what values are returned:

   (a) float(1/2)

   (b) float(1)/2

   (c) 1/float(2)

6. Why do you think 5a gave the answer it did? Try these commands and think about the order in which Python performs operations:

   (a) 1.0/(2+3)

   (b) (1.0/2 + 1.0/3)

   (c) 2.0/(2*3)

   (d) (2.0/2)*3

   Python always evaluates expressions inside parentheses first, i.e. it starts from the inside of an expression and works outwards. So if you call a function on an expression, it evaluates the expression first, then calls the function - see 5a above. This feature can sometimes be useful - you can nest operations inside each other.

7. Now, from a different terminal, enter interactive mode in Python 3 by typing "python". Then try the following and compare the answers to what was given with Python 2:

   (a) 1/1

   (b) 2/1

   (c) 1/2

   (d) float(1/2)

4 The *print* function

The print command that is commonly used is actually a function. In Python 2, the print function was given special treatment in that you can call it without using parentheses. In Python 3, this feature is removed for consistency and the parentheses must be used. Note that Python 2 will accept use of print with parentheses, but Python 3 will not accept use of print without parentheses.

4.1 Exercises

1. Open a terminal and enter interactive mode with Python 2, as before.
2. Try the following:
   (a) print "hello world"
   (b) print("hello world")
3. Now open a terminal and enter interactive mode with Python 3. Try the commands again. What happens?

5 *raw_input()* renamed to *input()*

...but the function works exactly the same way - see Worksheet 4. In Python 2, there was one function to get user input as a string (*raw_input()*) and another to get user input as an integer (*input()*). In Python 3, *input()* always gets input as a string and it is up to the programmer to fix the data type if needed.