Lecture 1

Ashwini Vaidya

March 13, 2017

0.1 The Python interpreter and the Python program

We have already learnt how to run a python program from the command line. It's possible to start this command line from a terminal, an IDLE or others. However, it's also possible to begin an *interactive* python session, simply by typing 'python'

```
python
Python 2.7.13 |Continuum Analytics, Inc.| (default, Dec 20 2016, 23:05:08)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>>
```

The $\rangle\rangle\rangle$ prompt indicates that you can begin typing a python code, which will be executed immediately.

```
>>> print 'hello world'
hello world
```

If you're unsure about your syntax or about the way a particular piece of logic works, the interactive mode is a great way to test it out. You can test pieces of code extensively before actually writing them into your program. You don't need to type 'print' in the interactive mode, just typing the name of the variable will print its contents. The examples that follow may sometimes use the angled brackets to indicate interactive mode.

Type Ctrl-D to exit on Unix machines or Ctrl-Z to exit from a Windows machine.

0.2 Object types – the basics

There is a difference between strings and integers in Python e.g. try

```
>>> myname='Monty Python'
>>> mynumber=2
>>> myname-mynumber

TypeError: unsupported operand type(s) for -: 'str' and 'int'
```

Python says it cannot subtract an integer from a string and throws up a **TypeError**. This is because you tried to do something with two different **objects**: strings and integers.

Strings and integers are part of Python's **built-in object types**, which means they come for free with Python for the programmer's convenience. This is not necessarily the case in all programming languages but this does make life very easy for the Python programmer.

It's also useful to start thinking about problems in terms of these objects. Will I need a bunch of string variables to represent a particular problem? Or am I dealing with integers? Soon, you will begin thinking about a problem using these objects as a shorthand.

Python automatically knows the difference between different object types like strings and integers, but *you* need to remember it too! (Hence the rule about having meaningful variable names).

0.3 Integers

Integers and floats are two of the basic numeric types to remember (there are others, not included here). Floats are numeric types with decimal points, and are treated differently from integers (which have none).

```
>>> myint1=40
>>> myint2= 3.4
>>> type(myint1)
>>> type(myint2)

<type 'int'>
<type 'float'>
```

```
>>> type(myint1+myint2)
<type 'float'>
```

Numbers can be added, subtracted, divided and multiplied using standard operations. The operators used are familiar: (+, -, *,/).

0.4 Strings in use

Manipulating, counting and analyzing strings will be important from the point of view of the linguist. It's useful to look at some of the basic string operations that can be carried out.

It's important to remember that Python treats strings as a *sequences* that are **immutable** i.e. they cannot be changed in place once assigned. This is different from re-assignment. We'll see some examples of this below. Lists are also a type of sequence and the operations that are applicable to strings are also possible with lists. In simple_words.py we iterated over a sentence as a sequence.

It's possible to find the length of a string using **len()**. Interestingly, Python can also concatenate strings, as well as multiply them.

```
>>> len('hello world')
11
>>> 'hello' + 'namaste'
'hellonamaste'
>>> 'hi'*5
'hihihihihi'
>>>
```

Strings are usually enclosed in single quotes, but double quotes will also work. For text that is longer than a single line, a sequence of three single quotes is used.

```
>>> greeting="hello world"
>>> greeting='''hello world
... My name is XYZ, and I live in Delhi'''
```

If the string itself contains a quotation mark, an escape character is needed to print the string correctly.

0.4.1 Slices

Slices are a very useful functionality in Python and an example of its higher-level nature as compared to other programming languages.

c	h	e	e	s	e

Table 1: We can refer to the indices of a string by 'cutting' the word. Indexes are where the knife will cut the word. Indices always begin with 0

```
>>> word="cheese"
>>> word[0]
'c'
>>> word[4]
's'
>>> word[-2]
's'
```

- The first item is always at 0, hence word[0] gives us 'c'
- The negative offset gets us the second word from the end. It is the same as word[len(word)-2].

More generally, it's possible to refer to the slices as index i, which will fetch components or parts of the word. We can represent the slices as sequences from [start: end].

Start is the upper bound number, end is the lower bound. Start bounds (left-most) are inclusive in these sequence slices. End bounds (right most) are non inclusive.

```
>>> word[1:3]
'he'
>>> word[1:]
'heese'
>>> word[:-1]
'chees'
```

- The slice word[1:3] will include the character 'h' (remember indexes start at 0) as the start bound is inclusive and the character 'e'.

- The slice word[1:] will fetch all the sequences past the first, hence 1 is the lower bound
- Finally, the slice word[:-1] will get everything but the last item. By default, the lower bound is zero, or refers to the last item, non-exclusive

Note though that slicing cannot replace the original string, a new string will have to be created in order to do this. Strings are immutable.

```
>>> word[-1] = 'y'
Traceback (most recent call last):
...
TypeError: 'str' object does not support item assignment

>>> nword = word + word[-2]
>>> nword
'cheeses'
```

0.5 String methods

String methods are handy text processing tools available in Python. [1] We saw the first of these in simple_words.py, the function **split()** This takes the sentence "Python is named after Monty Python" and splits it into a list of words.

```
>>> sentence1 = "Python is named after Monty Python"
>>> sentence1.split()
['Python', 'is', 'named', 'after', 'Monty', 'Python']
```

It's possible to supply other kinds of delimiters to split() e.g. commas in 'aa, bb, cc' will split the string into ['aa','bb','cc']

Another useful string method is **rstrip()**, which removes a newline character from the rightmost end of a string. This is useful when reading text from a file.

```
>>> line = 'aaa,bbb,ccccc,dd\n'
 >>> line = line.rstrip( )
>>> line
'aaa,bbb,ccccc,dd'
```

[1] String methods are only applicable to strings, not other sequences like lists

Check the link below for a full description of string methods.

`https://docs.python.org/2.7/library/stdtypes.html#string-methods`

0.6 References

- *Learning Python* by Mark Lutz and David Ascher
- *A Byte of Python* by Swaroop CH