Linux Supporter's Group Adelaide

Python

Python is an interpreted programming language available an most platforms – Linux, Windows, Apple.

The most common versions are Version 2 (My Linux has Version 2.7) and version 3 (The latest is 3.4). There are a few differences between them, and many people have stuck with Version 2 to avoid compatibility issues.

The major differences that I have noticed are

- The **print** command – a statement in Version 2 and a function in Version 3.
- Integer division. In Version 2, the result of integer division is an integer, in Version 3 a float. If you type 3/2 you will get either 1 or 1.5.

I will stick with Version 2 for tonight.

You can download various versions from www.python.org. There is also a basic tutorial there.

There are several ways of running python interactively:

- From the command line, just type in "python" and press return. This will give you basic python in interactive mode.
- There is an interactive version, ipython (available from ipython.org) which is more interactive – it has automatic command completion, interactive help, and lots more.
- There is IDLE – the default python IDE. This also has interactive help, and in addition can load and save python scripts.

And you can run a python script directly.

- `Python` *`filename`* will load python, execute the commands in the named file, and then exit.
- Putting
  ```
  !# /usr/env python
  ```
  on the first line of a file, marking it as executable (chmod a+x *filename*) and then typing
  ```
  ./filename
  ```
  at the command prompt will have the same effect. Since you can look for files, open them, read and write them in python, this can do anything a shell-script command can do, and is probably simpler for the more complex things.

So, let's get started. Choose one of the interactive modes.

1. Type your name and press "enter" (try both just your first name and first and last names separated by a space). It will not understand.
2. Put quotes around your name – single or double, but must use the same at both ends. It will type your name back at you.
3. Type a numeric expression – `2 * 3 + 4 * 5` – and press "enter". Does it understand the "natural" order of operations (we used to call it BODMAS, but I am informed it is now called BEDMAS).
4. You can name values:
   ```
   peter = 3
   peter * 2
   ```
5. And rename then:

```
peter = peter + 1
peter
peter = “Peter Perry”
peter
```

6. The rules for names are: Upper and lower case characters, digits, underscore are allowed. Names are case sensitive”

```
Peter
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Peter' is not defined
```

7. There are more complex data structures:
   - `lists:  l = [1, 2, 3, 'a', 'fred']`
   - `tuples:  t = (1, 2, 3, 'a', 'fred')`
8. You can select elements from a list or a tuple using square-bracket subscripts, starting from 0; you can also select sub-ranges in either:
   - `l[0]`
   - `t[4]`
   - `l[1:3]`
9. You can change values in a list, but not in a tuple:
   -
```
l[3] = “Jim”
l
```
10. Structured statements end with a colon; subsequent lines are indented **by the same number of spaces.** Ipython and idle will do this for you; raw python needs you to do it manually:
    -
```
for val in l:
  print val
```
11. Let's look at **print** a little more. This is Version 2 stuff. You can print several things on a line by separating them by commas; put a comma at the end of the line to not start a new line.

```
print val, ' * 12 = ', val * 12
```

    In Version 3, print is a function:

```
print(val, ' * 12 = ', val * 12)
```

    and to stay on the same line:

```
print(val, ' * 12 = ', val * 12, end='')
```

12. A useful function with loops is the **range** function:

```
range(0,10)
for num in range(1,12):
    print num
```

Now a little programming exercise: print out the 12 times table (not just the answers, but like you learnt it as a kid.

```
1 x 12 = 12
2 x 12 = 24
```

A crucial part of any programming language (with the exception of a few like Prolog) is the ability to define procedures and functions. Lets define a function to print just one line of the table:

```
def tableLine(number, multiplier = 12):
    print number, 'x', multiplier, '=', number * multiplier

tableLine(3)
tableLine(5, 7)
for num in range(0,13):
    tableLine(num)
tableLine(multiplier=5, number = 7)
```

The ability to give parameters default values and to pass parameters by name is one of the neater features of python. Remember, values do not have to be numbers, they can be strings or lists or more complex structures (provided the procedure knows what to do with them). Try this:

```
TableLine([1,2.,3])
```

That was a procedure; to make a function, simply put `return` statement in it.

```
def mysqrt(x):
    y = x * 0.5
    for i range(1,5):
        y = (y + x / y) * 0.5
     return y

mysqrt(3)
```

You can return lists – look at this:

```
def func(x, y=10, z = 2):
    return (x + y, y + z)

a,b = func(1)
print a, b
```

I should show you how to do tests (most useful in a definition):

```
name = 'fred'
if (name == “joe”):
   print 'this is joe'
else:
    print 'this is not joe'
```

You can of course test for `!=, >, <, >=, <=`

Another useful feature is the built in help

```
help(l)
```

This should give you pages of things you can do with a list. Press space to get the next page, q to exit.

Dictionaries are another special structure:

```
d = {1:'one', 100:'one hundred'}
d[1]
d[100]
d[2] = 'Three'
```

The first entry can be a string:

```
ds={'one':'ONE', 'two':'too many'}
d['two']
```

I have not mentioned the useful `while:` structure, nor the `try: .. except:` structure (less useful for beginner programmers), nor defining your own classes – perhaps we could talk about "object-oriented-programming" another time.

Of course you can read and write files from python. Look up the python tutorial on the web (https://docs.python.org/2/tutorial for python version 2, and https://docs.python.org/3/tutorial/ for python version 3) to find out how.

There are lots of packages you can download from the web – good sources are `www.numpy.org` and `scikits.appspot.com/scikits` and `www.SciPy.org`. For example, there is a package for numerical processes (solving equations, etc) called numpy:

```
import numpy as np
x = np.reshape(range(1,13), (3,4))
x.shape   ==> (3,4) – rows then columns
np.linalg.svd(x)
x = np.random.randn(3, 4)
y = np.random.randn(4,7)
x.dot(y)   ==> Matrix multiply
```

And I did not mention, to get out of normal python, used ctrl-D or the `quit()` function; ipython should be the same, while IDLE has a menu exit.