

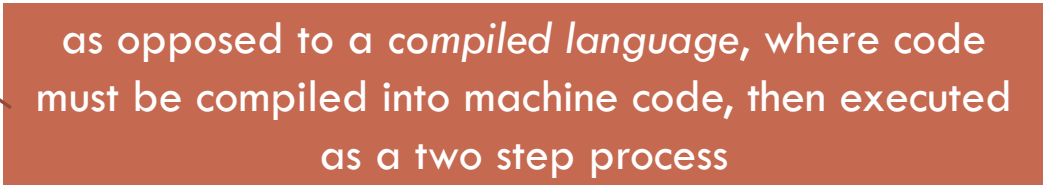


ATMS 305 WEEK 4: INTRODUCTION TO PYTHON

Lecture 1: Using python in a linux
environment

GETTING STARTED WITH PYTHON

python is an *interpreted language* that has been around since 1990.



as opposed to a *compiled language*, where code must be compiled into machine code, then executed as a two step process

Other examples of interpreted languages: javascript, MATLAB™

Other examples of compiled languages: c, c++, FORTRAN, java

Interpreted languages are more flexible, but less efficient.

Compiled languages are less flexible (must recompile every change to code), but are very efficient. (i.e. short startup time, and low-level machine code)

TYPES OF PYTHON

We will be using Cpython (usually just called python). It is written “under the hood” in the c language.

There are also other types of python (jython, pypy, ironpython, stackless python).

We will be using python version 3.5 in this course. Python 2.7 is also in common use, but will be decommissioned in 2020.

There are a few syntax differences between these versions.

There are many ways to install python on a computer – it is available for almost all operating systems (linux, Windows, OS X).

We will be using a python distribution from Continuum Analytics, Inc. called Anaconda python.

(<https://docs.continuum.io/anaconda/>)

It is free, and comes with an easy to use package manager called **conda**.

There are other distributions available.

USING ANACONDA PYTHON ON KEELING

DAS operates a python distribution on keeling that we will use for this class.

To activate it, type

```
module load Anaconda2
```

```
module load Anaconda3
```

 **We will use this one for Anaconda python 3.5**

To check which python distribution you are using, make sure it says “Anaconda”:

```
netID@keeling ~$ python
```

```
Python 3.5.2 |Anaconda custom (64-bit)| (default, Jul 2 2016, 17:53:06) 
```

```
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

Tip: If you have your own version of python installed on keeling, this will supersede it for this session.

Log out and log back in to reset.

PYTHON VS IPYTHON

At the command line, there are two options to run python interactively.

`python`

which is the bare bones command interpreter, and is quickest to start up, and

`ipython`

which stands for “interactive python”, and contains a bit more user friendliness (i.e., syntax coloring, online help). It also is a bit slower to start up and use than python.

Give each a try.

To exit from python, type `exit()` or Control-D.

WELCOME TO PYTHON

Some conventions:

python scripts are given the extension
.py

python scripts are called with the
following syntax

```
python my_script.py
```

There are other ways to run python and
ipython outside of the command line,
which we will cover later in the course.



PYTHON RULES OF THE ROAD

- python is **case sensitive** Variable != variable
- functions must be called with **parentheses**, for example

```
print()
```

```
print('Hello world')
```

- python uses **indentation** to mark blocks of code, beginning with a line ending in a colon :

```
for i in range(0,5):
```

```
    x = i**2
```

```
    print(x)
```

```
print('all done')
```

Note: no begin or end. The indentation handles that in python!

PYTHON RULES OF THE ROAD

- Lines in python are continued with a backslash \
- ```
var1 = 'I have a lot to say about this \
 but I don't have room to say it.'
```

One exception is if a statement is contained in brackets:

```
print('You say goodbye
 and I say hello')
```

- Comments are made using the #, like bash
- ```
#Don't interpret me wrong
```


VARIABLE NAMES AND ASSIGNMENT

- Variables can have almost any name you want, except
 - They can't start with a number
 - Unless you know what you're doing, don't start a variable with an underscore _
 - Remember, case sensitivity Pressure != pressure

- Variables are assigned using =

a = 6.7

a, b, c = 6.7, 'hello', 8

RESERVED WORDS

You may not name your variables any of the following words as they mean special things in Python:

and	assert	break	class	continue	def	del	elif	else	except
exec	finally	for	from	global	if	import	in	is	lambda
not	or	pass	print	raise	return	try	while		

Do NOT use any of the following words either (although they are not strictly Python reserved words, they conflict with the names of commonly-used Python functions):

Data	Float	Int	Numeric	Oxphys	array	close	float	int	input
open	range	type	write	zeros					

You should also avoid all the names defined in the math library (you must avoid them if you import the library):

acos	asin	atan	cos	e	exp	fabs	floor	log	log10
pi	sin	sqrt	tan						

NUMERIC DATA TYPES

Computers can represent data in several ways. These are useful for various purposes, including counting, doing calculations, complex numbers, etc.

- A boolean variable can be True or False

```
bool(3)
```

```
bool(0)
```

```
bool(-3)
```

```
bool(None)
```

```
bool([]) #an empty list
```

```
bool([3, 5, 'hi'])
```

INTEGER DATA TYPES

There are two integer data types in python, **integer** and **long integer**. Integer data types can represent numbers from about -200 million to 200 million (as a 32-bit representation). Long integers can extend much farther, but use more memory.

```
a = 56
```

```
a = 56L
```

```
c = 4808305794375897348957394534534
```

FLOATING POINT

Floating point numbers are represented as 64-bit representations. (This would be a double precision number in other languages like FORTRAN and MATLAB™)

$f = 4.56$

$f = 3.2e10$

$f = .22E-64$

COMPLEX NUMBERS

Python can represent complex numbers with the following syntax:

```
a = 4.9 - 8.3j
```

```
b = complex(5.3, 2.4)
```

Try `a+b`

CONVERSION BETWEEN DATA TYPES

The following functions are available for converting between data types

```
float(4)
```

```
int(4.6)
```

```
long(4.6)
```

```
c = int(round(4.6))
```

```
d = int(round(4.4))
```

```
print(c,d)
```