

AMath 483/583 — Lecture 5

This lecture:

- Python concepts and objects
- Data types, lists, tuples
- Modules
- Demo — plotting and IPython notebook

Reading:

- [class notes: Python sections](#)
- [class notes: Python in bibliography](#)
- [IPython notebook documentation](#)

Notes:

Homeworks

We will try to clone all repositories today.
We will send email if we fail.

[Make sure you gave us read permission in your bitbucket repository.](#)

Note that `git log` gives time of the commit you want graded, so even if there are glitches with bitbucket we will know if you completed it on time. (Due at 11:00pm PDT.)

[Homework 2 will be posted by tomorrow, due next Wednesday.](#)

Notes:

AMath 483/583 — Lecture 5

This lecture:

- Python concepts and objects
- Data types, lists, tuples
- Modules
- Demo — plotting and IPython notebook

Reading:

- [class notes: Python sections](#)
- [class notes: Python in bibliography](#)
- [IPython notebook documentation](#)

Notes:

Python

Python is an **object oriented** general-purpose language

Advantages:

- Can be used interactively from a Python shell (similar to Matlab)
- Can also write scripts to execute from Unix shell
- Little overhead to start programming
- Powerful modern language
- Many **modules** are available for specialized work
- Good graphics and visualization modules
- Easy to combine with other languages (e.g. Fortran)
- Open source and runs on all platforms

Notes:

Python

Disadvantage: Can be slow to do certain things, such as looping over arrays.

Code is **interpreted** rather than compiled

Need to use suitable modules (e.g. NumPy) for speed.

Can easily create custom modules from compiled code written in Fortran, C, etc.

Can also use extensions such as **Cython** that makes it easier to mix Python with C code that will be compiled.

Python is often used for high-level scripts that e.g., download data from the web, run a set of experiments, collate and plot results.

Notes:

Object-oriented language

Nearly everything in Python is an **object** of some **class**.

The class description tells what data the object holds (**attributes**) and what operations (**methods** or functions) are defined to interact with the object.

Every **“variable”** is really just a pointer to some object. You can reset it to point to some other object at will.

So variables don't have **“type”** (e.g. integer, float, string). (But the objects they currently point to do.)

Notes:

Object-oriented language

```
>>> x = 3.4
>>> print id(x), type(x) # id() returns memory add
8645588 <type 'float'>

>>> x = 5
>>> print id(x), type(x)
8401752 <type 'int'>

>>> x = [4,5,6]
>>> print id(x), type(x)
1819752 <type 'list'>

>>> x = [7,8,9]
>>> print id(x), type(x)
1843808 <type 'list'>
```

Notes:

Object-oriented language

```
>>> x = [7,8,9]
>>> print id(x), type(x)
1843808 <type 'list'>

>>> x.append(10)
>>> x
[7, 8, 9, 10]
>>> print id(x), type(x)
1843808 <type 'list'>
```

Note: Object of type 'list' has a method 'append' that **changes** the object.

A list is a **mutable object**.

Notes:

Object-oriented language — gotcha

```
>>> x = [1,2,3]
>>> print id(x), x
1845768 [1, 2, 3]

>>> y = x
>>> print id(y), y
1845768 [1, 2, 3]

>>> y.append(27)
>>> y
[1, 2, 3, 27]

>>> x
[1, 2, 3, 27]
```

Note: x and y point to the same object!

Notes:

Making a copy

```
>>> x = [1,2,3]
>>> print id(x), x
1845768 [1, 2, 3]

>>> y = list(x)    # creates new list object
>>> print id(y), y
1846488 [1, 2, 3]

>>> y.append(27)

>>> y
[1, 2, 3, 27]

>>> x
[1, 2, 3]
```

Notes:

integers and floats are immutable

If `type(x)` in `[int, float]`, then setting `y = x` creates a **new object** `y` pointing to a new location.

```
>>> x = 3.4
>>> print id(x), x
8645588 3.4

>>> y = x
>>> print id(y), y
8645572 3.4

>>> y = y+1

>>> print id(y), y
8645572 4.4

>>> print id(x), x
8645588 3.4
```

Notes:

Lists

The **elements of a list** can be **any objects** (need not be same type):

```
>>> L = [3, 4.5, 'abc', [1,2]]
```

Indexing starts at 0:

```
>>> L[0]
3

>>> L[2]
'abc'

>>> L[3]
[1, 2]

>>> L[3][0] # element 0 of L[3]
1
```

Notes:

Lists

Lists have several built-in methods, e.g. `append`, `insert`, `sort`, `pop`, `reverse`, `remove`, etc.

```
>>> L = [3, 4.5, 'abc', [1,2]]
>>> L2 = L.pop(2)
>>> L2
'abc'
>>> L
[3, 4.5, [1, 2]]
```

Note: L still points to the same object, but it has changed.

In IPython: Type `L.` followed by `Tab` to see all attributes and methods.

Notes:

Lists and tuples

```
>>> L = [3, 4.5, 'abc']
>>> L[0] = 'xy'
>>> L
['xy', 4.5, 'abc']
```

A **tuple** is like a list but is **immutable**:

```
>>> T = (3, 4.5, 'abc')
>>> T[0]
3
>>> T[0] = 'xy'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support
      item assignment
```

Notes:

Python modules

When you start Python it has a few basic built-in types and functions.

To do something fancier you will probably **import** modules.

Example: to use square root function:

```
>>> from numpy import sqrt
>>> sqrt(2.)
1.4142135623730951
```

Notes:

Python modules

When type `import modname`, Python looks on its [search path](#) for the file `modname.py`.

You can add more directories using the Unix environment variable `PYTHONPATH`.

Or, in Python, using the `sys` module:

```
>>> import sys
>>> sys.path # returns list of directories
['', '/usr/bin', ...]

>>> sys.path.append('newdirectory')
```

The empty string "" in the search path means it looks first in the current directory.

Notes:

Python modules

Different ways to import:

```
>>> from numpy import sqrt
>>> sqrt(2.)
1.4142135623730951
```

```
>>> from numpy import *
>>> sqrt(2.)
1.4142135623730951
```

```
>>> import numpy
>>> numpy.sqrt(2.)
1.4142135623730951
```

```
>>> import numpy as np
>>> np.sqrt(2.)
1.4142135623730951
```

Notes:

Graphics and Visualization

Many tools are available for plotting numerical results.

Some open source Python options:

- [matplotlib](#) for 1d plots and 2d plots (e.g. pseudocolor, contour, quiver)
- [Mayavi](#) for 3d plots (curves, surfaces, vector fields)

Mayavi is easiest to get going by installing the [Enthought Python Distribution \(EPD\)](#), which is available for many platforms. (Also includes NumPy, SciPy, matplotlib.)

Notes:

Graphics and Visualization

Open source packages developed by National Labs...

- [Visit](#)
- [ParaView](#)

Harder to get going, but designed for large-scale 3d plots, distributed data, adaptive mesh refinement results, etc.:

Each have stand-alone GUI and also Python scripting capabilities.

Based on [VTK](#) (Visualization Tool Kit).

Notes: