Python Basics

Leigh J. Little SUNY Brockport November 2, 2020

Contents

1 Languages

There are many computer languages available (some would argue that there are too many). Python has rapidly become one of the most popular programming languages for general purpose programming.

1.1 Types of Languages

Fortran, C, C++ and Javascript (among others) are called strongly typed languages. This means that every variable that must be given a specific type (double, integer, logical, complex, string) and array variables need to be given a fixed size before being used.

MATLAB is a weakly typed language. A variable can be whatever it needs to be at any time. For example, the sequence of statements below would is legal in MATLAB

```
>>  a = 1             % a starts as an integer
>>  a = 1.345         % a is now a double precision
>>  a = -2 + 2i       % a is now a complex
>>  a = (1>2)         % a is now a logical
>>  a = [1 2 3 4 5 6] % a is now an integer array
```

This sequence would not be permissible in most languages. The variable `a` would need to be given a specific, fixed type before it could be used and this type could not change during the program execution. For the last statement, the language would need to be explicitly told that `a` is an array variable and the array would need to be given a fixed length. Examples of how this would be done in other languages are given below:

```
INTEGER(10) :: a          % Fortran - define an array of 10 integers
integer a[10];            % C/C++
var a = new array(10);    % Javascript
```

Python is somewhere between MATLAB and strongly typed languages. In many cases, Python variables will behave like MATLAB variables in that they can change types as needed however, there are cases where variables will need to be given specific types.

One important thought to keep in mind is that Python is ultimately a programming language and thus has fundamental similarities with all other languages

a) It has a working environment.

b) It has variables.

c) It can perform operations such as basic math operations on these variables.

d) It can perform conditional logic (`if-then` testing).

e) It can perform looping operations.

f) It permits you to write your own functions.

2 Python Basics

There is only one MATLAB environment. Octave exists, but it aims to exactly reproduce the MATLAB environment. There many Python *environments*. One some level they are all the same, but they may change appearance and how you interact with them. We will use the `IDLE` environment (mainly because this is the one installed on the campus network and the one a basic Python installation comes with). You can download this for use on your own computer.

2.1 Starting Python

Due to its minimalistic nature, it can be difficult to tell Python what your working directory is and to change into that working directory. The steps below are the easiest way to start Python in a desired working directory:

- Copy the file `blank.py` from the course website into the directory you want to work in. This is an empty file with the Python `.py` extension.
- Right click on the `blank.py` file and choose `Edit with IDLE` from the context menu that comes up. This will open the file in the IDLE Editor.
- From the tool bar at the top of the Editor, choose `Run -> Run Module` or press `F5`.
- This last step will open the Python `shell`. Notice that the shell looks much like the MATLAB command window with one extra greater than sign.
- Your Editor and shell should now be set to work in the directory where `blank.py` is located.

2.2 Important - Emailing Python Files

The campus email system will block all `.py` files. When you email me your Python files, change the extension to `.txt`

2.3 Case Sensitivity

Like MATLAB, variable names in Python are case sensitive. For example, open up the Python shell if you have not already done so, then enter the following

```
>>> A = 3
>>> a = 4
>>> A
3
>>> a
4
>>> c
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    c
NameError: name 'c' is not defined
```

Like MATLAB, we can display the value of a variable on the screen just by typing the value at the shell command line. However, we can't automatically display the value by leaving a (`;`) off the end of the line. An error occurs when we try to display the value of `c` because this variable does not exist.

Syntax	Operation
`x**y`	Raise `x` to the `y` power
`x//y`	Divide `x` by `y` and round down
`x%y`	Remainder when `x` is divided by `y`

Table 1: Other math operators in Python.

2.4 Basic operations

The give basic operations (add, subtract, multiply, divide and assignment) are the same as in most other languages, but there are some additional operations to be familiar with There are actually a great many more operations, but we will not need to use them.

2.5 Basic Math Functions (and Libraries)

The basic math functions are also available. If you wanted to compute the sine of 5, you might try

```
>>> sin(5)
>>> Sin(5)
>>> sin[5]
>>> Sin[5]
```

however, all of these will give you a *name not found* error similar to the one in the example from the previous section. In order to gain access to these functions, we must make them available to Python.

These functions are contained in a *library*. A library is a collection of Python functions that can be used. Most languages (including MATLAB) have libraries for providing additional computing capabilities, however, MATLAB makes these known to the shell automatically. In Python, you need to explicitly bring these into your shell using the `import` command.

```
>>> import math as m      #   Comment symbol in Python is the # sign
>>> a = sin(5)            #   This command gives an error
>>> a = m.sin(5)
>>> a
-0.9589242746631385       ####  Python uses 16 digit double precision
```

The first line in the above statements is

```
>>> import math as m      # Import all of math with prefix m
                          # It is possible to import only parts of a library
                          #    but we won't need to do that in this class.
```

This tells Python to import all of the functions in the `math` library into the Python environment with a *prefix* of `m`. You can choose any name for the prefix other than a prefix you are already using.

The reason for this unusual behavior is that the Python language was written in a way such that its programs and environment consume very little memory. As such, it starts with only a very basic set of commands and functions. This enables Python programs to be used in what are called *embedded applications*. These are situations where it is not easy to modify a program once it is written. This includes programs burned onto an integrated circuit and programs in devices such as cash registers, automatic bank tellers, coin exchange machines, *etc.* These devices typically have very low available memory and it is critical that programs that run on them use as little memory or other system resources as possible.

Because the sine function is part of the `math` library, it needs to be preceded by the prefix `m` in order to access it. This is characteristic of all library functions in Python. We will use several libraries in the next couple of weeks.

Some of the standard math functions in the library are given in Table 2.

Function Name	Operation
`sqrt`	Square root function
`sin, cos, tan`	Sine, cosine and tangent
`asin, acos, atan`	Inverse sine, cosine and tangent
`sinh, cosh, tanh`	Hyperbolic sine, cosine and tangent
`asinh, acosh, atanh`	Inverse hyperbolic sine, cosine and tangent
`atan2(y,x)`	Four quadrant inverse tangent of `y/x`.
`exp`	Exponential
`log`	Natural log
`log(x,y)`	Base `y` log of `x`
`degrees, radians`	Convert degrees to radians or vice versa
`pi, e, nan, inf`	Built in values for π, e, not a number and infinity.

Table 2: Some of the math functions in the Python Math library. Remember that each of these needs to be preceded by the `m` prefix. Also, all angles are assumed to be in radians when working with the trigonometric functions.

2.6 Script Files

As with MATLAB, you can enter your commands from the shell or you can use a *script file*. This is a file that contains a series of Python commands that get executed as if you typed them in at the shell prompt. Python script files are plain text files and have the `.py` extension. For example, enter the follwing into a file called `test.py`

```
# Contents of test.py
import math as m
a = m.sin(5)
b = m.exp(2)
c = (a+b)/2
d = (a+b)//2
print(a)
print(b)
print(c)
print(d)
```

then either press `F5` or choose `Run -> Run Module` from the Editor menu to execute the script.