Python Basics

Leigh J. Little SUNY Brockport November 4, 2022

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. Depending on which list you find, it is consistently ranked somewhere in the top five.

1.1 Types of Languages

Fortran, C, C++, and Javascript (among others) are called strongly typed languages. This means that every variable that is used in a program 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 is legal in MATLAB

```
>>  a = 1               % a starts as an integer
>>  a = 1.345           % a is now double precision
>>  a = -2 + 2i         % a is now complex
>>  a = (1>2)           % a is now 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. In the case of an array variable, the program would need to be explicitly told that `a` is an array variable and the array would need to be given a fixed length before it could be used; for example:

```
INTEGER :: a(10)          % Fortran
integer a[10];            % C/C++
var a = new array(10);    % Javascripit
```

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 an environment to work in. This refers to how programs are written and run.

b) It has rules of syntax.

c) It has variables (integer, double precision, string, vectors)

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

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

f) It can perform looping operations.

g) It permits you to write your own functions.

Most languages offer many capabilities beyond these fundamental building blocks, but it is important to understand that the list of items above forms the foundation on which all advanced capabilities are built.

2 Python Basics

There is only 1 MATLAB environment. Octave exists, but it aims to (almost) exactly replicate 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. In this class, we will use the *Anaconda* Python environment with the *Spyder* code editor. You can download this for use on your own computer (see course website).

Python has gone through several revisions over its lifetime. For much of this time, the second version (called Python 2) was used. It recently underwent a significant upgrade to Python 3. One result of this upgrade was a change in some syntax. In this course we will be using Python 3, but you should be aware that Python 2 is still used due to the volume of material that has been written.

Go to the Windows search box and start to type Anaconda. You should see the Anaconda Navigator come up. Click on this. Alternatively, you can search for Sypder and click on this. The Spyder environment has 3 panes; the large pane on the left is the code editor. The small panel in the upper right is the Spyder console and the small panel in the lower right is the Python console (or command line).

2.1 Case Sensitivity

Like MATLAB, variable names in Python are case sensitive. For example, type the following in the Python console:

```
In [1]: A = 4
In [2]: a = 3
In [3]: A
Out [3]: 4
In [3]: a
Out [3]: 3
In [4]: c
Traceback (most recent call last):

  File "<ipython-input-5-2b66fd261ee5>", 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. Also, an error occurs when we try to display the value of `c` because this variable does not exist.

The `In [*]` and `Out [*]` values can be used to bring back any previous input or output, similar to Mathematica. For example,

```
In [5]: In[1]
Out [5]: 'A = 4'
In [6]: Out[3]
Out [6]: 4
```

Not all Python environments have this capability.

2.2 Basic operations

The four basic arithmetic operations (add, subtract, multiply, divide) 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,

Syntax	Operation
`x = y`	Set `x` equal to `y`
`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: Math operators in Python.

but we will not need to use them.

2.3 Basic Math Functions (and Libraries)

Suppose we wanted to compute the sine function for some angle. There are various commands we could try:

```
In []:  sin(5)
In []:  Sin(5)
In []:  sin[5]
In []:  Sin[5]
```

however each of these results in a error indicating that the `sin` function is not defined. In order to gain access to the sine function, we must make it available to Python.

The sine function is 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.

```
In []:  import math as m
In []:  sin(5)
(get sin not found error)
In []:  m.sin(5)
Out[]: -0.9589242746631385
```

The first line in the sequence of statements above is

```
In []:  import math as m   ### Import all of the math library with prefix m
                           ### You can choose to import an entire library, or
                           ### only portions of a library.  We will see this later
                           ### in the semester.
```

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. Other standard functions in the math library are shown in Table 2.

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.

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.

An important note on importing libraries. You can always use a syntax like

Function Name	Operation
`sqrt`	Square root function
`sin, cos, tan`	Sine, cosine and tangent functions
`asin, acos, atan`	Inverse sine, cosine and tangent functions
`sinh, cosh, tanh`	Hyperbolic sine, cosine and tangent functions
`asinh, acosh, atanh`	Inverse hyperbolic sine, cosine and tangent functions
`atan2(x,y)`	Four quadrant inverse tangent function
`exp`	Exponential function
`log`	Natural log function
`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. All angles are assumed to be in radians. Inverse trig functions return an angle in radians. These need to be preceded by your chosen prefix to access them.

```
In []:  import math
```

when importing a library. If you do this, then you can access the functions in the math library without a prefix, for example

```
In []:  sin(5)
```

This approach is strongly discouraged. The reason for this is that a library might contain hundreds of functions. There is a very real possibility that the name of one of the functions in the library will conflict with the name of a function in another library or a function you have written yourself. By giving each library you import a unique prefix you ensure that there will not be conflicts in functions names.

3 Writing Programs

3.1 Working Directory

Before you start writing any programs, you should put Anaconda into your preferred working directory. You can change your working directory using the path browser located at the far right of the main toolbar.

3.2 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, click on blank sheet of paper at the far left of the main toolbar. This will create a new Python script file. I prefer to remove all the existing commands that are present in the file.

Enter the commands below into a file and save the file as `test.py` (make sure you save this in the same directory as your working directory).

```
# In Python, the comment symbol is the pound sign
#
# There is no 'official' way to do block commenting, but you can
# unofficially use a sequence of triple double quotes
#
#    """
#        Block of comments
#    """
import math as m
a = m.sin(5)
```

```
b = m.exp(2)
c = (a+b)/2
d = (a+b)//2
print('a = ',a)
print('b = ',b)
print('c = ',c)
print('d = ',d)
```

You can run the script by clicking on the green arrow (right below the word Run) in the main toolbar. You should see something like:

```
runfile('D:/Classes/AF2021/ESC251/Progs/python/test.py',
   wdir='D:/Classes/AF2021/ESC251/Progs/python')
a =  -0.9589242746631385
b =  7.38905609893065
c =  3.215065912133756
d =  3.0
```

3.3 The `input` Command

The `input` command is used to get user input from the keyboard while a script is running. However, the input is always a string type. If you need to input numerical data, you need to use a type conversion command to get the input as either an integer or a double. Remember that an integer has no digits after the decimal place and is usually used for indexing or counting. Most input variables will be double precision.

```
# You can use either single or double quotes for the prompt string
x = int(input('Input x '))    # Input an integer for x
y = float(input('Input y '))  # Input a double precision value for y
```

3.4 Emailing `.py` Files

The campus email system will block all `.py` files. If you need to email me a `.py` file, replace the `.py` extension with `.txt`.