Center for Teaching, Research & Learning
Social Science Research Lab
American University, Washington, D.C.
http://www.american.edu/provost/ctrl/
202-885-3862

Introduction to Python

Python is free **software and is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. It is used for its readability and productivity.**

Course Objective

This course is designed to give a basic understanding of Python. It assumes some basic familiarity with computer programming. By the end of this course you will know the basic use of the Python Shell.

NOTE: Most of our examples are from the Python website's tutorial, http://docs.python.org/py3k/contents.html so if you are comfortable with programming languages feel free to check out the tutorials there. They are more comprehensive, and go into greater depth.

Installing and starting Python

Python is distributed free from the website http://www.python.org/. There are distributions for Windows, Mac, Unix, and Linux operating systems. From http://python.org/download there are all of these distribution files for easy installation. The latest version of Python is version 3.1.

Once installed, you can find the Python3.1 folder in "Programs" of the Windows "start" menu. For computers on campus, look for the "Math and Stats Applications" folder. To open python, select "IDLE (Python GUI)". A new window titled "Python Shell" will appear.

The program IDLE does two things. First, it allows you to create, run, and save more comprehensive programs in Python. Secondly, and our focus of this introduction, is that it also acts as an interpreter. This means instead of having to compile code, you can simply type code into the Python Shell after the >>> prompt, and IDLE will execute that code. This is the ease of development in Python, because you can test bits of code immediately to see if it works properly.

Using the Python Shell as a calculator

Because Python is an interpreted language, one can use the Python Shell as a calculator. When you open IDLE, the Python Shell starts and there is a prompt “>>>” followed by a blinking cursor. Try typing some of the commands below.

```
>>> 2+2
4
>>> # This is a comment
... 2+2
4
>>> 2+2  # and a comment on the same line as code. Python ignores them!
4
>>> (50-5*6)/4
5.0
>>> 8/5 # Fractions aren't lost when dividing integers
1.6
```

Use Python to manipulate strings

String is a sequence of letters. Besides numbers, Python can also manipulate strings, which can be expressed in several ways. They can be enclosed in single/double quotes.

```
>>> "doesn't"
"doesn't"
>>> name = "American University"          # String can be assigned to a variable as well.
>>> name
'American University'
>>>print name                             # “print” statement displays the content of string object.
American University
```

(Note: See Online Tutorial 3.1.2. for more about strings. It is at http://docs.python.org/py3k/tutorial/introduction.html)

Compound data types: List, Tuples, Dictionaries

Python has a number of ways to store compounded data types.
Three of such data types (list, tuple, and dictionaries) will be explained in this section.

“List, example”

```
>>> a = ['spam', 'eggs', 100, 1234]       # “list” can be written as a list of comma-separated values
                                          # between square brackets.
>>> a
['spam', 'eggs', 100, 1234]
>>> a[0]                                  # List indices item starting at 0.
'spam'
```

```
>>> a[:2]                                  # List can be sliced.
['spam', 'eggs']
>>> a[:2]+["bacon"]                        # List can be concatenated.
['spam', 'eggs', 'bacon']
>>> a[0] = "Ham"
>>> a
['Ham', 'eggs', 100, 1234]                    # Element of list can be replaced.
```

“Tuple, example”

```
>>> t = 12345, 54321, 'hello!'             # ”tuple” consists of a number of values separated by
commas.
>>> t
(12345, 54321, 'hello!')
>>> t[0]                                   # Tuple indices item starting at 0, like “list”.
12345
>>> t[0] = 54321                           # Element of tuple cannot be replaced.
(TypeError Message)
```

“Dictionary, example”

```
>>> tel = {'jack': 4098, 'sape': 4139}     # “dictionary” can be written as a comma-separated list of
                                           key:value # pairs, where “key” indices “value”.
>>> tel['jack']                            # Value(4098) is called by the corresponding key(‘jack’).
4098
>>> tel['jack']=1000                       # Element of dictionary can be replaced.
>>> tel
{'sape': 4139, 'jack': 1000}
>>> tel.keys()                             # keys() method returns a list of all the keys used in the
                                          dictionary

 ['sape', 'jack']
```

(Note: See Online Tutorial 3.1.4 and 5.1 for more about List)

Other data types

Boolean has two types of values: True (=1) and False (=0) that follow Boolean logic.

“Boolean, example”

```
>>> True and False
False
```

(See Online Tutorial 6.1. and 6.2. for more about Boolean Operations.)

Set is a compound data type, but its elements are not ordered and unique. “Common uses include membership testing, removing duplicates from a sequence, and computing mathematical operations such as intersection, union, difference, and symmetric difference.” (Online Tutorial 6.7.)

```
”Set, example”
 >>> A = set( [] )                        # set( iterable ) defines a set. In this case, the set is empty.
>>> A
set([])
>>> B = set( [1,2,2,3] )                  # set is an unordered collection of unique elements
 >>> B set([1, 2, 3])
 >>> B[0]
 (TypeError Message)
```

(See Online Tutorial 6.7. for more about set data type.)

Built-in functions and methods

There are various functions in Python. A function is a portion of code that performs a specific task. Arguments are put in the following parenthesis.

```
“Built-in function examples”
>>> abs(-23)                              # abs(x) returns the absolute value of the input
23
>>> len( [1,2,3,4] )                      # len(s) returns the length of the object
4
```

(Note: See Online Library Reference for more about built-in functions. It is strongly recommended to always refer to it.)

There are also various methods in Python. A method is similar to function as it performs a specific task, unlike a function however, it is associated with a specific object. Following the object and period (.), methods are called.

```
“Method examples”
>>> name = "American University"
>>> name.upper()                          # .upper() returns a copy of the string, converted to upper
…
                                          # …letters.
'AMERICAN UNIVERSITY'
>>> L = ["A", "B"]
>>> L.append("C")                         # .append(x) adds x to the end of the sequence in list.
>>> L
['A', 'B', 'C']
```

Help!

Overwhelmed by the number of functions? Python has a well-documented “help” function.

“Python help, example”
```
>>> help(abs)
>>> help(name.upper)
>>> help(help)
```

Alternatively, you can read online documentation (http://docs.python.org/) or use the “module docs” (located in Python 2.6 folder) to find the descriptions of the function.