Introduction to Python

Goddard Python User's Group

pythonbootcamp@bigbang.gsfc.nasa.gov

Goddard Space Flight Center

June 13, 2016

What is Python?

Python is an elegant and robust programming language that combines the power and flexibility of traditional compiled languages with the ease-of-use of simpler scripting and interpreted languages.

What is Python?

- High level
- Interpreted
- Scalable
- Extensible
- Portable
- Easy to learn, read and maintain
- Robust
- Object oriented
- Versatile

Why Python?

- Free and Open source
- Built-in run-time checks
- Nested, heterogeneous data structures
- OO programming
- Support for efficient numerical computing
- Can be used to build server-side web applications
- Good memory management
- Can be integrated with C, C++, Fortran and Java
- Easier to create stand-alone applications on any platform

What Will be Covered Today

1. **Using the Python Interpreter**
2. **Simple Program**
3. **Print Statement**
4. **Python Expression**
5. If Statements, Loops, While Statements
6. List, Tuples, Dictionary
7. Functions and Modules
8. Files I/O
9. Exception Handling
10. List Comprehension (if possible)

Using the Python Interpreter

- **Interactive interface to Python**

```
% python
Python 2.7.9 |Anaconda 1.8.0 (x86_64)| (default, Dec
[GCC 4.2.1 (Apple Inc. build 5577)] on darwin
Type "help", "copyright", "credits" or "license" for
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and http
>>>
```

- **Python prompts with** `>>>` **.**

Using the Python Interpreter

- **Python interpreter evaluates inputs:**

```
>>>  -2*(4+9)
-26
>>>
```

- **To exit Python:**
  - CTRL-D

Examples

Type

```
% python
>>>
```

and let us do some basic operations.

Scientific Hello World

- Provide a number to the script
- Print 'Hello World' and the *sine* value of the number

To run the script, type:

```
%python helloWorld.py 3.14
```

Purpose of the Script

- Read a command line argument
- Call a math (sine) function
- Work with variables
- Print text and numbers

The Code

```
1  #!/usr/bin/env python
2
3  import sys
4  import math
5
6  r = float(sys.argv[1])
7  s = math.sin(r)
8  print "Hello, World! sin(" + str(r) + ")=" + str(s)
```

Header

- **Explicit path to the interpreter:**

```
#!/usr/bin/python
```

- **Using env to find the first Python interpreter in the path:**

```
#!/usr/bin/env python
```

Importing Python Modules

The standard way of loading a module is:

```
import scipy
```

We can also use:

```
from scipy import *
```

We may choose to load a sub-module of the main one:

```
import scipy.optimize
from scipy.optimize import *
```

We can choose to retrieve a specific function of a module:

```
from scipy.optimize import fsolve
```

You can even rename a module:

```
import scipy as sp
```

Alternative Print Statements

- **String concatenation:**

```
print "Hello, World! sin(" + str(r) + ")=" + str(s)
```

- **C printf-like statement:**

```
print "Hello, World! sin(%g)=%g" % (r,s)
```

- **Variable interpolation:**

```
print "Hello, World! sin(%(r)g)=%(s)g" % vars()
```

Printf Format Strings

%d : integer

%5d : integer in a field of width 5 chars

%-5d : integer in a field of width 5 chars, but adjusted to the left

%05d : integer in a field of width 5 chars, padded with zeroes from the left

%g : float variable in %f or %g notation

%e : float variable in scientific notation

%11.3e : float variable in scientific notation, with 3 decimals, field of width 11 chars

%5.1f : float variable in fixed decimal notation, with one decimal, field of width 5 chars

%.3f : float variable in fixed decimal form, with three decimals, field of min. width

%s : string

%-20s : string in a field of width 20 chars, and adjusted to the left

Exercise

Write a Python script that:

- Takes your first name and the city you live in, and
- Prints: My name is **firstName** and I live in **cityName**

Python Types

- Numbers: float, complex, int (+ bool)
- Sequences: list, tuple, str, NumPy arrays
- Mappings: dict (dictionary/hash)
- Instances: user-defined class
- Callables: functions, callable instances

Numerical Expressions

- **Python distinguishes between strings and numbers:**

```
b =  1.2            # b is a number
b = '1.2'           # b is a string
a = 0.5 * b         # illegal: b is NOT converted to float
a = 0.5 * float(b) # this works
```

- **All Python objects are compared with:**

```
==   !=  <  >  <=  >=
```

Boolean Expressions

- bool is True or False
- Can mix bool with int 0 (false) or 1 (true)
- Boolean tests:

```
a = '';  a = [];  a = ();  a = ;  # empty structures
a = 0;   a = 0.0
if a:                             # false
if not a:                         # true
```

other values of a: if a is true

Strings

- **Single- and double-quoted strings work in the same way:**

```
s1 = "some string with a number %g" % r
s2 = 'some string with a number %g' % r      # = s1
```

- **Triple-quoted strings can be multi line with embedded newlines:**

```
text = """
large portions of a text can be conveniently
placed inside triple-quoted strings
(newlines are preserved)"""
```

- **Raw strings, where backslash is backslash:**

```
s3 = r"\(\s+\.\d+\)"
# with ordinary string (must quote backslash):
s3 = '\\(\\s+\\.\\d+\\)'
```

Variables and Data Types

Type	Range	To Define	To Covert
float	numbers	x=1.0	z=float(x)
integer	numbers	x=1	z=int(x)
complex	complex numbers	x=1+3j	z=complex(a,b)
string	text string	x='test'	z=str(x)
boolean	True or False	x=True	z=bool(x)