INTRO TO PYTHON

FOR AOSC

OCT 17, 2016

ABOUT PYTHON

- Python is an *interpreted* programming language.
- In an interpreted language the source code file is read line-by-line and each statement converted into machine language and executed before the next line is read.
- Compiled languages (Fortran) are generally more efficient and faster than interpreted languages, since the conversion to machine language only occurs one time, and then the machine language code can be executed whenever the program needs to be run.
- Interpreted languages are often more flexible and changes can be easily made to the program at runtime.

PYTHON IS OPEN SOURCE & FREE

- Unlike Matlab and IDL, Python is Open Source.
- Python can be installed on any computer.
- The source code is available for you to inspect.
- Both support and Development are community driven

- How to get help?
  - Google “How do I [what you want to do] in Python”
  - PyAOS: Community of users in Atmospheric and Oceanic Science
    - **http://pyaos.johnny-lin.com**

SHARED LIBRARIES

- MATPLOTLIB documentation
  http://matplotlib.sourceforge.net

- Numerical Python (NumPy) documentation page:
  http://docs.scipy.org/doc/numpy/reference

- ScientificPython (SciPy) documentation
  http://docs.scipy.org/doc/scipy/reference

- PIP Packages for Atmospheric Sciences (Link)

PYTHON VERSION

- Python has recently undergone a major redesign.
- Python 3.x
  - Syntax has change and is not compatible with 2.x
- Python 2.x
  - Most commonly used by scientific community
  - MANY MANY libraries written for 2.x few have been migrated to 3.x
- Python 2.7
  - Bridge between the versions
  - Syntax from both will work

PYTHON BASICS

- Comments begin with #
  - Can be at the beginning of the line or in middle
    - #This is a comment
    - A = 45 #This sets the value of A
- Python IS case sensitive
  - Apple = 'red'
  - apple = 67.1
- Names can NOT start with a number
- Beware of “Reserved Words”
- Careful starting or ending names with underscores _
- Variable are empty (None or NULL) until assigned a value.

RUNNING A PYTHON PROGRAM

- python filename.py
  - This will run the program and return you to the command prompt.
- python -i filename.py
  - Runs the program in "interactive" mode.
  - When program ends you will still be in the python environment and can examine the data types and values of variables
- ./filename.py or filename.py (depending on your path)
  - File must be executable
  - 1st line must be #!/usr/bin/python
    - This line must match output of "which python" and may vary from system to system.

DATA TYPING

- Python is dynamically typed.
- Most variables do not need to be declared as real, integer, string, etc.
- The type of a variable is defined by the value assigned to it.
- The type of a variable can change throughout the program execution.
- The opposite of dynamically-typed is *statically-typed*. Fortran and C are examples of statically-typed languages.

DATA TYPES

- Boolean
  - Any non-Zero value becomes True
  - Zero or None is False
- Integer
  - Between -2147483648 and 2147483648
  - Any integer larger, smaller, or ending in a capital L is a Long Integer
- Float
  - Automatically 64-bit or Double Precision
  - E or e indicates Scientific Notation.

CHANGING THE DATA TYPE

- The initial data type is set by the value assigned to it.
- Change to Float
  - x = float(x)
- Change to Integer
  - x = int(x) or x = long(x) for Long Integer
- Change to String
  - x = string(x)
- Change to Boolean
  - x = bool(x)
  - Be careful to remember the boolean rules

STRINGS

- Can be any combination of characters including non-ASCII
- Strings are surrounded by quotes: single, double or triple quotes
  - Triple quotes preserve formatting (''', or """)
- Tab (\t) and new line (\n)
- Escape character is \
  - If you actually want the characters \t to be in the string, not a tab you must "escape" it.
  - Text = "A tab is included in a string as \\t"

FORMATTING STRINGS

- x = 123.456789
- Text = "The value of x is {0:7.2f}".format(x)
  - print Text
  - The value of x is 123.46

- See Page 32 of DeCaria for more examples of the format() method.

LISTS VS ARRAYS

- Lists and Tuples Are NOT the same as Arrays
  - Lists and Tuples can hold a mix of data types
  - All elements of Arrays must be the same data type

- Python does have built in arrays, but we will be using NumPy Arrays

- More on Arrays, List, Truples, etc on Wednesday

NUMERICAL OPERATORS

- + Addition
- - Subtraction
- * Multiplication
- / Division
- // truncating division
  - 16.3 // 5.2 => 3.0
- ** power
  - 2**3 = 8
- % modulo (returns the remainder)
  - 5 % 3 => 2

AUGMENTED ASSIGNMENT OPERATORS

- The augmented assignment operators are shorthand operators and take the form x += y, which is the same as x = x + y.

- This works not only with addition (+), but also with subtraction, multiplication, division, truncated division, and powers.

COMPARISON OPERATORS

- < less than
- > greater than
- == equal to
- != not equal to
- >= greater than or equal to
- <= less than or equal to

- The three Boolean operators in Python are or, and, and not.
  - These can combine multiple comparisons into 1 statement.
  - X < 3 and X > 0.5 and X != 1

FLOW CONTROL

- User input from the command line
  - input = raw_input("prompt string ")

- Concerns with user input
  - Data Type
  - Range
  - Escape characters

FLOW CONTROL

- Conditional Statements
- The conditional ends with a colon and the code block is indented

```
if x<12:
        do this
elif x>=12 and x<30:
        do this
        and this
        and this
else:
        do this
```

FLOW CONTROL

- Conditional Statements
- The conditional ends with a colon and the code block is indented

```
if x:
    do this
else:
    do this
```

FLOW CONTROL

- Loops
- Same format as Conditionals

```
while condition:
        Do this
```

FLOW CONTROL

- Loops
- Same format as Conditionals

```
while x < 10:
        print x
```

FLOW CONTROL

- Loops
- Same format as Conditionals

```
while x < 10:
        print x
        x = x + 1
```

FLOW CONTROL

- Loops
- Same format as Conditionals

```
while x < 10:
        print x
        x += 1
```

FLOW CONTROL

- Loops
- Same format as Conditionals

```
x = 0
while x < 10:
        print x
        x += 1
```

LET'S TRY IT

- Copy ~jeff/aosc652/week08/hello.py

```
end = raw_input('Enter a number between 3 and 10: ')
while n < end:
        print('hello world: {0}'.format(n))
```