CS2204
Program Design and Data Structures for Scientific Computing

Spring 2016

Announcements

- Programming assignments will start next week
- Make sure you have Python and PyCharm installed (or some alternate Python programming environment)
  - Always bring your laptop to class so you can follow along
- We begin our investigation of Python today

Introduction to Python

- Python is an open source scripting/programming language.
- Developed by Guido van Rossum in the early 1990s
- Named after the Monty Python comedy troupe

- Available for download from http://www.python.org

Python vs. Java

Python

- Interpreted
- Dynamically typed
- Concise

```
print(“Hello World”)
```

Java / C++

- Compiled
- Statically typed
- Verbose

```
public class HelloWorld
{
  public static void main (String[] args)
  {
    System.out.println("Hello, world!");
  }
}
```

Python vs. Matlab

Python

- Free, open-source
- No ; syntax
- Better as a general programming language

Matlab

- Commercial product
- Less cumbersome array syntax

  [1 0 0; 0 1 0; 0 0 1] vs
  array([1., 0., 0.], [0.. 1., 0.], [0., 0., 1.])

- Better matrix/vector math and plotting capability

The Python Interpreter

- Python is an interpreted language
- The interpreter provides an interactive environment in which to use the language
- Results of computations and expressions are printed out on the window

```
>>> 3 + 7
10
>>> 3 < 15
True
>>> 'print me'
'print me'
>>> print(‘print me’)
print me
>>>
```

Python Code Example

- Python can also run scripts or programs

```
x = 34 - 23                # A comment
y = "Hello"                # Another one.
z = 3.45
if z == 3.45 or y == "Hello":
    x = x + 1
    y = y + " World"       # String concatenation
print(x)
print(y)
```

- Later, we will see how to store and recall these

Understanding the Code

- Assignment uses `=` and comparison uses `==`.
- For numbers `+ - * / %` are as expected.
  - Special use of `+` for string concatenation.
  - Special use of `%` for string formatting (as with printf in C/Java)
    - Using the `%` string operator in combination with the print command, we can format our output text:

      ```
      >>> print(“%s eggs %d" % ("spam", 42))
      spam eggs 42
      ```
- Logical operators are words (`and, or, not`) *not* symbols
- The basic printing command is `print`.
- The first assignment to a variable creates it.
  - Variable types don’t need to be declared.
  - Python figures out the variable types on its own.

Whitespace

Whitespace is meaningful in Python: especially indentation and placement of newlines.

- Use a newline to end a line of code.
  - Use `\` when must go to next line prematurely.
- No braces `{ }` to mark blocks of code in Python… Use *consistent* indentation instead.
  - The first line with *less* indentation is outside of the block.
  - The first line with *more* indentation starts a nested block
- Often a colon appears at the start of a new block. (E.g., for function and class definitions.)

Comments

- Start comments with # – the rest of line is ignored.

```
>>> 'this will print'
'this will print'
>>> #'this will not'
>>>
```

- Can include a “documentation string” as the first line of any new function or class that you define.
- The development environment, debugger, and other tools use it: it’s good style to include one.

```
def my_function(x, y):
  """This is the docstring. This
    function does blah blah blah."""
    # The code would go here...
```

Variables

- Everything in Python is an object
- All variables contain references to objects
- Variables are not declared, just assigned (like Matlab, unlike Java) – **dynamic typing**
- The variable is created the first time you assign it a value
- Type information is with the object, not the variable/ reference
- A variable does not have a specific type (e.g., integer) and can refer to different types of objects at different times

```
>>> x = 12345     # x holds an integer
>>> x = 12.345    # x now holds a float
>>> x = "12345"   # and now a string
```

Integers

- Integer – the equivalent of a C long
- Long Integer – an unbounded integer value. Newer versions of Python will automatically convert to a long integer if you overflow a normal one

```
>>> 132224
132224
>>> 132323 ** 2
17509376329L
>>>
```

- Long integers represent the large value exactly, but can be slower to operate on.

Floating Point Numbers

- int(x) converts x to an integer
- float(x) converts x to a floating point
- The interpreter may show
  a lot of digits, including the variance in floating point
- To avoid this use “print”

```
>>> 1.23232
1.2323200000000001
>>> print 1.23232
1.23232
>>> 1.3E7
13000000.0
>>> int(2.0)
2
>>> float(2)
2.0
```

Complex Numbers

- Built into Python
- Same operations are supported as integer and float

```
>>> x = 3 + 2j
>>> y = -1j
>>> x + y
(3+1j)
>>> x * y
(2-3j)
```

- Do not get confused by j (a variable) and 1j (a complex number)

Flow Control

- There are several Python expressions that control the flow of a program. All of them make use of Boolean conditional tests.
  - if statements
  - while loops

if Statements

```
if x == 3:
    print(“X equals 3.”)
elif x == 2:
    print(“X equals 2.”)
else:
    print(“X equals something else.”)
print(“This is outside the ‘if'.")
```

Notes:

- Use of indentation for blocks
- Colon (:) after boolean expression and else
- The else clause is optional
  - You can have a simple if, an if/else, or a multibranch if/elif/else
- If statements can be nested inside one another
  - No dangling-else problem like Java/C++

while Loops

```
x = 3
while x < 10:
    x = x + 1
    print(“Still in the loop.”)
print(“Outside of the loop.”)
```

Notes:

- Use of indentation for blocks
- Colon (:) after boolean expression
- Python does not have a post-test (i.e., do-while) loop

while Loops

```
x = 3
while x < 10:
    x = x + 1
    print("Still in the loop.")
else:
    print("Normal exit from the loop.")
print("Outside of the loop.")
```

Notes:

- Optional ‘else’ clause after while loop
  - Executed when the loop test is false before continuing
  - The ‘else’ clause is skipped if a ‘break’ statement is used to exit the loop early

break and continue

- You can use the keyword break inside a loop to leave the while loop entirely.
  - Using break will also skip over the else clause.

- You can use the keyword continue inside a loop to stop processing the current iteration of the loop and to immediately go on to the next one.