
python

Start a conversation:
andy@wi.mit.edu

Hi, I'm Andy!

I've been programming for nearly 20 years.

I love it.

Goals Today:

Learn essential syntax.

Write a few programs.

Understand how to learn more.

What is Python?

An interpreted, object-oriented, high-level programming language with dynamic semantics! (Woah! What?!)

- A formal language.
- Stored in a text file.
- Run by the Python interpreter.

Why Python?

Readability
Lots of code to come.

Community
Mailing lists, Irc, Wiki, Docs.

Libraries (20,114 of them)
Numpy, Biopython, Cogent.

Running Python: Interactively

Interactive Prompt

Only good for fooling around.

```
andy@tak ~$ python
Python 2.6.5 (r265:79063, Apr 16 2010, 13:57:41)
[GCC 4.4.3] on linux2
Type "help", "copyright", "credits"
or "license" for more information.
>>> print "type your commands here"
type your commands here
>>> 2+2
4
```

Running Python: Scripts

Script Files

are specially formatted text files.

say_hello.py

```
#!/usr/bin/env python                  # hashbang (this is a python file)
question = "What is your name?\n"      # assign a variable
name = raw_input(question)             # assign a variable (via user input)
print "Hello " + name + "!"            # print output.
```

shell

```
andy@tak ~$ gedit say_hello.py
andy@tak ~$ chmod +x say_hello.py
andy@tak ~$ ./say_hello.py
What is your name?
Andy<ENTER>
Hello Andy!
```

Syntax

- Comments
- Assignment
- Data Types
- Logical Operators
- Tests / Loops
- Functions

Syntax: Comments

```
# This is a comment.
# Any line that starts with a '#' is.
# They are for your benefit.
# They are ignored by the interpreter.

# When should I use them?
# * At the top of the file describing the goal.
# * At the beginning of a method, or long process.
# * Any time the code isn't obvious,
#   but don't repeat the logic of the code.
```

Syntax: Assignment

```
# Name your data.
# Use meaningful names!

# constants are usually all caps,
# and at the top of the file.
PHI = 1.61803399
SECONDS_PER_DAY = 24*60*60

# regular variables are lower case
# with underscores instead of spaces.
first_name = 'Guido'
last_name = 'van Rossum'

# rules:
#   start with a letter
#   contain only letters, numbers and underscores '_'.
#   cannot be a reserved word (listed later).
```

Syntax: Basic Data Types

```
# String, either "" or ''
course_name = "python"
description = """
a basic introduction
to syntax and scripting
in the python language.
"""

# Integer
lecture_length = 1

# Float (aka fractions)
exercise_length = 0.999
```

```
# List (aka array)
topics = ["syntax",
          "scripts",
          "help"]

# Dictionary (aka hash)
glossary = {
  'key': 'value',
  'python': 'type of snake',
  'five': 5
}
```

Syntax: Logical Operators

equality tests:

```
a == b # equals
a != b # not equals
```

greater/lesser tests

```
>, <, >=, <=
```

composing multiple tests (use parens for clarity)

```
(a == b) and (b == c) # and
(a == b) or (a == c) # or
```

Syntax: Flow Control

```
# if/else control
n = 13
if n % 2 == 1:
    print "Odd Number"
else:
    print "Even Number"
# => "Odd Number"
```

```
# for loop
# count from 5 to 10
for n in range(5,11):
    print n

# while loop
# count from 10 to 1
n = 10
while n > 0:
    print n
    n = n - 1
```

Functions

Built-In
- 80 functions
- available everywhere
- provided by Python

```
abs(-1) # => 1
len([1,2,3]) # => 3
min([2,4,6]) # => 2
max([1,3,5]) # => 5
```

see:
docs.python.org/library/functions.html

Library
- Standard libraries
- Community libraries
- (thousands and growing)

```
math.cos(theta)
scipy.std(numbers)
```

see:
docs.python.org/library
pypi.python.org/pypi

Object
- Provided by the object

```
"String".upper()
"String".find("ring")
[1,2,3].append(4)
```

Functions for Numbers

Operators

```
1 + 1    # => 2
1 - 1    # => 0
3 * 2    # => 6
3 / 2    # => 1 (int math)
3.0 / 2  # => 1.5 (phew!)
3 % 2    # => 1 (modulous)
2**8     # => 256 (exponent)
```

Functions

```
# changing between types
float(1) # => 1.0
int(1.7) # => 1 # floors!
str(5)   # => '5'
chr(65)  # => 'A'

round(1.5) # => 2.0
abs(-1)    # => 1
pow(2,8)   # => 256
```

Functions for Lists

```
# lists, sorted, indexed
first_5 = range(1,6) # => [1,2,3,4,5]
2 in first_5 # => True
[1,2] + [3,4] # => [1,2,3,4]
list("str") # => ['s','t','r']

min(first_5) # => 1
max(first_5) # => 5
len(first_5) # => 5
```

Functions for Lists (indexing)

```
residues = list('GTCA') # => ['G','T','C','A']
# indexes start at 0,     e.g. 0   1   2   3

# residues[start:stop:step]
#    start - defaults to 0
#    stop - blank means the end of the string
#    step - defaults to 1 (use -1 to go backwards)
residues[0] # 'G' first char
residues[0:2] # 'GT' first two chars
residues[:3] # 'GTC' first three
residues[2:] # 'CA' last two chars
residues[::-1] # 'ACTG', reverse of the string
```

Functions for Strings

```
string = 'Go to work\n'

string = string.replace('work','the park')
                # => 'Go to the park\n'

string.lower() # => 'go to the park\n'

string.upper() # => 'GO TO THE PARK\n'

string.count('t') # => 2

string.index('p') # => 10

string.rstrip('\n') # => 'Go to the park'


# Strings can also be treated as lists.
# So all the List indexes work.

string[10,4] # => 'park'
```

Functions for Dictionaries

```
# WARNING: Bad variable names! Don't do this.
d = {'A':1, 'B':2}
d['A']       # => 1
d.keys()     # => ['A','B']
d.values()   # => [1,2]
d['C'] = 3

# keys are not sorted
print d      # => {'A': 1, 'C': 3, 'B': 2}
```

Functions for you!

```
def is_odd(n):
    # check if n is odd
    return n % 2 == 1

is_odd(5) # => True

def is_even(n):
    return not is_odd(n)

is_even(2) # => True
```

Whitespace & Colons

- Whitespace is used for grouping blocks.
  4 spaces, or a tab (be consistent)
- Colons end all lines that begin indented blocks.
- **`def`**, **`if`**, **`elsif`**, **`for`**, **`while`**, `and` **`class`** are the keywords that begin new blocks.

Libraries

- Standard
- Community
- Usage

Libraries: Standard

do more, write less

Standard Libraries

String Services: string, re, struct, difflib, StringIO, cStringIO, textwrap, codecs, unicodedata, stringprep, fpformat

Data Types: datetime, calendar, collections, heapq, bisect, array, sets, sched, mutex, Queue, weakref, UserDict, UserList, UserString, types, new, copy, pprint, repr

Numeric and Mathematical Modules: numbers, math, cmath, decimal, fractions, random, itertools, functools, operator

File and Directory Access: os.path, fileinput, stat, statvfs, filecmp, tempfile, glob, fnmatch, linecache, shutil, dircache, macpath

Data Persistence: pickle, cPickle, copy_reg, shelve, marshal, anydbm, whichdb, dbm, gdbm, dbhash, bsddb, dumbdbm, sqlite3

Data Compression and Archiving: zlib, gzip, bz2, zipfile, tarfile

File Formats: csv, ConfigParser, robotparser, netrc, xdrlib, plistlib, Cryptographic Services, hashlib, hmac, md5, sha

Generic Operating System Services: os, io, time, argparse, optparse, getopt, logging, logging.config, logging.handlers, getpass, curses, curses.textpad, curses.ascii, curses.panel, platform, errno, ctypes

Optional Operating System Services: select, threading, thread, dummy_threading, dummy_thread, multiprocessing, mmap, readline, rlcompleter

Interprocess Communication and Networking: subprocess, socket, ssl, signal, popen2, asyncore, asynchat

Internet Data Handling: email, json, mailcap, mailbox, mhlib, mimetools, mimetypes, MimeWriter, mimify, multifile, rfc822, base64, binhex, binascii, quopri, uu

Structured Markup Processing Tools: HTMLParser, sgmllib, htmllib, htmlentitydefs, xml.parsers.expat, xml.dom, xml.dom.minidom, xml.dom.pulldom, xml.sax, xml.sax.handler, xml.sax.saxutils, xml.sax.xmlreader, xml.etree.ElementTree,

Internet Protocols and Support: webbrowser, cgi, cgitb, wsgiref, urllib, urllib2, httplib, ftplib, poplib, imaplib, nntplib, smtplib, smtpd, telnetlib, uuid, urlparse, SocketServer, BaseHTTPServer, SimpleHTTPServer, CGIHTTPServer, cookielib, Cookie, xmlrpclib, SimpleXMLRPCServer, DocXMLRPCServer

Multimedia Services: audioop, imageop, aifc, sunau, wave, chunk, colorsys, imghdr, sndhdr, ossaudiodev

Details at http://docs.python.org/library/

Libraries: Community

Community

SciPy - Scientific Tools

Biopython - Biological Computation Tools

PyCogent - Genomic Biology

Among 20,114 available libraries (vs perl's 104,923)

See http://pypi.python.org/pypi for more

Libraries: Usage

- `import library_name`
  makes methods available via the library_name prefix

```
import math
math.pi # => 3.141592653589793
```

- `from library_name import function,function,...`
  Makes specified functions available without the prefix

```
from math import cos
cos(math.pi) # => -1.0
```

Filesystem Input/Output

```
import os
filename = '/tmp/testfile'

# write a file
if not os.path.isfile(filename):
    out = open(filename,'w')
    out.write("new file contents")
    out.close()

# read a file
if os.path.isfile(filename):
    for line in open(filename):
        print line # => "new file contents"
```

Composing a script

Let's take what we've learned
and see it do something

DNA Complement

```
#!/usr/bin/env python
import fileinput
# usage: ./dna_complement_manual.py input.fasta
# std. dna complements
reverse = { 'A':'T', 'C':'G', 'G':'C', 'T':'A' }
for line in fileinput.input():
    line = line.rstrip('\n')
    if line: # skip empty lines
        if line[0] == ">": # info lines
            print line
        else: # reverse complement current line
            complement = ""
            for letter in line:
                complement += reverse[letter]
            print complement
```

DNA Complement: Results

```
% head -n 3 NC_003279.6.fasta
>gi|193203938:4762885-4772799 Caenorhabditis elegans chromosome I, complete sequence
TCGAAGAATCGCATAAACTCCGAAACTTTAATTTTTTTAAGTTCATTGCCCGAGAGGAGAACACGGCCGA
GAATCTGAAAAATCATTTGCACGCGGAATTCAAATTAGATCGAGGAAAAGAGTAGTATTTGGAACTTTGT

% python examples/dna_complement_manual.py NC_003279.6.fasta | head -n 3
>gi|193203938:4762885-4772799 Caenorhabditis elegans chromosome I, complete sequence
AGCTTCTTAGCGTATTTGAGGCTTTGAAATTAAAAAAATTCAAGTAACGGGCTCTCCTCTTGTGCCGGCT
CTTAGACTTTTTAGTAAACGTGCGCCTTAAGTTTAATCTAGCTCCTTTTCTCATCATAAACCTTGAAACA
```

DNA complement: with a library

```
#!/usr/bin/env python
import fileinput
from Bio import SeqIO

for record in SeqIO.parse(fileinput.input(),'fasta'):
    record.seq = record.seq.complement()
    print record.format('fasta')
```

Learning More

- Commandline
- On the Internet
- On paper
- Intermediate Topics

Learning More: Commandline

Pydoc Quick Reference (lookup any term)

```
% pydoc int
Help on class int in module __builtin__:

class int(object)
| int(x[, base]) -> integer
|
| Convert a string or number to an integer, if possible. A floating point
| argument will be truncated towards zero (this does not include a string
| representation of a floating point number!) When converting a string, use
| the optional base. It is an error to supply a base when converting a
| non-string. If base is zero, the proper base is guessed based on the
| string content. If the argument is outside the integer range a
| long object will be returned instead.
|
| Methods defined here:
|
| __abs__(...)
| x.__abs__() <==> abs(x)
|
| __add__(...)
| x.__add__(y) <==> x+y
...
```

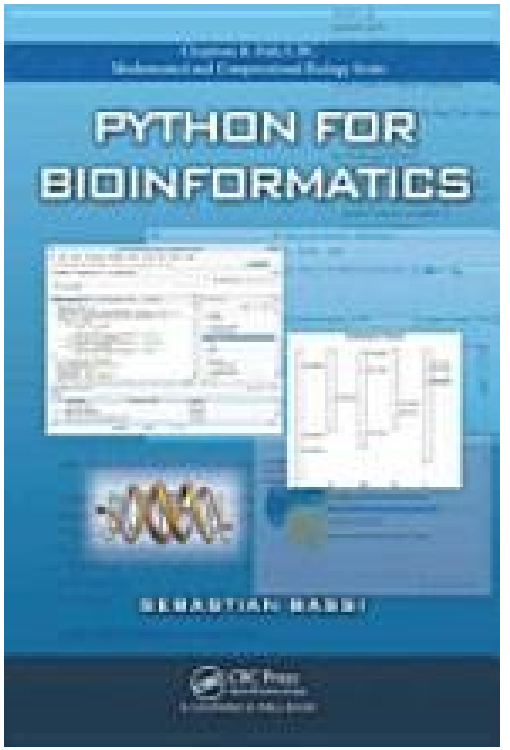
Learning More: Internet

- The Python Wiki
  http://wiki.python.org/moin/
- A Gentle Introduction to Programming Using Python
  MIT Courseware
  http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-189-a-gentle-introduction-to-programming-using-python-january-iap-2011/index.htm
- A Primer on Python for Life Science Researchers
  by Sebastian Bassi
  http://www.ploscollections.org/article/info%3Adoi%2F10.1371%2Fjournal.pcbi.0030199

Learning More: Paper

Learning Python
by Mark Lutz
ISBN 0596158068

Programming Python
by Mark Lutz
ISBN 0596158106



Python for Bioinformaticists
By Sebastian Bassi
ISBN 1584889292

Learning More: Intermediate Topics

- **Classes**
  build your own objects
- **Revision Control**
  keep a history of code changes (subversion)
- **Python aware Editor**
  syntax, error help (idle-python, eclipse, etc.)

Thank you.

- Next: hands on scripting
  - DNA Reverse Complement
  - FASTQ De-duplication by quality

Hands on scripting!

- ssh -Y username@tak
- wget http://jura/bio/education/hot_topics/Unix_Perl_Python/python_lecture_files.tar.gz
- tar xvfz python_lecture_files.tar.gz
- cd python_lecture_files/exercise1
- idle-python2.6 dna_complement_reverse.py

Appendix: Installing

- Windows:
  - Cygwin cygwin.com
- Linux
  - (debian): `sudo apt-get install python2.7`
  - (fedora): `sudo yum install python`
- Mac
  - Comes with an old version
  - Upgrade instructions at python.org/getit/mac

Appendix: Python Editor

IDLE-Python knows about python.

It executes scripts and helps clarify errors.

```
andy@home:~$ ssh user@tak -Y
andy@tak:~$ idle-python2.6
```

Appendix: Reserved Words

```
and       assert    break     class     continue
def       del       elif      else      except
exec      finally   for       from      global
if        import    in        is        lambda
not       or        pass      print     raise
return              try       while
```

Do not try to use these as variable names.

Also avoid type names, library names and common methods too
e.g. `Float, Int, Numeric, math, range, cos, pi,` etc.