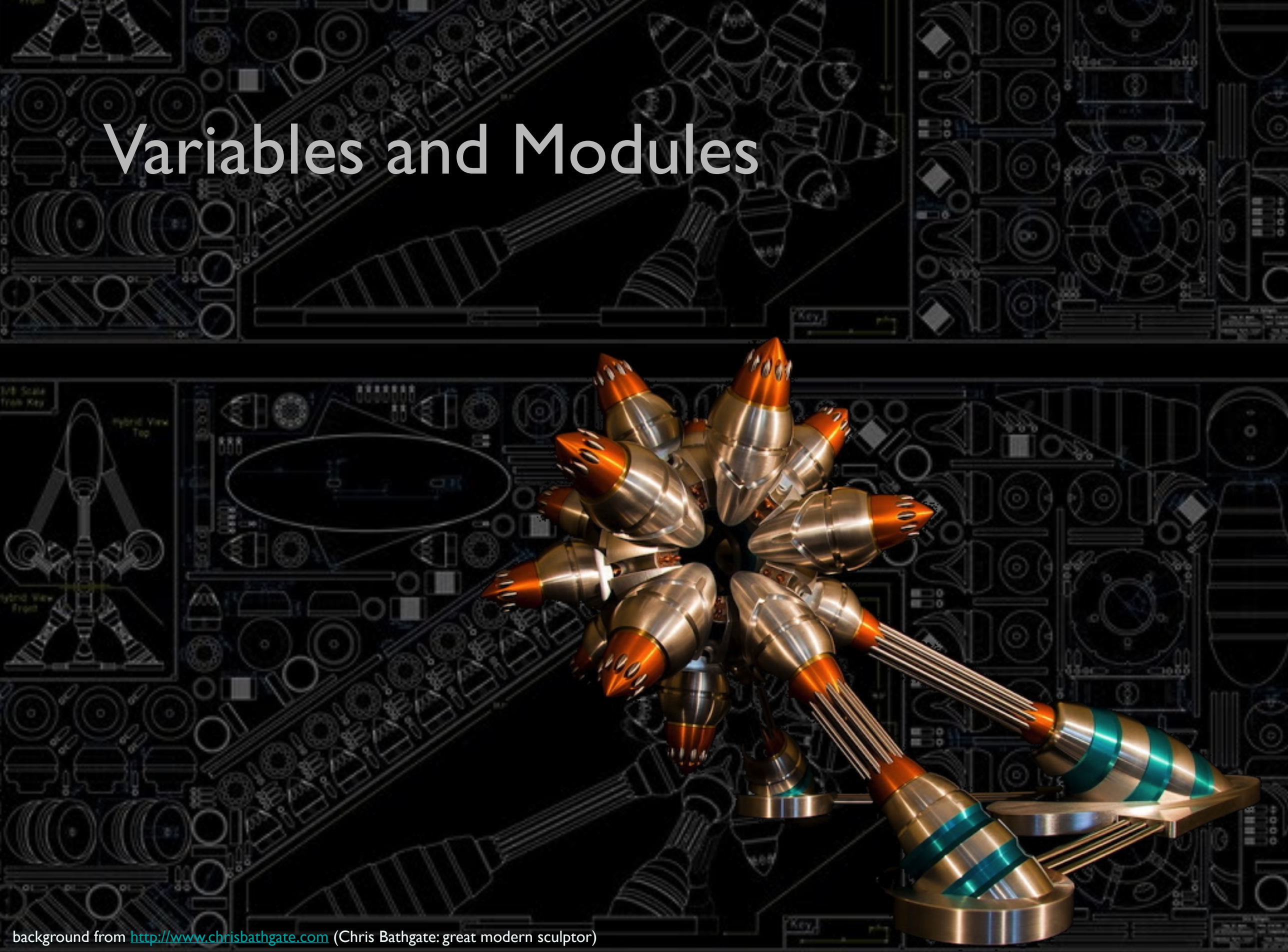
Variables and Modules



background from http://www.chrisbathgate.com (Chris Bathgate: great modern sculptor)

Variables in Context

- All variables inside a function are local
- Global variables can be accessed but not changed except via the *global* declaration
  (the web: "If you want to simply access a global variable you just use its name. However to **change** its value you need to use the global keyword.")

- A global variable can be defined from within a function!

```
def example():
    global v
    v = 5

print v          # returns 5. Exception without global
```

```
total = 9
globvar = 11

def add(a, b):
    total = 3       # does not affect global variable
    return a+b+total+globvar

def add_glob(a, b):
    global total
    total = 3       # changes the global variable total
    return a+b+total

print add(3,5)       # returns 22 (3+5+3+11)
print total          # returns 9
print add_glob(3,5)  # returns 17
print total          # returns 3
```

Variable Types

- locals() : local variables
- globals() : global variables
- vars(obj) : object dictionary
- vars() : same as locals()

```
total, newvar = 9, 11

def add(a, b):
    total = 3
    print "locals= ", locals()
    print "globals= ", globals()
    return a+b+total+newvar

def add_glob(a, b):
    global total
    total = 3  # changes the global total
    print "locals= ", locals()
    print "globals= ", globals()
    return a+b+total

add(3,4)
print "------------------------"
add_glob(3,4)
```

Output

```
locals=  {'a': 3, 'total': 3, 'b': 4}
globals=  {'add_glob': <function
add_glob at 0x379830>, '__builtins__':
<module '__builtin__' (built-in)>,
'__file__': 'context.py', '__package__':
None, 'add': <function add at
0x379870>, '__name__': '__main__',
'total': 9, 'newvar': 11, '__doc__': None}
------------------------
locals=  {'a': 3, 'b': 4}
globals=  {'add_glob': <function
add_glob at 0x379830>, '__builtins__':
<module '__builtin__' (built-in)>,
'__file__': 'context.py', '__package__':
None, 'add': <function add at
0x379870>, '__name__': '__main__',
'total': 3, 'newvar': 11, '__doc__': None}
```

Variables

- Way to label and access information
- Naming variables
  - A variable name can contain only numbers, letters and underscores.
  - A variable name cannot start with a number.
- Guidelines
  - Choose descriptive names (ex. ***score*** instead of ***s***)
  - Be consistent with multiword names (***high_score*** or ***highscore***)
  - Follow the traditions of the language
    - variables start with a lowercase letter
    - avoid using an underscore as the first character
  - Keep the length in check
    - try to keep names under 15 characters

Python's Keywords

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

- Can't use these keywords as variable names.

```
In [28]: and=1
  File "<ipython-input-28-2b18fd315f8e>", line 1
    and=1
      ^
SyntaxError: invalid syntax
```

Numeric Variable Types

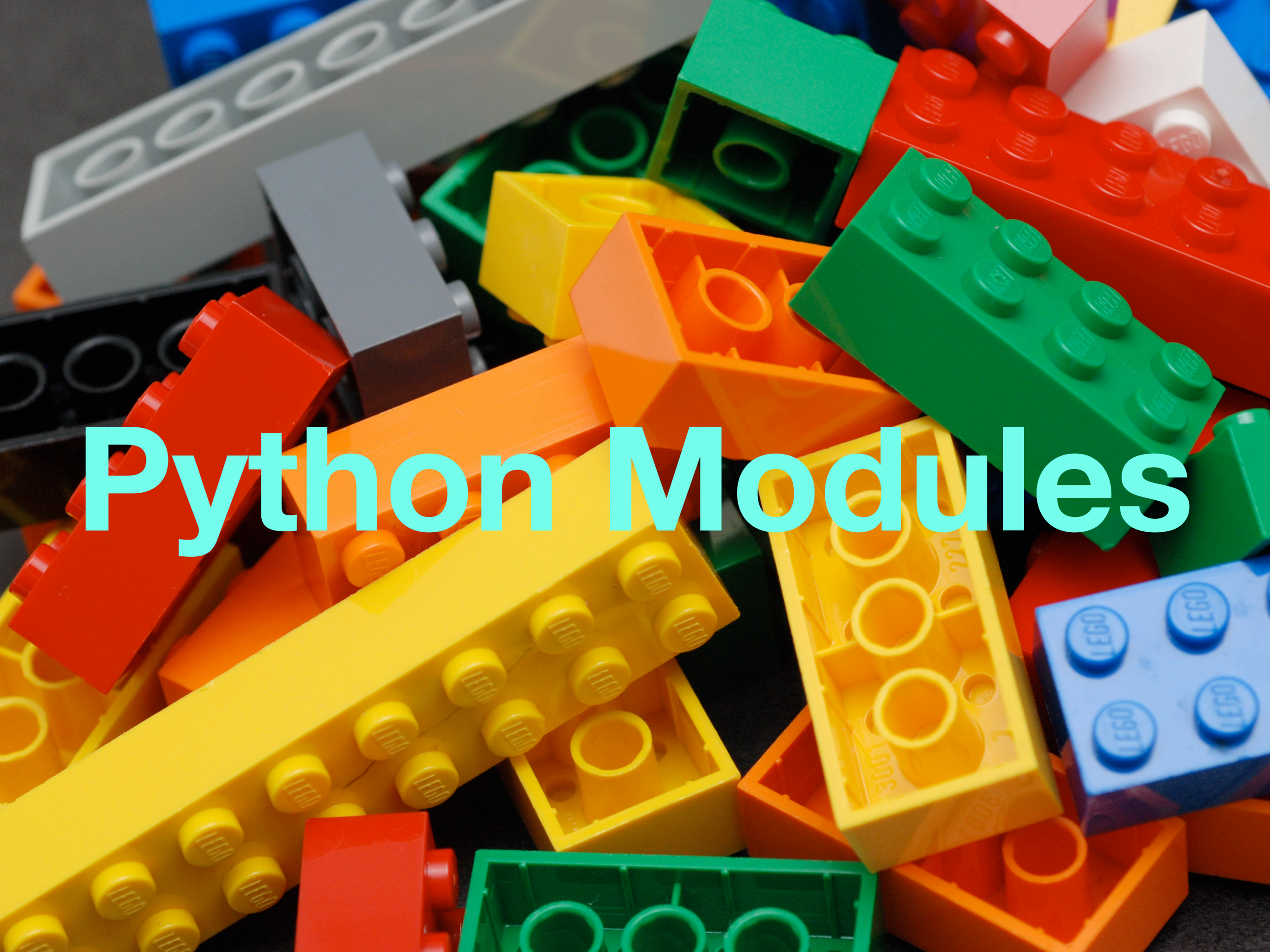
- In languages like C/C++/Java, one has 4 byte ints, 8 byte longs, 4 byte floats, 8 byte doubles (on a 32-bit machine)
- In Python
  - int
    - 4 bytes on 32-bit machines,
    - 8-bytes on 64-bit machine
  - long
    - infinite precision
    - sys.maxint (on my mac) returns 9223372036854775807 = 2**63 - 1
  - float
    - equivalent to double in C++, so 8 bytes on 32-bit machine
  - complex (**j** represents cmath.sqrt(-1))
    - a = 1+3j

Digression

- One byte = 8 bits
- one bit: 0 or 1
- two bits: 00, 01, 10, 11 = 0, 1, 2, 3
  - represents the integers 0 to 3=2**2-1
    (** is exponentiation), or integers -2,0,1,2 = [-2**1,2**1-1]
- 3 bits: 000, 001, 010, 011, 100, 101, 110, 111
  - represents the positive integers 0 to 7=2**3-1 or
    pos/neg integers [-2**2, 2**2-1] = [-4,3]
- 8 bits represents integers 0 to 2**8-1 = 255 or [-128,127]
- 32 bits represents ints 0 to 2**32-1 or [-2**31, 2**31-1] = **2147483647L**
- **64** bits represents ints 0 to 2**63-1 or [-2**63, 2**63-1] =
  [**0,9223372036854775807L]**
- sys.maxint = **9223372036854775807**

What is going on?

- 2147483647 versus 2147483647**L**
- An **L** denotes *long,* which means infinite precision
- Given that 2**31-1 fits into 4 bytes, why is there a **L**?
  - ON 32bit machines:
    it is because 2**31 does *not* fit into 4 bytes, and it must be calculated first.
  - second step: subtract 1 from a long; therefore, Python does automatic format conversion from the longer format to the shorter one (not to lose precision).
  - The final result is *not* converted back to int size.



Python Modules

OS module
of interest **in blue**

Stored in list: [a,b,c]

```
>>> import os
>>> dir(os)
['EX_CANTCREAT', 'EX_CONFIG', 'EX_DATAERR', 'EX_IOERR', 'EX_NOHOST', 'EX_NOINPUT', 'EX_NOPERM',
'EX_NOUSER', 'EX_OK', 'EX_OSERR', 'EX_OSFILE', 'EX_PROTOCOL', 'EX_SOFTWARE', 'EX_TEMPFAIL',
'EX_UNAVAILABLE', 'EX_USAGE', 'F_OK', 'NGROUPS_MAX', 'O_APPEND', 'O_ASYNC', 'O_CREAT',
'O_DIRECTORY', 'O_EXCL', 'O_EXLOCK', 'O_NDELAY', 'O_NOCTTY', 'O_NOFOLLOW', 'O_NONBLOCK',
'O_RDONLY', 'O_RDWR', 'O_SHLOCK', 'O_SYNC', 'O_TRUNC', 'O_WRONLY', 'P_NOWAIT', 'P_NOWAITO',
'P_WAIT', 'R_OK', 'SEEK_CUR', 'SEEK_END', 'SEEK_SET', 'TMP_MAX', 'UserDict', 'WCONTINUED', 'WCOREDUMP',
'WEXITSTATUS', 'WIFCONTINUED', 'WIFEXITED', 'WIFSIGNALED', 'WIFSTOPPED', 'WNOHANG', 'WSTOPSIG',
'WTERMSIG', 'WUNTRACED', 'W_OK', 'X_OK', '_Environ', '__all__', '__builtins__', '__doc__', '__file__', '__name__',
'__package__', '_copy_reg', '_execvpe', '_exists', '_exit', '_get_exports_list', '_make_stat_result', '_make_statvfs_result',
'_pickle_stat_result', '_pickle_statvfs_result', '_spawnvef', 'abort', 'access', 'altsep', 'chdir', 'chmod', 'chown', 'chroot',
'close', 'closerange', 'confstr', 'confstr_names', 'ctermid', 'curdir', 'defpath', 'devnull', 'dup', 'dup2', 'environ', 'errno',
'error', 'execl', 'execle', 'execlp', 'execlpe', 'execv', 'execve', 'execvp', 'execvpe', 'extsep', 'fchdir', 'fchmod', 'fchown', 'fdopen',
'fork', 'forkpty', 'fpathconf', 'fstat', 'fstatvfs', 'fsync', 'ftruncate', 'getcwd', 'getcwdu', 'getegid', 'getenv', 'geteuid', 'getgid',
'getgroups', 'getloadavg', 'getlogin', 'getpgid', 'getpgrp', 'getpid', 'getppid', 'getsid', 'getuid', 'isatty', 'kill', 'killpg', 'lchmod',
'lchown', 'linesep', 'link', 'listdir', 'lseek', 'lstat', 'major', 'makedev', 'makedirs', 'minor', 'mkdir', 'mkfifo', 'mknod',
'name', 'nice', 'open', 'openpty', 'pardir', 'path', 'pathconf', 'pathconf_names', 'pathsep', 'pipe', 'popen', 'popen2',
'popen3', 'popen4', 'putenv', 'read', 'readlink', 'remove', 'removedirs', 'rename', 'renames', 'rmdir', 'sep', 'setegid',
'seteuid', 'setgid', 'setgroups', 'setpgid', 'setpgrp', 'setregid', 'setreuid', 'setsid', 'setuid', 'spawnl', 'spawnle', 'spawnlp',
'spawnlpe', 'spawnv', 'spawnve', 'spawnvp', 'spawnvpe', 'stat', 'stat_float_times', 'stat_result', 'statvfs', 'statvfs_result',
'strerror', 'symlink', 'sys', 'sysconf', 'sysconf_names', 'system', 'tcgetpgrp', 'tcsetpgrp', 'tempnam', 'times', 'tmpfile',
'tmpnam', 'ttyname', 'umask', 'uname', 'unlink', 'unsetenv', 'urandom', 'utime', 'wait', 'wait3', 'wait4', 'waitpid', 'walk',
'write']
```

Module sys

```
>>> dir(sys)
['__displayhook__', '__doc__', '__excepthook__', '__name__',
'__package__', '__stderr__', '__stdin__', '__stdout__', '_clear_type_cache',
'_current_frames', '_getframe', 'api_version', 'argv',
'builtin_module_names', 'byteorder', 'call_tracing', 'callstats', 'copyright',
'displayhook', 'dont_write_bytecode', 'exc_clear', 'exc_info', 'exc_type',
'excepthook', 'exec_prefix', 'executable', 'exit', 'flags', 'float_info',
'getcheckinterval', 'getdefaultencoding', 'getdlopenflags',
'getfilesystemencoding', 'getprofile', 'getrecursionlimit', 'getrefcount',
'getsizeof', 'gettrace', 'hexversion', 'last_traceback', 'last_type', 'last_value',
'maxint', 'maxsize', 'maxunicode', 'meta_path', 'modules', 'path',
'path_hooks', 'path_importer_cache', 'platform', 'prefix', 'ps1', 'ps2',
'py3kwarning', 'setcheckinterval', 'setdlopenflags', 'setprofile',
'setrecursionlimit', 'settrace', 'stderr', 'stdin', 'stdout', 'subversion',
'version', 'version_info', 'warnoptions']
```

Sys within ipython

```
In [21]: import sys

In [22]: print sys.platform
linux2

In [23]: print sys.stdout
<open file '<stdout>', mode 'w' at 0x324078>

In [24]: sys.argv
Out[24]: ['/usr/local/bin/ipython']
```

List of directories to search for modules

```
In [1]: import sys
In [2]: sys.path
Out[2]:
['',
'/Library/Frameworks/Python.framework/Versions/6.3/bin',
'/Applications/Panda3D/1.6.2/lib', '/Users/erlebach/Downloads', '/Library/Frameworks/Python.framewo
Versions/6.3/lib/python26.zip', '/Library/Frameworks/Python.framework/Versions/6.3/lib/python2.6', '/
Library/Frameworks/Python.framework/Versions/6.3/lib/python2.6/plat-darwin', '/Library/Frameworks/
Python.framework/Versions/6.3/lib/python2.6/plat-mac', '/Library/Frameworks/Python.framework/
Versions/6.3/lib/python2.6/plat-mac/lib-scriptpackages', '/Library/Frameworks/Python.framework/Versi
6.3/lib/python2.6/lib-tk', '/Library/Frameworks/Python.framework/Versions/6.3/lib/python2.6/lib-old', '/
Library/Frameworks/Python.framework/Versions/6.3/lib/python2.6/lib-dynload', '/Library/Frameworks/
Python.framework/Versions/6.3/lib/python2.6/site-packages', '/Library/Frameworks/Python.framework/
Versions/6.3/lib/python2.6/site-packages/PIL', '/Library/Frameworks/Python.framework/Versions/6.3/lib
python2.6/site-packages/IPython/Extensions',
u'/Users/erlebach/.ipython']
```

results in a list

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

def print_args(args):
    n = len(args)

    if "-h" in args:
        print "Syntax: is python arguments.py arg1 arg2, ...."
        sys.exit(-1)

    for i,ai in enumerate(args):
        print "using enumerate", i,ai

    for i,ai in zip(range(n),args):
        print "using zip", i, ai


if __name__ == "__main__":
    print_args(sys.argv)
```

Environment variables

stored in dictionary

```
>>> import(os)
>>> print os.environ
{'_': '/usr/bin/python', 'PROMPT_COMMAND': 'echo -ne "\\033]0;${USER}@${HOSTNAME%%.*}:${PWD/
#$HOME/~}\\007"', '__CF_USER_TEXT_ENCODING': '0x1F5:0x0:0x0',
'VERSIONER_PYTHON_PREFER_32_BIT': 'no', 'TERM_PROGRAM_VERSION': '400', 'LOGNAME': 'beerli',
'USER': 'beerli', 'GMTHOME': '/usr/local/gmt', 'HOME': '/Users/beerli', 'PATH': '.:/Users/
beerli/bin:/usr/local/bin:/usr/bin:/usr/local/openmpi/bin:/bin:/usr/local/sbin:/usr/sbin:/
sbin:/opt/X11/bin:/Library/TeX/texbin:/usr/local/cuda/bin:/usr/local/gmt/bin:/usr/local/gmt:/
usr/texbin:/usr/local/mysql-5.1.36-osx10.5-x86/bin:/Users/beerli/Sources/baliphy/bin:/
Applications/BEAST-1.8.0/bin:/usr/local/git/bin:/usr/local/visIt/bin:/usr/local/genome_tools/
bin::/usr/local/genome_tools/MUMmer3.23:/Users/beerli/src/checker-276:/Applications/
Julia-0.2.1.app/Contents/Resources/julia/bin:/Users/beerli/src/bowtie2-2.2.3:/usr/local/
share/python:/usr/local/migrate/bin:/usr/local/go/bin', 'DISPLAY': '/private/tmp/
com.apple.launchd.FjRORBrxkS/org.macosforge.xquartz:0', 'TERM_PROGRAM': 'Apple_Terminal',
'LANG': 'en_US.UTF-8', 'HISTCONTROL': 'ignoredups', 'TERM': 'xterm', 'SHELL': '/bin/bash',
'FASTMIGRATE': 'installed', 'VERSIONER_PYTHON_VERSION': '2.7', 'SHLVL': '1',
'SECURITYSESSIONID': '186a7', 'XPC_FLAGS': '0x0', 'EDITOR': 'emacs', 'MANPATH': '/usr/local/
man:/usr/share/man', 'papers': '/Users/beerli/Documents/Work/manuscripts-working',
'SVN_EDITOR': 'emacs', 'TERM_SESSION_ID': 'A2B0B3E1-0103-43E0-A759-7910459F7CE5',
'XPC_SERVICE_NAME': '0', 'PYTHONPATH': '/usr/local/lib/python2.7/site-packages:/Library/
Python/2.7/site-packages:/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/
site-packages', 'SSH_AUTH_SOCK': '/private/tmp/com.apple.launchd.1LxcXmF3Wn/Listeners',
'Apple_PubSub_Socket_Render': '/private/tmp/com.apple.launchd.ESW59I9nkO/Render', 'TMPDIR':
'/var/folders/j5/5fmljc597bs2dshlnskxrz480000gn/T/', 'PERL5LIB': '/Users/beerli/src/genomics/
vcftools/lib/perl5/site_perl', 'OLDPWD': '/Users/beerli/Documents/Work/talks/CLASSES/
ISC-4304/prep', 'reviews': '/Users/beerli/Documents/Work/reviews', 'PWD': '/Users/beerli/
Documents/Work/talks/CLASSES/ISC-4304/prep/class', 'SAMTOOLS_HOME': '/Users/beerli/src/
genomics/samtools-0.1.19'}
```

math/cmath modules

- math: real numbers
- cmath: complex numbers

```
>>> import math
>>> dir(math)
['__doc__', '__file__', '__name__', '__package__', 'acos', 'acosh', 'asin', 'asinh', 'atan',
'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'exp', 'fabs', 'factorial', 'floor',
'fmod', 'frexp', 'fsum', 'hypot', 'isinf', 'isnan', 'ldexp', 'log', 'log10', 'log1p', 'modf', 'pi', 'pow',
'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']

>>> import cmath
>>> dir(cmath)
['__doc__', '__file__', '__name__', '__package__', 'acos', 'acosh', 'asin', 'asinh', 'atan',
'atanh', 'cos', 'cosh', 'e', 'exp', 'isinf', 'isnan', 'log', 'log10', 'phase', 'pi', 'polar', 'rect', 'sin', 'sinh',
'sqrt', 'tan', 'tanh']
```

import math

```
In [11]: import math

In [12]: math.cos(2.)
Out[12]: -0.4161468365471424 1

In [13]: math.cos(2.+3j)
---------------------------------------------------------------------TypeError
Traceback (most recent call last)
/Users/erlebach/Documents/sc/courses/
programming_scientific_applications_s2011/code/<ipython console> in
<module>()

TypeError: can't convert complex to float

In [14]:
```

Complex arithmetic not allowed

Uncaught exception ==> code crash

import cmath

Complex math module

```
In [14]: import cmath

In [15]: cmath.cos(2.)
Out[15]: (-0.41614683654714241-0j)

In [16]: cmath.cos(2.+3j)
Out[16]: (-4.189625690968807-9.109227893755337j)
```

Name conflicts

```
>>> import math
>>> import cmath
>>> math.cos(3)
-0.9899924966004454
>>> cmath.cos(3)
(-0.9899924966004454-0j)
>>>
```

Identical function names in each module: **cos**

However, module names are different:

math, cmath

Modules
Simplified Usage

- Shortcut names:
  - import math as m
  - import cmath as cm # cm.sqrt(3.45j)
- Import selected commands
  - from math import sqrt
    - only sqrt is available
- Import everything from module
  - from math import * # sqrt(3.45)

However,
there is danger ...

Don't Do This

```
>>> from cmath import *
>>> cos(3j)(10.067661995777765-0j)
>>>
>>> from math import *
>>> cos(3j)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: can't convert complex to float
>>>
```

Overwrite complex functions

Useful Modules

- Construct a python file (a.py)
- Define a simple function inside a.py (print the argument)
- Construct b.py, and *import a*
- Function *a.fun* is now available within b.py
- dir(a) # contents of a

```
In [2]: dir(a)
Out[2]:
['__builtins__',
 '__doc__',
 '__file__',
 '__name__',
 '__package__',
 'myprint',
 'myreverse']
```

a.py

```
#!/usr/bin/env python
# simple module
# contains two functions
# - prints out argument
# - reverse text

def myprint(a):
    print a

def myreverse(a):
    print a[::-1]

if __name__ == "__main__":
    myprint("testing, testing")
    myreverse("testing, testing")
```

b.py

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

if __name__ == "__main__":
    a.myprint("Hello world")
    a.myreverse("Hello World")
```

Doc Strings

iPython

import a
help a

```
Help on module a:

NAME
    a

FILE
    /Users/beerli/ISC-4304/lectures/L2/a.py

DESCRIPTION
    # simple module
    # prints out argument and reverse text

FUNCTIONS
    myprint(a)
        Prints out its argument

    myreverse(a)
        prints out reverse text
```

a.py

```
#!/usr/bin/env python
# simple module
# contains two functions
# - prints out argument
# - reverse text

def myprint(a):
    """
    Prints out its argument
    """
    print a

def myreverse(a):
    """
    prints out reverse argument
    """
    print a[::-1]

if __name__ == "__main__":
    myprint("testing, testing")
    myreverse("testing, testing")
```

Use triple quotes
String as first line of module and first line of any function

Running a script.py file

- From the command line
  - python script.py
  - python ./a/b/script.py
  - python /full_path/script.py
  - ./script.py
  - ./a/b/script.py
- From within the python shell
  - execfile('script.py')
  - execfile('./a/b/script.py')
  - execfile('full_path/script.py')

Calls without python require that the location of python binary be in the first line

#!/usr/bin/env python

and that the file is executable on commandline:

chmod +x file.py

Experiments

- Experiment with
  - changing directories
  - moving/copying files
  - creating directories
  - creating temporary files
  - real and complex arithmetic
  - explore sys