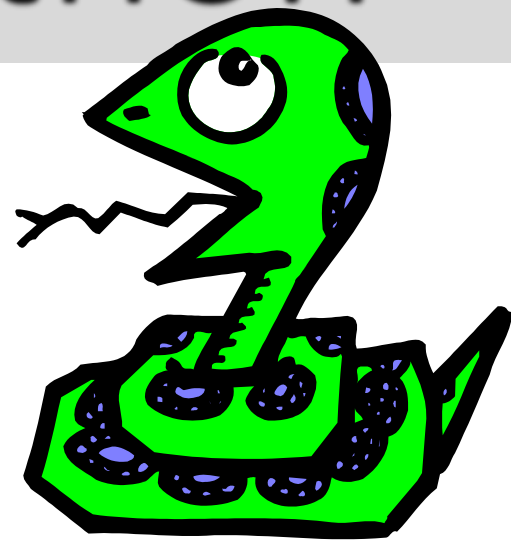
Running Python



The Python Interpreter

- Typical Python implementations offer both an interpreter and compiler
- Interactive interface to Python with a read-eval-print loop

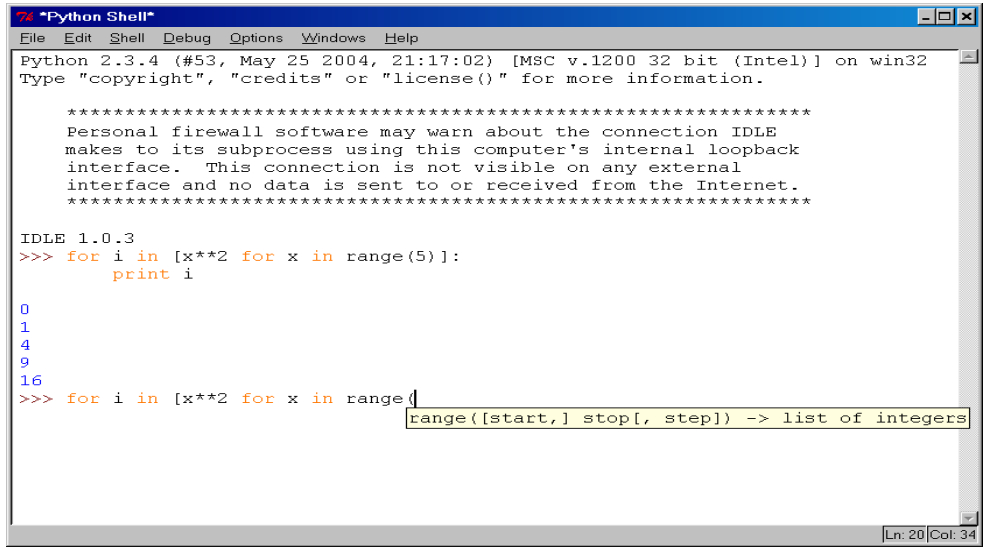
```
[finin@linux2 ~]$ python
Python 2.4.3 (#1, Jan 14 2008, 18:32:40)
[GCC 4.1.2 20070626 (Red Hat 4.1.2-14)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> def square(x):
...   return x * x
...
>>> map(square, [1, 2, 3, 4])
[1, 4, 9, 16]
>>>
```

Installing

- Python is pre-installed on most Unix systems, including Linux and MAC OS X
- The pre-installed version may not be the most recent one (2.6.2 and 3.1.1 as of Sept 09)
- Download from http://python.org/download/
- Python comes with a large library of standard modules
- There are several options for an IDE
  - IDLE – works well with Windows
  - Emacs with python-mode or your favorite text editor
  - Eclipse with Pydev (http://pydev.sourceforge.net/)

IDLE Development Environment

- IDLE is an Integrated DeveLopment Environ-ment for Python, typically used on Windows
- Multi-window text editor with syntax highlighting, auto-completion, smart indent and other.
- Python shell with syntax highlighting.
- Integrated debugger with stepping, persis-tent breakpoints, and call stack visi-bility



Editing Python in Emacs

- Emacs *python-mode* has good support for editing Python, enabled enabled by default for .py files
- Features: completion, symbol help, eldoc, and inferior interpreter shell, etc.

Terminal — ssh — 80×23

```
File Edit Options Buffers Tools IM-Python Python Help
#! /usr/bin/python
# primes N will print the primes <= N

from math import sqrt
from sys import argv

if len(argv) < 2:
    print "usage: primes N"
    exit()
else:
    max = int(argv[1])

def is_prime(n):
    """is_prime(n) returns True if n is a prime number"""
    for i in range(2, 1+sqrt(n)):
        if 0 == n % i:
            return False
    return True

for n in range(1,max):
----:**-F1  primes.py        (Python)--L1--Top----------------------------------
Mark set
```

Running Interactively on UNIX

On Unix…

```
% python
>>> 3+3
6
```

- Python prompts with ‘>>>’.
- To exit Python (not Idle):
  - In Unix, type CONTROL-D
  - In Windows, type CONTROL-Z + <Enter>
  - Evaluate exit()

Running Programs on UNIX

- Call python program via the python interpreter

```
% python fact.py
```

- Make a python file directly executable by
  - Adding the appropriate path to your python interpreter as the first line of your file

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

  - Making the file executable

```
% chmod a+x fact.py
```

  - Invoking file from Unix command line

```
% fact.py
```

Example ‘script’: fact.py

```
#! /usr/bin/python

def fact(x):
    """Returns the factorial of its argument, assumed to be a posint"""
    if x == 0:
        return 1
    return x * fact(x - 1)

print
print ’N fact(N)’
print "---------"

for n in range(10):
    print n, fact(n)
```

Python Scripts

- When you call a python program from the command line the interpreter evaluates each expression in the file
- Familiar mechanisms are used to provide command line arguments and/or redirect input and output
- Python also has mechanisms to allow a python program to act both as a script and as a module to be imported and used by another python program

Example of a Script

```
#! /usr/bin/python

""" reads text from standard input and outputs any email
    addresses it finds, one to a line.
"""

import re
from sys import stdin

# a regular expression ~ for a valid email address
pat = re.compile(r'[-\w][-.\w]*@[-\w][-\w.]+[a-zA-Z]{2,4}')

for line in stdin.readlines():
    for address in pat.findall(line):
        print address
```

results

```
python> python email0.py <email.txt
bill@msft.com
gates@microsoft.com
steve@apple.com
bill@msft.com
python>
```

Getting a unique, sorted list

```
import re
from sys import stdin

pat = re.compile(r'[-\w][-.\w]*@[-\w][-\w.]+[a-zA-Z]{2,4}’)
# found is an initially empty set (a list w/o duplicates)
found = set( )
for line in stdin.readlines():
    for address in pat.findall(line):
        found.add(address)
# sorted() takes a sequence, returns a sorted list of its elements
for address in sorted(found):
    print address
```

results

```
python> python email2.py <email.txt
bill@msft.com
gates@microsoft.com
steve@apple.com
python>
```

Simple functions: ex.py

```
"""factorial done recursively and iteratively"""


def fact1(n):
    ans = 1
    for i in range(2,n):
        ans = ans * n
    return ans


def fact2(n):
    if n < 1:
        return 1
    else:
        return n * fact2(n - 1)
```

Simple functions: ex.py

```
671> python

Python 2.5.2 …

>>> import ex

>>> ex.fact1(6)

1296

>>> ex.fact2(200)

788657867364790503552363213932185 07…000000L

>>> ex.fact1

<function fact1 at 0x902470>

>>> fact1

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

NameError: name 'fact1' is not defined
```