What is Python?

Python in a very high level (scripting) language which has gained widespread popularity in recent years. It is:

- Cross-platform
- Object-oriented
- Open source
Python in a very high level (scripting) language which has gained widespread popularity in recent years. It is:

- cross platform
Python in a very high level (scripting) language which has gained widespread popularity in recent years. It is:

- cross platform
- object oriented
What is Python?

Python in a very high level (scripting) language which has gained widespread popularity in recent years. It is:

- cross platform
- object oriented
- open source
Why should I care?

You may need to use a computer to run simulations, crunch data, display data... Python's 3rd-party libraries can help you with these tasks.
Why should I care?

You may need to use a computer to

- run simulations
- crunch data
- display data...
Why should I care?

You may need to use a computer to
- run simulations
- crunch data
- display data...

Python’s 3rd-party libraries can help you with these tasks.
Python is enhanced by a large set of scientific libraries that are being actively developed.
Suppose you want:
Python’s Scientific Libraries

Python is enhanced by a large set of scientific libraries that are being actively developed. Suppose you want:

- standard science and engineering functions or plotting (MATLAB)
  - SciPy, Matplotlib
- a computer algebra system (Mathematica)
  - SAGE
- data processing
  - Modular toolkit for Data Processing (MDP)
- bioinformatics functions
  - Biopython
- machine learning functions
  - PyML, mlpy, SHOGUN
- neural nets
  - Fast Artificial Neural Network (FANN) Library
- artificial intelligence or robotics routines
  - Python Robotics (Pyro)
Python’s Scientific Libraries

Python is enhanced by a large set of scientific libraries that are being actively developed. Suppose you want:

- standard science and engineering functions or plotting (MATLAB)
  - SciPy, Matplotlib
- a computer algebra system (Mathematica)
  - SAGE

Eric Christiansen  Scientific Programming in Python
Python’s Scientific Libraries

Python is enhanced by a large set of scientific libraries that are being actively developed. Suppose you want:

- standard science and engineering functions or plotting (MATLAB)
  - SciPy, Matplotlib
- a computer algebra system (Mathematica)
  - SAGE
- data processing
  - Modular toolkit for Data Processing (MDP)
Python’s Scientific Libraries

Python is enhanced by a large set of scientific libraries that are being actively developed. Suppose you want:

- standard science and engineering functions or plotting (MATLAB)
  - SciPy, Matplotlib
- a computer algebra system (Mathematica)
  - SAGE
- data processing
  - Modular toolkit for Data Processing (MDP)
- bioinformatics functions
  - Biopython
Python’s Scientific Libraries

Python is enhanced by a large set of scientific libraries that are being actively developed.

Suppose you want:

- standard science and engineering functions or plotting (MATLAB)
  - SciPy, Matplotlib
- a computer algebra system (Mathematica)
  - SAGE
- data processing
  - Modular toolkit for Data Processing (MDP)
- bioinformatics functions
  - Biopython
- machine learning functions
  - PyML, mlpy, SHOGUN
Python is enhanced by a large set of scientific libraries that are being actively developed. Suppose you want:

- standard science and engineering functions or plotting (MATLAB)
  - SciPy, Matplotlib
- a computer algebra system (Mathematica)
  - SAGE
- data processing
  - Modular toolkit for Data Processing (MDP)
- bioinformatics functions
  - Biopython
- machine learning functions
  - PyML, mlpy, SHOGUN
- neural nets
  - Fast Artificial Neural Network (FANN) Library
Python’s Scientific Libraries

Python is enhanced by a large set of scientific libraries that are being actively developed.

Suppose you want:

- standard science and engineering functions or plotting (MATLAB)
  - SciPy, Matplotlib
- a computer algebra system (Mathematica)
  - SAGE
- data processing
  - Modular toolkit for Data Processing (MDP)
- bioinformatics functions
  - Biopython
- machine learning functions
  - PyML, mlpy, SHOGUN
- neural nets
  - Fast Artificial Neural Network (FANN) Library
- artificial intelligence or robotics routines
  - Python Robotics (Pyro)
Is it hard to learn?
Is it hard to learn?

I learned it last night! Everything is so simple!

Hello world is just print "Hello, world!"

I dunno... dynamic typing? whitespace?

Come join us! programming is fun again! It's a whole new world up here!

But how are you flying?

I just typed import antigravity

That's it?

... I also sampled everything in the medicine cabinet for comparison.

But I think this is the Python.
Advantages of MATLAB:

- already widely used
- designed specifically for scientific computing
- easy to find documentation
- good IDE with debugging and profiling support "out of the box"

Advantages of Python:

- open source means no limits on use
- appears to approximately superset MATLAB's functionality
- modern language with support for object orientation
- support for calling functions in other languages
Advantages of MATLAB:
  - already widely used
Advantages of MATLAB:
- already widely used
- designed specifically for scientific computing
Advantages of MATLAB:
- already widely used
- designed specifically for scientific computing
- easy to find documentation
Advantages of MATLAB:
- already widely used
- designed specifically for scientific computing
- easy to find documentation
- good IDE with debugging and profiling support “out of the box”
Advantages of MATLAB:

- already widely used
- designed specifically for scientific computing
- easy to find documentation
- good IDE with debugging and profiling support “out of the box”

Advantages of Python:
Advantages of MATLAB:
- already widely used
- designed specifically for scientific computing
- easy to find documentation
- good IDE with debugging and profiling support “out of the box”

Advantages of Python:
- open source means no limits on use
Advantages of MATLAB:
- already widely used
- designed specifically for scientific computing
- easy to find documentation
- good IDE with debugging and profiling support “out of the box”

Advantages of Python:
- open source means no limits on use
- appears to approximately superset MATLAB’s functionality
Advantages of MATLAB:
- already widely used
- designed specifically for scientific computing
- easy to find documentation
- good IDE with debugging and profiling support “out of the box”

Advantages of Python:
- open source means no limits on use
- appears to approximately superset MATLAB’s functionality
- modern language with support for object orientation
Advantages of MATLAB:
- already widely used
- designed specifically for scientific computing
- easy to find documentation
- good IDE with debugging and profiling support “out of the box”

Advantages of Python:
- open source means no limits on use
- appears to approximately superset MATLAB’s functionality
- modern language with support for object orientation
- support for calling functions in other languages
To get information on an object from the interpreter

```
help <object>
```

Commenting:
- Inline comments are preceded with `#`
- Block comments are surrounded with `"""`

Code blocks are denoted with indentation:

```
if x == 2:
    print x
```

Python is dynamically typed:

```
a = "hello"  # a is a string
a = 4        # a is now an integer
```
Many of these functions come from SciPy.

```python
from scipy import *

Vectors:

N = 5                     # a scalar
v = [1, 2, 3]            # a list
v = array([1, 2, 3])     # a column vector
v = array([[1], [2], [3]]) # a column vector
v = array([[1, 2, 3]])    # a column vector
v = transpose(v)          # transpose a vector
                          # (row to column
                          # or column to row)

v = arange(-4, 4)         # a vector in
                          # a specified range:

v = pi*arange(-4, 4)/4    # arange(start, stop, step)

v = []                    # empty list
```
Matrices:

\[ m = \text{zeros}([2,3]) \]  \# a matrix of zeros
\[ v = \text{ones}([1,3]) \]  \# a matrix of ones
\[ v = \text{rand}(3,1) \]  \# rand matrix (see also randn)
\[ m = \text{eye}(3) \]  \# identity matrix (3x3)
```python
v = array([5, 6, 7])  # access a vector element
# vector(index) – arrays are zero-indexed
v[2]  # vector(index)

len(v)  # number of elements in
# a vector

m = array([[1, 2, 3],
            [4, 5, 6]])  # a 2x3 matrix
m[1,2] = m[1][2]  # access a matrix element
# matrix[row, column]
```
m[1,:]
  # access a matrix row
  # (bottom row)

m[:,0]
  # access a matrix column
  # (left column)

m.reshape([4,1])
  # turn matrix into a column
  # vector (concatenate rows)

m.shape
  # size of a matrix [rows, cols]

m.shape[0]
  # number of rows

m.shape[1]
  # number of columns

zeros(m.shape)
  # create a new matrix with
  # size of m

m = array([[ 'hello', sum],
            [1,2]])
  # put whatever you want
  # into an array

m[0,1](m[1])
  # call sum on bottom row of m
Arithmetic operations performed on arrays are done “element by element”.

```
a = array([1, 2, 3, 4])  # vector
2 * a                   # scalar multiplication
a / 4                   # scalar division
b = array([5, 6, 7, 8])  # vector
a + b                   # pointwise vector addition
a - b                   # pointwise vector subtraction
a ** 2                  # pointwise vector squaring
a * b                   # pointwise vector multiply
a / b                   # pointwise vector divide
log(a)                  # pointwise logarithm
around(array([[.6],
               [.5]]))  # pointwise rounding
# (.5 rounds to 0)
```
Vector Operations

\[ a = \text{array}([1, 4, 6, 3]) \] # vector
sum(a) # sum of vector elements
mean(a) # mean of vector elements
var(a) # variance
std(a) # standard deviation
max(a) # maximum

\[ a = \text{array}([[1, 2, 3], [4, 5, 6]]) \] # matrix
mean(a, 0) # mean of each column
amax(a, 1) # max of each row
amax(a) # to obtain max of matrix
# note we use 2
# different max functions
Matrix Operations

```python
dot(transpose(array([1, 2, 3])),
    array([4, 5, 6]))  # row vector 1x3 times column
    # vector 3x1 results in a
    # single number, also known
    # as dot/inner product

dot(array([[1], [2], [3]]),
    array([[4, 5, 6]]))  # column vector 3x1 times row
    # vector 1x3 results in 3x3
    # matrix, also known
    # as outer product

a = rand(3, 2)  # 3x2 matrix
b = rand(2, 4)  # 2x4 matrix
dot(a, b)  # 3x4 matrix
```
import cPickle as pickle  # this module lets you
    # save and reload objects

f = open('save_file', 'w')  # open archive file
pickle.dump(obj, f)  # dump object to archive
f.close()  # close archive file

def obj  # clear object
    # from memory

f = open('save_file', 'r')  # open archive file
obj = pickle.load(f)  # read object
    # from archive
f.close()  # close archive file
Example: given a list \( v \), create a new list \( u \) with values equal to \( v \) if they are greater than 0, and equal to 0 if they less than or equal to 0.

Using a for loop:

\[
v = [3, 5, -2, 5, -1, 0]
\]
\[
u = [0] \ast \text{len}(v) \neq all \ zeros
\]
\[
\text{for } i \text{ in range(len}(v)): \quad \text{if } v(i) > 0: \quad u(i) = v(i)
\]

Using list comprehension:

\[
v = [3, 5, -2, 5, -1, 0]
\]
\[
u = [\text{max}(e, 0) \text{ for } e \text{ in } v]
\]
Save the following code to “mylib.py”:

```python
def myfunc(a, b):
    return a+b**2
```

Import and use myfunc. Note, we might need to configure PYTHONPATH.

```python
from mylib import myfunc
myfunc(1,2)
myfunc(b=2,a=1)  # same as above
```

Python also supports class creation:

```python
class MyClass:
    def __init__(self):
        print "hello!"
my_class = MyClass()
```
The library matplotlib / pylab is your friend:

```python
from pylab import *
xs = arange(-2, 2, .01)
plot(xs, sin(xs))
show()
```
We use the Python Imaging Library as well as matplotlib / pylab

```python
from pylab import *
import Image

im = Image.open( 'my_image.jpg' )
im.show()  # we can display the image
ima = array(im)  # typecasting to array
               # extracts pixel values
imr = Image.fromstring( 'RGB',
     (ima.shape[1],ima.shape[0]),
     ima.tostring()  # convert array into image
)
img = ima.mean(2)  # average color intensities
                   # for each pixel

imshow(img)
autumn()  # set default colormap to autumn
show()
```

Eric Christiansen  Scientific Programming in Python
How do I start?

Many guides and tutorials are available online:
How do I start?

Many guides and tutorials are available online:
Dive Into Python

- python introduction for programmers
How do I start?

Many guides and tutorials are available online:
- Dive Into Python
  - python introduction for programmers
- A list of tutorials for Python and some of its many libraries can be found at http://www.awaretek.com/tutorials.html
Questions?