Introduction to Python

Luis Pedro Coelho
Carnegie Mellon University

Lisbon Machine Learning School
Python was started in the late 80’s.

It was intended to be both *easy to teach* and *industrial strength*.

It is (has always been) open-source.

It has become one of the most widely used languages (top 10).
There are two major versions, currently: 2.7 and 3.2.

We are going to be using 2.7 (but 2.6 should be OK too).
print "Hello World"
Task

Average

Compute the average of the following numbers:

1. 10
2. 7
3. 22
4. 14
5. 17
numbers = [10, 7, 22, 14, 17]

sum = 0
n = 0
for val in numbers:
    sum = sum + val
    n = n + 1
return sum / n
“Python is executable pseudo-code.”
—Python lore (often attributed to Bruce Eckel)
numbers = [10, 7, 22, 14, 17]

sum = 0
n = 0

for val in numbers:
    sum = sum + val
    n = n + 1

return sum / n
Python Types

Basic Types

- Numbers (integers and floating point)
- Strings
- Lists and tuples
- Dictionaries
A = 1
B = 2
C = 3

```
print A+B*C
```

Outputs 7.
Python Types: Numbers II: Floats

A = 1.2
B = 2.4
C = 3.6

```
print A + B*C
```

Outputs 9.84.
A = 2
B = 2.5
C = 4.4

```
print A + B*C
```

Outputs 22.0.
total = total + n

Can be abbreviated as

total += n
first = 'John'
last = "Doe"
full = first + " " + last

print full
first = 'John'
last = "Doe"
full = first + " " + last

print full

Outputs John Doe.
What is a String Literal

- Short string literals are delimited by ("”) or (’’).
- Short string literals are one line only.
- Special characters are input using escape sequences. (\n for newline,…)

multiple = 'He: May I?\nShe: No, you may not.'
alternative = "He: May I?\nShe: No, you may not."
We can input a long string using triple quotes (""" or """") as delimiters.

```python
long = """Tell me, is love
Still a popular suggestion
Or merely an obsolete art?

Forgive me, for asking,
This simple question,
I am unfamiliar with his heart."""
```
Python Types: Lists

courses = ['PfS', 'Political Philosophy']

print "The first course is", courses[0]
print "The second course is", courses[1]

Notice that list indices start at 0!
mixed = ['Banana', 100, ['Another', 'List'], []]
print len(mixed)
Python Types: Lists

```python
fruits = ['Banana', 'Apple', 'Orange']
fruits.sort()
print(fruits)
```

Prints ['Apple', 'Banana', 'Orange']
emails = { 'Luis': 'lpc@cmu.edu',
          'Mark': 'mark@cmu.edu' }

print "Luis's email is", emails['Luis']

emails['Rita'] = 'rita@cmu.edu'
student = 'Rita'
average = gradeavg(student)
if average > 0.7:
    print student, 'passed!'
    print 'Congratulations!!'
else:
    print student, 'failed. Sorry.'
Unlike almost all other modern programming languages, Python uses indentation to delimit blocks!

```
if <condition>:
    statement 1
    statement 2
    statement 3
next statement
```
Convention

1. Use 4 spaces to indent.
2. Other things will work, but confuse people.
Conditionals

Examples

- `x == y`
- `x != y`
- `x < y`
- `x < y < z`
- `x in lst`
- `x not in lst`
Nested Blocks

```python
if <condition 1>:
    do something
    if condition 2>:
        nested block
    else:
        nested else block
elif <condition 1b>:
    do something
```
For loop

```python
students = ['Luis', 'Rita', 'Sabah', 'Mark']
for st in students:
    print(st)
```
While Loop

```python
while <condition>:
    statement1
    statement2
```
Other Loopy Stuff

```python
for i in range(5):
    print i
```

prints

0
1
2
3
4

This is because `range(5)` is the list `[0, 1, 2, 3, 4]`. 
rita_enrolled = False
for st in students:
    if st == 'Rita':
        rita_enrolled = True
        break
Booleans

- Just two values: *True* and *False*.
- Comparisons return booleans (e.g., $x < 2$)

Conditions

- When evaluating a condition, the condition is converted to a boolean:
- Many things are converted to *False*:
  1. `[]` (the empty list)
  2. `{}` (the empty dictionary)
  3. `'` (the empty string)
  4. `0` or `0.0` (the value zero)
  5. ...
- Everything else is *True* or not convertible to boolean.
A = []
B = [1, 2]
C = 2
D = 0

if A:
    print 'A is true'
if B:
    print 'B is true'
if C:
    print 'C is true'
if D:
    print 'D is true'
Numbers

Two Types of Numbers

1. Integers
2. Floating-point

Operations

1. Unary Minus: \(-x\)
2. Addition: \(x + y\)
3. Subtraction: \(x - y\)
4. Multiplication: \(x \times y\)
5. Exponentiation: \(x ^ {y}\)
Division

What is 9 divided by 3?
What is 10 divided by 3?
Division

What is 9 divided by 3?
What is 10 divided by 3?

Two types of division

1. Integer division: $x \div y$
def double(x):
    '''
    y = double(x)

    Returns the double of x
    '''

    return 2*x
A = 4

```
print double(A)
print double(2.3)
print double(double(A))
```
Numpy
numpy.array or numpy.ndarray.

Multi-dimensional array of numbers.
```python
import numpy as np
A = np.array([[0,1,2],
              [2,3,4],
              [4,5,6],
              [6,7,8]])
print A[0,0]
print A[0,1]
print A[1,0]
```
```python
import numpy as np
A = np.array([
    [0, 1, 2],
    [2, 3, 4],
    [4, 5, 6],
    [6, 7, 8]])

print A[0,0]
print A[0,1]
print A[1,0]
```

```
0
1
2
```
Why Numpy?

Why do we need numpy?

```python
import numpy as np
lst = [0.,1.,2.,3.]
arr = np.array([0.,1.,2.,3.])
```
A Python List of Numbers

- float 0.0
- float 1.0
- float 2.0
- float 3.0
# A Numpy Array of Numbers

<table>
<thead>
<tr>
<th>Type</th>
<th>0.0</th>
<th>1.0</th>
<th>2.0</th>
<th>3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Numpy Arrays

Advantages

- Less memory consumption
- Faster
- Work with (or write) code in other languages (C, C++, Fortran, ...)

Luis Pedro Coelho (Carnegie Mellon University)  ⋆  Introduction to Python  ⋆  Lisbon Machine Learning School (42 / 77)
Some Array Properties

```python
import numpy as np
A = np.array([[0,1,2],
              [2,3,4],
              [4,5,6],
              [6,7,8]])
print(A.shape)
print(A.size)
```
Some Array Functions

... 

```
print A.max()
print A.min()
```

- `max()`: maximum
- `min()`: minimum
- `ptp()`: spread (max - min)
- `sum()`: sum
- `std()`: standard deviation
- ...
Other Functions

- np.exp
- np.sin
- ...

All of these work element-wise!
import numpy as np
A = np.array([0,1,2,3])
B = np.array([1,1,2,2])

print A + B
print A * B
print A / B
Arithmetic Operations

```python
import numpy as np
A = np.array([0,1,2,3])
B = np.array([1,1,2,2])

print A + B
print A * B
print A / B
```

Prints

```
array([1, 2, 4, 5])
```
import numpy as np
A = np.arange(100)
print A + 2
A += 2
numpy.ndarray is a homogeneous array of numbers.

Types

- Boolean
- int8, int16, …
- uint8, uint16, …
- float32, float64, …
- …
import numpy as np
A = np.array([0,1,1], np.float32)
A = np.array([0,1,1], float)
A = np.array([0,1,1], bool)
Reduction

A = np.array(
    [[0,0,1],
     [1,2,3],
     [2,4,2],
     [1,0,1]])

print A.max(0)
print A.max(1)
print A.max()

prints

[2,4,3]
[1,3,4,1]
4

The same is true for many other functions.
import numpy as np
A = np.array([[0, 1, 2],
              [2, 3, 4],
              [4, 5, 6],
              [6, 7, 8]])
print A[0]
print A[0].shape
print A[1]
print A[:, 2]
import numpy as np
A = np.array([[0, 1, 2],
              [2, 3, 4],
              [4, 5, 6],
              [6, 7, 8]])

print A[0]
print A[0].shape
print A[1]
print A[:, 2]

[0, 1, 2]
(3,)
[2, 3, 4]
[2, 4, 6, 8]
import numpy as np
A = np.array([[0,1,2],[2,3,4],[4,5,6],[6,7,8]])
B = A[0]
B[0] = -1
print A[0,0]
def double(A):
    A *= 2

A = np.arange(20)
double(A)
Pass is By Reference

```python
def double(A):
    A *= 2

A = np.arange(20)
double(A)

A = np.arange(20)
B = A.copy()
```
Logical Arrays

A = np.array([-1, 0, 1, 2, -2, 3, 4, -2])
print (A > 0)
A = np.array([-1,0,1,2,-2,3,4,-2])
print ((A > 0) & (A < 3)).mean()

What does this do?
Logical Indexing

\[ A[A < 0] = 0 \]

or

\[ A *= (A > 0) \]
print 'Mean of positives', A[A > 0].mean()
Some Helper Functions

Constructing Arrays

A = np.zeros((10,10), dtype=np.int8)
B = np.ones(10)
C = np.arange(100).reshape((10,10))
...

Multiple Dimensions

img = np.zeros((1024,1024,3), dtype=np.uint8)
http://docs.scipy.org/doc/
Matplotlib & Examples
Matplotlib

- Matplotlib is a plotting library.
- Very flexible.
- Very active project.
import numpy as np
import matplotlib.pyplot as plt
X = np.linspace(-4, 4, 1000)
plt.plot(X, X**2*np.cos(X**2))
plt.savefig('simple.pdf')

\[ y = x^2 \cos\left(x^2\right) \]
import numpy as np
import scipy.stats.distributions as dists
import matplotlib.pyplot as plt

r = dists.laplace()  # Laplacian with default parameters
S = r.rvs(10000)  # get 10k random variates

# 1000 values from -10 to +10
x = np.linspace(-10, 10, 1000)
plt.hist(S, 1000, normed=True)
plt.plot(x, r.pdf(x))
plt.savefig('laplace_10k.pdf')

S = r.rvs(100000)
plt.hist(S, 1000, normed=True)
plt.savefig('laplace_100k.pdf')
Sample Laplacian
Sample Laplacian
Principal Component Analysis

```
sigma = [[8, 6], [6, 8]]
points = np.array([np.random.multivariate_normal([1, 1], sigma)
                   for i in xrange(1000)])
plt.plot(points[:, 0], points[:, 1], 'rx')
plt.axis('equal')
...
```

Samples from $\mathcal{N}( [1, 1], \Sigma^{-1})$ with

$$
\Sigma = \begin{pmatrix}
8 & 6 \\
6 & 8
\end{pmatrix}
$$
mu = points.mean(0)
_, V = pca(points-mu, False)
plt.plot([mu[0], mu[0] + V[0,1]*4], [mu[1], mu[1]+V[1,1]*4], 'g->', lw=4)
plt.plot([mu[0], mu[0] + V[0,0]*4], [mu[1], mu[1]+V[1,0]*4], 'g->', lw=4)
Gaussian Mixture

```python
def point():
    s = np.random.random() < .4
    if s:
        return np.random.multivariate_normal([1,1], sigma)
    else:
        return np.random.multivariate_normal([5,5], sigma)

points = np.array([point() for i in xrange(1000)])
plt.plot(points[:,0], points[:,1], 'rx')
plt.savefig('mixture.pdf')
...
```
assignments, centroids = kmeans(points, 2, R=0)
plt.plot(points[assignments == 0,0],
point[assignments == 0,1], 'rx')
plt.plot(points[assignments == 1,0],
point[assignments == 1,1], 'bo')
plt.plot(centroids[0,0], centroids[0,1], 'g<')
plt.plot(centroids[1,0], centroids[1,1], 'g<')
plt.savefig('kmeans.pdf')
Resources

- **Numpy+scipy docs**: [http://docs.scipy.org](http://docs.scipy.org)
- **Matplotlib**: [http://matplotlib.sf.net](http://matplotlib.sf.net)
- **Python docs**: [http://docs.python.org](http://docs.python.org)

- These slides are available at [http://luispedro.org/talks/2011](http://luispedro.org/talks/2011)
- I’m available at lpc@cmu.edu
Thank you.