Introduction to Python Programming

Luis Pedro Coelho
§
On the web: http://luispedro.org
On twitter: @luispedrocoelho

European Molecular Biology Laboratory

November 14, 2014
Let’s digress for a moment discussing the language...
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.
In the last 10 years, it has become one of the most widely used languages (top 10).
The current versions of Python are 2.7 and 3.4
This class assumes you have 2.6–2.7
There are some small differences when compared to version 3.x
What is a Computer?

1. Memory
2. Processor
3. Magic
1. Objects
2. Operations on objects
3. Magic
Python Example

print "Hello World"
Running Python

1. From a file
2. Interactively
```
print 'Hello World'
```
Running a Program

1. Shell
2. IDE
Let me show you a demonstration...
More Complex Example

What is 25 times 5?
What is 25 times 5?

```
print 25 * 5
```
name = 2
other = 3
yetanother = name + other
name = 5
print yetanother + name
Conditionals

```python
if <condition>:
    <statement 1>
    <statement 2>
else:
    <statement 3>
```
print 'Before testing. . .' 
if 3.3*9.2 > 30:
    print 'Greater than 30'
else:
    print 'Smaller or equal'
print 'After'
print 'Before testing. . .'
if 3.3*9.2 > 31:
    print 'Greater than 31'
elif 3.3*9.2 > 30:
    print 'Greater than 30'
else:
    print 'Smaller or equal'
print 'After'
print 'Before testing. . .'
v = 3.3*9.2
if v > 31:
    print 'Greater than 31'
elif v > 30:
    print 'Greater than 30'
else:
    print 'Smaller or equal'
print 'After'
Lists

```python
students = [ 'Luis', 'Mark', 'Rita' ]

print students[0]
print students[1]
print students[2]
```
Loops

students = [ 'Luis', 'Mark', 'Rita', ... ]

for st in students:
    print st
Example

```
values = [0.11, -0.23, -0.16, 0.18, 0.23, 0.19]
```
Example

```python
values = [0.11, -0.23, -0.16, 0.18, 0.23, 0.19]

sum = 0
for v in values:
    sum = sum + v

print sum
```
Exercise

- How do you obtain the number of elements in a list?
- Use this to compute the mean of a list of numbers
Example

\[
\text{values} = [0.11, -0.23, -0.16, 0.18, 0.23, 0.19]
\]

\[
\text{sum} = 0.0 \\
\text{sum2} = 0.0 \\
\text{for } v \text{ in } \text{values}: \\
\quad \text{sum} = \text{sum} + v \\
\quad \text{sum2} = \text{sum2} + v \ast v
\]

\[
\text{mu} = \text{sum}/\text{len(values)} \\
\text{mu2} = \text{sum2}/\text{len(values)} \\
\text{print 'Average: } \{0\} \text{'.format(mu)} \\
\text{print 'Std Dev: } \{0\} \text{'.format(mu2 - mu*mu)}
\]
Example

\[
\text{values} = [0.11, -0.23, -0.16, 0.18, 0.23, 0.19]
\]

\[
\text{sum} = 0.0
\]

\[
\text{sum2} = 0.0
\]

\[
\text{for } v \text{ in values:}
\]

\[
\quad \text{sum } += v
\]

\[
\quad \text{sum2 } += v \times v
\]

\[
\mu = \frac{\text{sum}}{\text{len(values)}}
\]

\[
\mu2 = \frac{\text{sum2}}{\text{len(values)}}
\]

\[
\text{print } '\text{Average: }{0}' . \text{format} (\mu)
\]

\[
\text{print } '\text{Std Dev: }{0}' . \text{format} (\mu2 - \mu \times \mu)
\]
Example

$$\text{values} = [0.11, \ -0.23, \ -0.16, \ 0.18, \ 0.23, \ 0.19]$$

def example():
    mu = 0.0
    mu2 = 0.0
    for v in values:
        mu += v
        mu2 += v * v
        mu /= len(values)
        mu2 /= len(values)
    print 'Average: {0}'.format(mu)
    print 'Std Dev: {0}'.format(mu2 - mu*mu)
Example

```python
values = [0.11, -0.23, -0.16, 0.18, 0.23, 0.19]

mu = 0.0
mu2 = 0.0
for v in values:
    mu += v
    mu2 += v * v

mu /= len(values)
mu2 /= len(values)
print 'Average: {0}'.format(mu)
print 'Std Dev: {0}'.format(mu2 - mu*mu)
```
Adapt the code to ignore negative numbers.
Exercise

Adapt the code to ignore negative numbers.

```
values = [0.11, -0.23, -0.16, 0.18, 0.23, 0.19]

mu = 0.0
mu2 = 0.0
n = 0.0
for v in values:
    if v >= 0.0:
        mu += v
        mu2 += v * v
    n += 1

mu /= n
mu2 /= n
print 'Average: {0}'.format(mu)
print 'Std Dev:  {0}'.format(mu2 - mu*mu)
```
Greatest Common Divisor (Euclid’s Method)

\[
gcd(a, b) = \begin{cases} 
    a & \text{if } b = a \\
    \gcd(a - b, b) & \text{if } a > b \\
    \gcd(a, b - a) & \text{o.w.}
\end{cases}
\]
Greatest Common Divisor (Euclid’s Method)

\[
gcd(a, b) = \begin{cases} 
a & \text{if } b = a \\
gcd(a - b, b) & \text{if } a > b \\
gcd(a, b - a) & \text{o.w.}
\end{cases}
\]

\[a = 9344\]
\[b = 6497\]

```python
while a != b:
    if a > b:
        a, b = a - b, b
    else:
        a, b = a, b - a
print a
```
Python So Far

Python

1. Basic types: int, float, list
2. Control flow: for, while, if, else, elif
```python
students = [ 'Luis' , 'Rita' , 'Sabah' , 'Grace' ]
print students[0]
print students[1:2]
print students[1:]
print students[-1]
print students[-2]
```
A = (0, 1, 2)
B = (1, )

print A[0]
print len(B)
Tuples are like immutable lists.
Dictionaries

- Dictionaries are associative arrays.

```python
gene2ensembl = {}
gene2ensembl[ 'SMAD9' ] = 'ENSG00000120693'
gene2ensembl[ 'ZNF670' ] = 'ENSG00000135747'

print gene2ensembl[ 'SMAD9' ]
```
```
gene2expression = {
    'SMAD9' : 12.3,
    'ZNF670' : 4.3,
}

print len(gene2ensembl)
print gene2ensembl.keys()
```