Python Basics

Copyright © Software Carpentry 2010
This work is licensed under the Creative Commons Attribution License
See http://software-carpentry.org/license.html for more information.
A simple interpreted language
A simple interpreted language
no separate compilation step
A simple interpreted language
no separate compilation step

$ python

>>>
A simple interpreted language
no separate compilation step

$ python
>>> print 1 + 2
3
>>>
A simple interpreted language

no separate compilation step

$ python
>>> print 1 + 2
3
>>> print 'charles' + 'darwin'
charlesdarwin
>>>
Put commands in a file and execute that
Put commands in a file and execute that

```
$ nano very-simple.py
```
Put commands in a file and execute that

```
$ nano very-simple.py

print 1 + 2
print 'charles' + 'darwin'
```
Put commands in a file and execute that

```bash
$ nano very-simple.py

print 1 + 2
print 'charles' + 'darwin'

$ python very-simple.py
3
charlesdarwin
$
```
Use an *integrated development environment* (IDE)
Use an *integrated development environment* (IDE)
Use an *integrated development environment* (IDE)

- **Source file**
- **Execution shell**
Variables are names for values
Variables are names for values
Created by use
Variables are names for values
Created by use: no declaration necessary
Variables are names for values
Created by use: no declaration necessary

```python
>>> planet = 'Pluto'
```
Variables are names for values
Created by use: no declaration necessary

```python
>>> planet = 'Pluto'
>>> print planet
Pluto
```
Variables are names for values

Created by use: no declaration necessary

```python
>>> planet = 'Pluto'
>>> print planet
Pluto
```
Variables are names for values
Created by use: no declaration necessary

```python
>>> planet = 'Pluto'
>>> print planet
Pluto
>>> moon = 'Charon'
>>> print moon
Charon
```
Variables are names for values
Created by use: no declaration necessary

```python
>>> planet = 'Pluto'
>>> print planet
Pluto
>>> moon = 'Charon'
>>> p = planet
>>>```

<table>
<thead>
<tr>
<th>variable</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>planet</td>
<td>'Pluto'</td>
</tr>
<tr>
<td>moon</td>
<td>'Charon'</td>
</tr>
</tbody>
</table>
Variables are names for values
Created by use: no declaration necessary

```python
>>> planet = 'Pluto'
>>> print planet
Pluto
>>> moon = 'Charon'
>>> p = planet
>>> 
```

<table>
<thead>
<tr>
<th>variable</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>planet</td>
<td>'Pluto'</td>
</tr>
<tr>
<td>moon</td>
<td>'Charon'</td>
</tr>
<tr>
<td>p</td>
<td></td>
</tr>
</tbody>
</table>
Variables are names for values
Created by use: no declaration necessary

```python
>>> planet = 'Pluto'
>>> print planet
Pluto
>>> moon = 'Charon'
>>> p = planet
>>> print p
Pluto
```
A variable is just a name
A variable is just a name
Does not have a type
A variable is just a name
Does not have a type

```python
>>> planet = 'Pluto'
```
A variable is just a name
Does not have a type

```python
>>> planet = 'Pluto'
>>> planet  # 'Pluto'
```
A variable is just a name
Does not have a type

```python
>>> planet = 'Pluto'
>>> planet = 9
```

<table>
<thead>
<tr>
<th>variable</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>planet</td>
<td>'Pluto'</td>
</tr>
<tr>
<td>9</td>
<td>integer</td>
</tr>
</tbody>
</table>
A variable is just a name

Does not have a type

```python
>>> planet = 'Pluto'
>>> planet = 9
>>> planet  # 'Pluto' 9
```

Values are *garbage collected*
A variable is just a name
Does not have a type

```python
>>> planet = 'Pluto'
>>> planet = 9
>>> planet
'Pluto' 9
```

Values are *garbage collected*

If nothing refers to data any longer, it can be recycled
A variable is just a name
Does not have a type

```python
>>> planet = 'Pluto'
>>> planet = 9
```

Values are *garbage collected*
If nothing refers to data any longer, it can be recycled
Must assign value to variable before using it
Must assign value to variable before using it

```python
>>> planet = 'Sedna'
```
Must assign value to variable before using it

```python
>>> planet = 'Sedna'
>>> print plant  # note the deliberate misspelling
```
Must assign value to variable before using it

```python
>>> planet = 'Sedna'
>>> print plant       # note the deliberate misspelling
Traceback (most recent call last):
  print plant
NameError: name 'plant' is not defined
```
Must assign value to variable before using it

```python
>>> planet = 'Sedna'
>>> print plant               # note the deliberate misspelling
Traceback (most recent call last):
  print plant
NameError: name 'plant' is not defined
```
Must assign value to variable before using it

```python
>>> planet = 'Sedna'
>>> print plant               # note the deliberate misspelling
Traceback (most recent call last):
  print plant
NameError: name 'plant' is not defined
```

Python does not assume default values for variables

Doing so can mask many errors
Must assign value to variable before using it

```python
>>> planet = 'Sedna'
>>> print plant              # note the deliberate misspelling
Traceback (most recent call last):
    print plant
NameError: name 'plant' is not defined
>>> 
```

Python does not assume default values for variables
Doing so can mask many errors
Anything from # to the end of the line is a comment
Values *do* have types
Values *do* have types

```python
>>> string = "two"
>>> number = 3
>>> print string * number  # repeated concatenation
twotwotwo
```
Values *do* have types

```python
>>> string = "two"
>>> number = 3
>>> print string * number  # repeated concatenation
  twotwotwo
>>> print string + number
  Traceback (most recent call last)
    number + string
  TypeError: cannot concatenate 'str' and 'int' objects
>>> ```
Values *do* have types

```python
>>> string = "two"
>>> number = 3
>>> print string * number    # repeated concatenation
twotwotwo
>>> print string + number
Traceback (most recent call last)
  number + string
TypeError: cannot concatenate 'str' and 'int' objects

Would probably be safe here to produce 'two3'
```
Values *do* have types

```python
>>> string = "two"
>>> number = 3
>>> print string * number  # repeated concatenation
twotwotwo
>>> print string + number
Traceback (most recent call last)
  number + string
TypeError: cannot concatenate 'str' and 'int' objects
>>> 
Would probably be safe here to produce 'two3'

But then what should '2'+'3' be?
Values do have types

>>> string = "two"
>>> number = 3
>>> print string * number  # repeated concatenation
twotwotwo
>>> print string + number
Traceback (most recent call last)
  number + string
TypeError: cannot concatenate 'str' and 'int' objects

Would probably be safe here to produce 'two3'

But then what should '2'+'3' be?

Doing too much is as bad as doing too little...
Use functions to convert between types
Use functions to convert between types

```python
>>> print int('2') + 3
5
>>>```

Use functions to convert between types

```python
>>> print int('2') + 3
5
>>> print 2 + str(3)
23
>>> 
```
Numbers
<table>
<thead>
<tr>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
# Numbers

<table>
<thead>
<tr>
<th>14</th>
<th>32-bit integer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(on most machines)</td>
</tr>
<tr>
<td>14.0</td>
<td>64-bit float</td>
</tr>
<tr>
<td></td>
<td>(ditto)</td>
</tr>
</tbody>
</table>
### Numbers

<table>
<thead>
<tr>
<th>14</th>
<th>32-bit integer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(on most machines)</td>
</tr>
<tr>
<td>14.0</td>
<td>64-bit float</td>
</tr>
<tr>
<td></td>
<td>(ditto)</td>
</tr>
<tr>
<td>1+4j</td>
<td>complex number</td>
</tr>
<tr>
<td></td>
<td>(two 64-bit floats)</td>
</tr>
</tbody>
</table>
## Numbers

<table>
<thead>
<tr>
<th></th>
<th>32-bit integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>(on most machines)</td>
</tr>
<tr>
<td>14.0</td>
<td>64-bit float</td>
</tr>
<tr>
<td>1+4j</td>
<td>complex number</td>
</tr>
<tr>
<td>x.real, x.imag</td>
<td>real and imaginary parts of complex number</td>
</tr>
</tbody>
</table>
Arithmetic
| Arithmetic | Addition | 35 + 22 | 57 |
## Arithmetic

<table>
<thead>
<tr>
<th>Addition</th>
<th>+</th>
<th>35 + 22</th>
<th>57</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>'Py' + 'thon'</td>
<td>'Python'</td>
</tr>
</tbody>
</table>
## Arithmetic

<table>
<thead>
<tr>
<th>Operation</th>
<th>Symbol</th>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>+</td>
<td>35 + 22</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Py' + 'thon'</td>
<td>'Python'</td>
</tr>
<tr>
<td>Subtraction</td>
<td>-</td>
<td>35 - 22</td>
<td>13</td>
</tr>
</tbody>
</table>
## Arithmetic

<table>
<thead>
<tr>
<th>Operation</th>
<th>Symbol</th>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>+</td>
<td>35 + 22</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Py' + 'thon'</td>
<td>'Python'</td>
</tr>
<tr>
<td>Subtraction</td>
<td>-</td>
<td>35 - 22</td>
<td>13</td>
</tr>
<tr>
<td>Multiplication</td>
<td>*</td>
<td>3 * 2</td>
<td>6</td>
</tr>
</tbody>
</table>
## Arithmetic

<table>
<thead>
<tr>
<th>Operation</th>
<th>Operator</th>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>+</td>
<td>$35 + 22$</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>'Py' + 'thon'</code></td>
<td><code>'Python'</code></td>
</tr>
<tr>
<td>Subtraction</td>
<td>-</td>
<td>$35 - 22$</td>
<td>13</td>
</tr>
<tr>
<td>Multiplication</td>
<td>*</td>
<td>$3 \times 2$</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>'Py' \times 2</code></td>
<td><code>'PyPy'</code></td>
</tr>
</tbody>
</table>
## Arithmetic

<table>
<thead>
<tr>
<th>Operation</th>
<th>Operator</th>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>+</td>
<td>35 + 22</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Py' + 'thon'</td>
<td>'Python'</td>
</tr>
<tr>
<td>Subtraction</td>
<td>-</td>
<td>35 - 22</td>
<td>13</td>
</tr>
<tr>
<td>Multiplication</td>
<td>*</td>
<td>3 * 2</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Py' * 2</td>
<td>'PyPy'</td>
</tr>
<tr>
<td>Division</td>
<td>/</td>
<td>3.0 / 2</td>
<td>1.5</td>
</tr>
</tbody>
</table>
## Arithmetic

<table>
<thead>
<tr>
<th>Operation</th>
<th>Operator</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>+</td>
<td>35 + 22</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Py’ + 'thon’</td>
<td>'Python’</td>
</tr>
<tr>
<td>Subtraction</td>
<td>-</td>
<td>35 - 22</td>
<td>13</td>
</tr>
<tr>
<td>Multiplication</td>
<td>*</td>
<td>3 * 2</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Py’ * 2</td>
<td>'PyPy’</td>
</tr>
<tr>
<td>Division</td>
<td>/</td>
<td>3.0 / 2</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 / 2</td>
<td>1</td>
</tr>
</tbody>
</table>
## Arithmetic

<table>
<thead>
<tr>
<th>Operation</th>
<th>Operator</th>
<th>Expression 1</th>
<th>Expression 2</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>+</td>
<td>35 + 22</td>
<td></td>
<td>57</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Py' + 'thon'</td>
<td>'Python'</td>
<td></td>
</tr>
<tr>
<td>Subtraction</td>
<td>-</td>
<td>35 - 22</td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>Multiplication</td>
<td>*</td>
<td>3 * 2</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Py' * 2</td>
<td>'PyPy'</td>
<td></td>
</tr>
<tr>
<td>Division</td>
<td>/</td>
<td>3.0 / 2</td>
<td></td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 / 2</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Exponentiation</td>
<td>**</td>
<td>2 ** 0.5</td>
<td></td>
<td>1.41421356...</td>
</tr>
</tbody>
</table>
## Arithmetic

<table>
<thead>
<tr>
<th>Operation</th>
<th>Symbol</th>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>+</td>
<td>35 + 22</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Py' + 'thon'</td>
<td>'Python'</td>
</tr>
<tr>
<td>Subtraction</td>
<td>-</td>
<td>35 - 22</td>
<td>13</td>
</tr>
<tr>
<td>Multiplication</td>
<td>*</td>
<td>3 * 2</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Py' * 2</td>
<td>'PyPy'</td>
</tr>
<tr>
<td>Division</td>
<td>/</td>
<td>3.0 / 2</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 / 2</td>
<td>1</td>
</tr>
<tr>
<td>Exponentiation</td>
<td>**</td>
<td>2 ** 0.5</td>
<td>1.41421356...</td>
</tr>
<tr>
<td>Remainder</td>
<td>%</td>
<td>13 % 5</td>
<td>3</td>
</tr>
</tbody>
</table>
Prefer *in-place* forms of binary operators
Prefer *in-place* forms of binary operators

```python
>>> years = 500
```
Prefer *in-place* forms of binary operators

```python
>>> years = 500
>>> years += 1
```
Prefer *in-place* forms of binary operators

```python
>>> years = 500
>>> years += 1  # The same as years = years + 1
>>> `
Prefer *in-place* forms of binary operators

```python
>>> years = 500
>>> years += 1
>>> print years
501
>>> 
```
Prefer *in-place* forms of binary operators

```python
>>> years = 500
>>> years += 1
>>> print years
501
>>> years %= 10
>>>```

Prefer *in-place* forms of binary operators

```python
>>> years = 500
>>> years += 1
>>> print years
501
>>> years %= 10 # The same as years = years % 10
>>> 
```
Prefer *in-place* forms of binary operators

```python
>>> years = 500
>>> years += 1
>>> print years
501
>>> years %= 10
>>> print years
5
>>> 
```
Comparisons
## Comparisons

| 3 < 5 | True |
## Comparisons

<table>
<thead>
<tr>
<th>3 &lt; 5</th>
<th>True</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 != 5</td>
<td>True</td>
</tr>
</tbody>
</table>
## Comparisons

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 &lt; 5</td>
<td>True</td>
</tr>
<tr>
<td>3 != 5</td>
<td>True</td>
</tr>
<tr>
<td>3 == 5</td>
<td>False</td>
</tr>
</tbody>
</table>
## Comparisons

<table>
<thead>
<tr>
<th>Condition</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 &lt; 5</td>
<td>True</td>
</tr>
<tr>
<td>3 != 5</td>
<td>True</td>
</tr>
<tr>
<td>3 == 5</td>
<td>False</td>
</tr>
</tbody>
</table>

*Single = is assignment*  
*Double == is equality*
## Comparisons

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3 &lt; 5</td>
<td>True</td>
</tr>
<tr>
<td>3 != 5</td>
<td>True</td>
</tr>
<tr>
<td>3 == 5</td>
<td>False</td>
</tr>
<tr>
<td>3 &gt;= 5</td>
<td>False</td>
</tr>
</tbody>
</table>
Comparisons

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 &lt; 5</td>
<td>True</td>
</tr>
<tr>
<td>3 != 5</td>
<td>True</td>
</tr>
<tr>
<td>3 == 5</td>
<td>False</td>
</tr>
<tr>
<td>3 &gt;= 5</td>
<td>False</td>
</tr>
<tr>
<td>1 &lt; 3 &lt; 5</td>
<td>True</td>
</tr>
</tbody>
</table>
## Comparisons

<table>
<thead>
<tr>
<th>Condition</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 &lt; 5</td>
<td>True</td>
</tr>
<tr>
<td>3 != 5</td>
<td>True</td>
</tr>
<tr>
<td>3 == 5</td>
<td>False</td>
</tr>
<tr>
<td>3 &gt;= 5</td>
<td>False</td>
</tr>
<tr>
<td>1 &lt; 3 &lt; 5</td>
<td>True</td>
</tr>
<tr>
<td>1 &lt; 5 &gt; 3</td>
<td>True</td>
</tr>
</tbody>
</table>

But please don't do this
## Comparisons

<table>
<thead>
<tr>
<th>Condition</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$3 &lt; 5$</td>
<td>True</td>
</tr>
<tr>
<td>$3 != 5$</td>
<td>True</td>
</tr>
<tr>
<td>$3 == 5$</td>
<td>False</td>
</tr>
<tr>
<td>$3 &gt;= 5$</td>
<td>False</td>
</tr>
<tr>
<td>$1 &lt; 3 &lt; 5$</td>
<td>True</td>
</tr>
<tr>
<td>$1 &lt; 5 &gt; 3$</td>
<td>True</td>
</tr>
<tr>
<td>$3+2j &lt; 5$</td>
<td>error</td>
</tr>
</tbody>
</table>
created by

Greg Wilson

October 2010

Copyright © Software Carpentry 2010
This work is licensed under the Creative Commons Attribution License
See http://software-carpentry.org/license.html for more information.