Introduction to Python

Edward Loper

Outline

- Data
  - strings, variables, lists, dictionaries
- Control Flow
- Working with files
- Modules
- Functions
- Classes
- Etc.

Python

- Python is an open source scripting language.
- Developed by Guido van Rossum in the early 1990s
- Named after Monty Python
- Available on unagi, linc, gradient
  /pkg/p/Python2.1.1/bin/python
- Available for download from
  http://www.python.org

The Python Interpreter

- Interactive interface to Python
  % /pkg/p/Python-2.1.1/bin/python
  Python 2.1.1 (#6, Aug 28 2001, 15:43:27)
  [GCC 2.95.2 19991024 (release)] on sunos5
  Type "copyright", "credits" or "license" for more information.

- Enter an expression, and Python evaluates it:
  >>> 3*(7+2)
  27
**Strings**

- A string is a single piece of text.
- Strings are written ‘...’ or "...
  >>> "the king of spain"
  the king of spain
  >>> 'the king said "hello."'
  the king said "hello."
- Spaces are significant
  >>> 'the knights of ni'
  the knights of ni
- Backslashes mark special characters
  >>> 'hello
world' # '
' is a newline
  hello
  world

**Operations on Strings**

>>> 'the' + 'king'
  'theking'
>>> len('the df')
  6
  >>> 'the df'.count('the')
  1
  >>> 'the king'.replace('the', 'a')
  'a king'
  >>> 'the king'.upper()
  'THE KING'
  >>> ' hello there '.strip()
  'hello there'

**Variables**

- A variable is a name for a value.
  - Use "=" to assign values to variables.
    >>> first_name = 'John'
    >>> last_name = 'Smith'
    >>> first_name + ' ' + last_name
    'John Smith'
  - Variable names are case sensitive
  - Variable names include only letters, numbers, and "_"
  - Variable names start with a letter or "_"
  - Any variable can hold any value (no typing)

**Lists**

- A list is an ordered set of values
  - Lists are written [elt0, elt1, ..., elt_n]
    >>> [1, 3, 8]
    >>> ['the', 'king', 'of', ['spain', 'france']]
    >>> []
    >>> [1, 2, 'one', 'two']
  - lst[i] is the i**th** element of lst.
  - Elements are indexed from zero
    >>> words = ['the', 'king', 'of', 'spain']
    >>> words[0]
    'the'
    >>> words[2]
    'of'
Indexing Lists

```python
>>> lst = ['a', 'b', 'c', ['d', 'e']]
>>> determiners[0]
'a'
>>> determiners[-2]
'c'
>>> determiners[-1][0]
'd'
>>> determiners[0:2]
['a', 'b']
>>> determiners[2:]
['c', ['d', 'e']]
```

Operations on Lists

```python
>>> determiners = ['the', 'an', 'a']
>>> len(determiners)
3
>>> determiners + ['some', 'one']
['the', 'an', 'a', 'some', 'one']
>>> determiners.index('a')
2
>>> [1, 1, 2, 1, 3, 4, 3, 6].count(1)
3
```

Operations on Lists 2: List Modification

```python
>>> determiners
['the', 'an', 'a']
>>> del determiners[2]  # remove the element at 2
>>> determiners.append('every')  # insert at the end of the list
>>> determiners.insert(1, 'one')  # insert at the given index
>>> determiners
['the', 'one', 'an', 'every']
>>> determiners.sort()  # sort alphabetically
>>> determiners
['an', 'every', 'one', 'the']
>>> determiners.reverse()  # reverse the order
>>> determiners
['the', 'one', 'every', 'an']
```

Lists and Strings

- Strings act like lists of characters in many ways.
  ```python
  >>> 'I saw a man with a telescope'[-9:]
  'telescope'
  >>> list('a man')
  ['a', ' ', 'm', 'a', 'n']
  >>> 'a man'.split()
  ['a', 'man']
  >>> str(['a', 'man'])
  '[a, man]'
  >>> 'a man'.join(['a', 'man'])
  'a-man'
  ```
- Converting lists to strings:
  ```python
  >>> list('a man')
  ['a', ' ', 'm', 'a', 'n']
  >>> 'a man'.split()
  ['a', 'man']
  >>> str(['a', 'man'])
  '[a, man]'
  >>> 'a man'.join(['a', 'man'])
  'a-man'
  ```
Dictionaries

- A dictionary maps keys to values
  - Like a list, but indexes (keys) can be anything, not just integers.
  - Dictionaries are written `{key: val, ...}`
  - Dictionaries are indexed with `dict[key]`

Operations on Dictionaries

```python
>>> determiners = {'the': 'def', 'an': 'indef', ...
... 'a': 'indef'}
>>> determiners.keys()
['an', 'a', 'the']
>>> determiners.has_key('an')
1 # 1 is true
>>> del determiners['an']
>>> determiners.has_key('an')
0 # 0 is false
>>> determiners.items()
['the': 'def', 'a': 'indef']
```

Truth Values

- Every expression has a truth value
  - 0 is false, all other numbers are true.
  - '' is false, all other strings are true.
  - [] is false, all other lists are true.

```python
>>> 5 == 3+2
1 # == tests for equality
>>> 5 != 3+2
1 # != tests for inequality
>>> 5 > 3+2
0 # >, <, <=, == test for ordering
>>> 5 > 3+2 or 5<3+2
0 # or, and combine truth values
```

Control Flow

- if statement tests if a condition is true
  - If so, it executes a body
  - Otherwise, it does nothing

```python
body
...
... del determiners[3]
... print 'deleted the 3rd determiner'
```

- Indentation is used to mark the body.
- Note the ‘:’ at the end of the if line.
Control Flow 2

- **if-else statement**
  ```python
text = 'too short'
if len(text) > 3:
    print text
else:
    print 'too short'
```

- **if-elif statement**
  ```python
x = 10
if x < 3:
    print x*3
elif x < 6:
    print x*2
else:
    print x
```

- **while statement**
  ```python
x = 0
while x < 1000:
    print x
    x = x**2 + 3
```

- **for statement**
  ```python
for n in range(0, 10):
    print n*n
```

Working with Files

- **To read a file:**
  ```python
for line in open('corpus.txt', 'r').readlines():
    print line
```

- **To write to a file:**
  ```python
outfile = open('output.txt', 'w')
outfile.write(my_string)
outfile.close()
```

  **Example:**
  ```python
outfile = open('output.txt', 'w')
for line in open('corpus.txt', 'r').readlines():
    outfile.write(line.replace('a', 'some'))
outfile.close()
```

Modules

- **A module** is a collection of useful operations and objects.
  - e.g., networking, graphics, threads, NLP
- **Access modules with import**
  ```python
import re
re.search('[ab]c+', string)
```

- **Or use from...import**
  ```python
from re import search
search('[ab]c+', string)
```

Getting Help

- The **pydoc module** can be used to get information about objects, functions, etc.
  ```python
from pydoc import help
help(re)
```

- pydoc can also be used from the command line, to provide manpage-like documentation for anything in Python:
  ```bash
% pydoc re
dir() lists all operations and variables contained in an object (list, string, etc):
```
Functions

• A function is a reusable piece of a program.
• Functions are defined with `def`

```python
>>> def square(x):
...    return x*x
>>> print square(8)
64
```

• Optional arguments:

```python
>>> def power(x, exp=2):
...    if x <= 0:
...        return 1
...    else:
...        return x*power(x, exp-1)
```

Classes

• A class is a kind of object (like lists or strings) that contains variables and operations (or methods)
• The simplest class:

```python
>>> class Simple: pass
```

• Class objects are created with the constructor, which has the same name as the class:

```python
>>> obj = Simple()
```

• Variables are accessed as `obj.var`

```python
>>> obj.x = 3
```

An Example Class

```python
>>> class Account:
...    def __init__(self, initial):
...        self.balance = initial
...    def deposit(self, amt):
...        self.balance = self.balance + amt
...    def withdraw(self, amt):
...        self.balance = self.balance - amt
...    def getbalance(self):
...        return self.balance
```

• `__init__` defines the constructor
• `self` is the object that is being manipulated.
  – It is the first argument to every method.

Using the example class

```python
>>> a = Account(1000.00)
>>> a.deposit(550.23)
>>> print a.getbalance()
1550.23
>>> a.deposit(100)
>>> a.withdraw(50)
>>> print a.getbalance()
1600.23
```