Scripting and Python

Mark Howison

User Services & Support
Tue 7/26
Barus and Holley, Room 190
1:00 - 2:00 Welcome & Introduction to CCV
2:00 - 3:00 Effective use of CCV resources
3:00 - 3:30 [Refreshments]
3:30 - 4:30 TeraGrid

Wed 7/27
ROOM CHANGE: MacMillan 115
1:00 - 2:30 Scripting and Python
2:30 - 3:00 [Refreshments]
3:00 - 4:30 Programming with OpenMP

Thu 7/28
ROOM CHANGE: MacMillan 115
1:00 - 2:30 Programming with MPI
2:30 - 3:00 [Refreshments]
3:00 - 4:30 Programming with CUDA

Fri 7/29
Barus and Holley, Room 190
1:00 - 2:00 Visualization
2:00 - 2:30 Parallel I/O
2:30 - 3:00 [Refreshments]
3:00 - 4:30 Debugging and Profiling

http://www.brown.edu/CCV/workshops/summer2011
“Software Carpentry” Tutorials

- The Shell
  http://software-carpentry.org/4_0/shell/
- Python
  http://software-carpentry.org/4_0/python/
NumPy

- http://numpy.scipy.org/
- Adds scientific computing functionality to Python
  - Supports more specific data types (float32, int64, etc)
  - Efficient implementation of true arrays (not lists)
  - “f2py” easily integrates Fortran kernels into Python
  - Syntax and linear algebra routings are similar to Matlab
  - SciPy provides additional scientific routines built on top of NumPy
Dictionaries

- Also known as “hash table” or “lookup table” or “associative array”
- Keys are matched to values
- Created with `dict()` or with `{ key: value, ... }`
- Access values with `dict[key]`
  - Missing keys generate an exception

```python
>>> tel = dict()
>>> tel['guido']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'guido'
```
Dictionaries (Cont'd)

- **Common idiom:**

```python
>>> try:
...   tel['guido']
... except KeyError:
...   tel['guido'] = 0
```

- **More example usage:**

```python
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
>>> tel.keys()
['guido', 'irv', 'jack']
>>> 'guido' in tel
True
```
Key/value pairs are generated by `.iteritems()`

```python
>>> knights = {'gallahad': 'the pure', 'robin': 'the brave'}
>>> for k, v in knights.iteritems():
...    print k, v
...    gallahad the pure
    robin the brave
```

Can also access lists of keys with `.keys()` or of values with `.values()`

Like lists, dictionaries support `len()`

Test for key existance with `.has_key()` or with `key in dict`