PYTHON LAB BOOK

Python For Programmers
UCSC Extension Online

Lab 11 Packages

Topics

- Modules: `shutil`, `tempfile`
- Python Packages

v.4.0

lab10_1.py

```
 1 #!/usr/bin/env python
 2 """lab10_1.py Copy the cats.txt into your area.  Change the file so
 3 that all the cats become dogs and dogs become cats."""
 4
 5 import do_swap
 6 import shutil
 7
 8 def Swapper(file_name, apple, orange):
 9     """ Changes the text in the file, replacing apples with
10     oranges and oranges with apples."""
11     try:
12         open_file = open(file_name, "r+")
13     except IOError, msg:
14         print "Can't open", file_name, 'because', msg
15         return
16     try:  # Putting the whole file in memory!
17         text = open_file.read()
18         text = do_swap.DoSwap(text, apple, orange)
19         open_file.seek(0, 0)
20         open_file.truncate()
21         open_file.write(text)
22     finally:
23         open_file.close()
24
25 def main():
26     shutil.copy('cats.txt', 'cats2.txt')
27     Swapper('cats2.txt', 'cat', 'dog')
28     Swapper('www.txt', 'www', 'yyy')
29
30 if __name__ == '__main__':
31     main()
32 """
33 $ lab10_1.py
34 Can't open www.txt because [Errno 2] No such file or directory: 'www.txt'
35 $ cat cats2.txt
36
37 In my house we have 3 dogs who love to tease our old cat.  The old cat
38 gets quite bewildered when they take turns running in front of him and
39 getting in his way.  He steps around one dog, then the next dog, then
40 the next dog, just to find the first dog again in his way.  However,
41 at nap time, they all curl up together, 3 dogs and one old cat.
42 """
```

lab10_1_2.py

```
 1 #!/usr/bin/env python
 2 """lab10_1_2.py -- lab10_1 again.  This time using the builtin file
 3 iterator, so that all the file isn't in memory at one time, and
 4 tempfile."""
 5
 6 import os
 7 import shutil
 8 import tempfile
 9 import do_swap
10
11 def Swapper(file_name, apple, orange):
12     """ Changes the text in the file, replacing apples with
13     oranges and oranges with apples."""
14     try:
15         open_file = open(file_name)
16     except IOError, msg:
17         print "Cannot open", file_name, 'because', msg
18         return
19     t_fd, t_name = tempfile.mkstemp()
20     try:
21         for line in open_file:
22             swapped_line = do_swap.DoSwap(line, apple, orange)
23             os.write(t_fd, swapped_line)
24     finally:
25         open_file.close()
26         os.close(t_fd)
27     os.rename(t_name, file_name)
28
29 def main():
30     shutil.copy('cats.txt', 'cats2.txt')
31     Swapper('cats2.txt', 'cat', 'dog')
32     Swapper('www.txt', 'www', 'yyy')
33
34 if __name__ == '__main__':
35     main()
36 """
37 $ lab10_1_2.py
38 Can't open www.txt because [Errno 2] No such file or directory: 'www.txt'
39 $ cat cats2.txt
40
41 In my house we have 3 dogs who love to tease our old cat.  The old cat
42 gets quite bewildered when they take turns running in front of him and
43 getting in his way.  He steps around one dog, then the next dog, then
44 [The rest of the output is deleted.]"""
```

lab10_2.py

```
 1 #!/usr/bin/env python
 2 """lab10_2.py  Uses os.path.walk() and the previous exercise
 3 to change all the cats to dogs and dogs to cats."""
 4
 5 import os
 6 import sys
 7 import shutil
 8 import lab10_1 as swapper   # was the previous exercise
 9
10 def SwapTextFiles(swappers, dir_name, files):
11     """ Called by walk to change the text in all files.
12 Notice that the first argument can be any object.  Here we pass
13 a 2-tuple to unpack."""
14     try:
15         apple, orange = swappers
16     except ValueError:
17         print >> sys.stderr, 'swappers must be a 2-tuple of words to swap'
18         sys.exit(1)
19     for file_name in files:
20         file_name = os.path.join(dir_name, file_name)
21         # We're not touching directory names
22         if not os.path.isfile(file_name):
23             continue
24         swapper.Swapper(file_name, apple, orange)
25
26 def PrintFiles(dummy, dir_name, files):
27     for file_name in files:
28         file_name = os.path.join(dir_name, file_name)
29         if not os.path.isfile(file_name):
30             continue
31         print file_name + ':'
32         for line in open(file_name):
33             print line,
34         print "---"
35
36
37 def main():
38     if len(sys.argv) > 1:
39         dir_ = sys.argv[1]
40     else:
41         try:
42             shutil.rmtree("cats2")
43         except OSError: # not there
44             pass
```

```
45         shutil.copytree("./cats", "./cats2")
46         dir_ = 'cats2'
47     os.path.walk(dir_, PrintFiles, None)
48     os.path.walk(dir_, SwapTextFiles, ('cat','dog'))
49     os.path.walk(dir_, PrintFiles, None)
50
51 if __name__ == '__main__':
52     main()
53 """
54 $ lab10_2.py
55 cats2/cats.txt:
56
57 In my house we have 3 cats who love to tease our old dog.  The old dog
58 gets quite bewildered when they take turns running in front of him and
59 getting in his way.  He steps around one cat, then the next cat, then
60 the next cat, just to find the first cat again in his way.  However,
61 at nap time, they all curl up together, 3 cats and one old dog.
62
63 ---
64 cats2/more_cats.txt:
65
66 In my house we have 3 cats who love to tease our old dog.  The old dog
67
68 [much skipped]
69
70 cats2/deep_cats/deeper_cats/cats.txt:
71
72 In my house we have 3 dogs who love to tease our old cat.  The old cat
73 gets quite bewildered when they take turns running in front of him and
74 getting in his way.  He steps around one dog, then the next dog, then
75 the next dog, just to find the first dog again in his way.  However,
76 at nap time, they all curl up together, 3 dogs and one old cat.
77
78 ---
79 cats2/deep_cats/deeper_cats/more_cats.txt:
80
81 In my house we have 3 dogs who love to tease our old cat.  The old cat
82 gets quite bewildered when they take turns running in front of him and
83 getting in his way.  He steps around one dog, then the next dog, then
84 the next dog, just to find the first dog again in his way.  However,
85 at nap time, they all curl up together, 3 dogs and one old cat.
86
87 ---
88 $"""
```

lab10_3.py

```
#!/usr/bin/env python
"""lab10_3.py  Write a function that takes a file name and
counts how many times each word appears in the file."""

from __future__ import division
import sys
import string

def CountWords(text, total=0, counts_d=None):
    """Counts the number of each word in the text.  Returns a tuple =
    (total_words, dictionary of {"word":count, "word2":count, ...})
    If a counts_d dictionary, it gets updated with the new words found,
    otherwise a new dictionary is started.

    total and counts_d are initialized in the argument list, just in
    case we ever want to use this to accumulate the word count through
    multiple files"""
    if not counts_d:
        counts_d = {}
    text = text.lower().split()
    for word in text:
        word = word.strip(string.punctuation)
        if not word:
            continue
        total += 1
        if word in counts_d:
            counts_d[word] += 1
        else:
            counts_d[word] = 1
    return total, counts_d

def FindPopularWords(counts_d):
    """Returns a list of (word, count) pairs from the dictionary
    counts_d -- but only the 10 most popular words and all words that
    are as popular."""

    def ValueKey(this):
        """Sort by the value."""
        return counts_d[this]

    counts_l = []
    words = sorted(counts_d, key=ValueKey)
    # taking the first 10 and any more that tie for 10th
    # but maybe there are only 10 or fewer
```

```
45      number_of_words = len(words)
46      for word in reversed(words):
47          if number_of_words > 10 and \
48                 counts_d[word] < counts_d[words[-10]]:
49              break
50          counts_l += [(word, counts_d[word])]
51      return counts_l
52
53  def GenWordReport(file_name, total, counts_d):
54      """Returns a string containing the report of the words in the
55      counts_d dictionary, and total words given."""
56      if not total:
57          return "No words found."
58      counts_l = FindPopularWords(counts_d)
59      ret_str = "\n%s: %d total words" % (file_name, total)
60      ret_str += "\ncount  %    word\n"
61      for word, hits in counts_l:
62          ret_str += "%5d %.1f : %s\n" % (hits, hits * 100/total, word)
63      return ret_str
64
65  def GetText(file_name):
66      try:
67          file_obj = open(file_name)
68          try:
69              text = file_obj.read()
70          finally:
71              file_obj.close()
72      except IOError, msg:
73          print >> sys.stderr, "Can't read", file_name, ':', msg
74          sys.exit(1)
75      return text
76
77  def PopularWordReport(file_name):
78      """Driver for the program.  Returns a the popular word report for
79      the file_name."""
80      text = GetText(file_name)
81      total, counts_d = CountWords(text)
82      return GenWordReport(file_name, total, counts_d)
83
84  def main():
85      try:
86          file_name = sys.argv[1]
87      except IndexError:
88          file_name = 'zen.story'
89      print PopularWordReport(file_name)
```

```
 90
 91 if __name__ == '__main__':
 92     main()
 93
 94 """
 95 $ lab10_3.py
 96
 97 zen.story: 76 total words
 98
 99 count  %    word
100     4 5.3 : the
101     4 5.3 : a
102     3 3.9 : you
103     3 3.9 : and
104     2 2.6 : zen
105     2 2.6 : without
106     2 2.6 : who
107     2 2.6 : to
108     2 2.6 : through
109     2 2.6 : that
110     2 2.6 : zen
111     2 2.6 : without
112     2 2.6 : who
113     2 2.6 : to
114     2 2.6 : through
115     2 2.6 : that
116     2 2.6 : sword
117   [more deleted]
118 $ cat zen.story
119 A general in ancient China came to see a Zen master.
120
121 He drew his sword and pointed it at the teacher, and
122 announced: "Don't you know that I am a man who can
123 run you through without blinking an eye?"
124
125 To which the Zen master responded instantly: "Don't
126 you know that I am a man who can be run through
127 without blinking an eye?"
128
129 Deeply impressed, the general sheathed his sword and
130 remained for the teaching.
131 """
```

lab10_4.py

```
#!/usr/bin/env python
"""lab10_4.py starting_dir

Count and report the popular words in all the files, walking from
the starting directory.
"""
import os
import sys
import lab10_3 as words

def CountFiles(stats, directory, f_names):
    for f in f_names:
        whole_path = os.path.join(directory, f)
        if os.path.isdir(whole_path):
            continue
        stats[0], stats[1] = words.CountWords(
            words.GetText(whole_path), stats[0], stats[1])

def PopularWordWalk(starting_dir):
    stats = [0, {}]
    os.path.walk(starting_dir, CountFiles, stats)
    total, counts_d = stats
    return words.GenWordReport(starting_dir, total, counts_d)

def main():
    if len(sys.argv) != 2:
        print __doc__
        sys.exit(1)
    print PopularWordWalk(sys.argv[1])

if __name__ == '__main__':
    main()
"""
$ lab10_4.py ..

..: 98250 total words
count  %    word
 2558 2.6 : the
 1808 1.8 : a
 1549 1.6 : in
 1528 1.6 : print
[rest deleted]
"""
```

Python Packages

Packages are a collection of modules, in a directory structure. For example, here is some of the directory structure for Mailman, a popular listserver program written in Python:

```
.
|-- Mailman
|   |-- Archiver
|   |   |-- Archiver.py
|   |   |-- HyperArch.py
|   |   |-- HyperDatabase.py
|   |   |-- __init__.py
|   |   |-- pipermail.py
|   |-- Autoresponder.py
|   |-- Bouncer.py
|   |-- Bouncers
|   |   |-- BouncerAPI.py
|   |   |-- Compuserve.py
|   |   |-- [ 15 deleted ]
|   |   |-- Yale.py
|   |   |-- __init__.py
|   |-- Cgi
|   |   |-- Auth.py
|   |   |-- __init__.py
|   |   |-- admin.py
|   |   |-- [ 10 deleted ]
|   |   |-- subscribe.py
|   |-- Commands
|   |   |-- __init__.py
|   |   |-- cmd_confirm.py
```

```
(continued)
|   |   |-- [ 14 deleted ]
|   |-- Defaults.py
|   |-- Deliverer.py
|   |-- Digester.py
|   |-- Errors.py
|   |-- GatewayManager.py
|   |-- Gui
|   |   |-- Archive.py
|   |   |-- Autoresponse.py
|   |   |-- [ 13 deleted ]
|   |   |-- Usenet.py
|   |   |-- __init__.py
|   |-- HTMLFormatter.py
|   |-- Handlers
|   |   |-- Acknowledge.py
|   |-- [ many deleted ]
|   |-- Version.py
|   |-- __init__.py
|   |-- htmlformat.py
|   |-- i18n.py
|   |-- mm_cfg.py
|   |-- mm_cfg.py.dist
|   |-- versions.py
```


Note that each directory has an __init__.py file. If so, then the `import` facility will descend into the directory when it looks for modules.

Here's part of the directory structure for our class materials:

```
/labs
 |-- /lab_07_Important_Trick
 |   |-- lab07_1.py
 |-- /lab_08_Comprehensions
 |   |-- lab08_2.py
 |   |-- lab08_4.py
 |-- /lab_09_Dictionaries
 |   |-- py_dict.py
 |-- /lab_10_File_IO
 |   |-- /cats
 |   |-- cats.txt
 |   |   |-- cats.txt
 |   |   |-- /deep_cats
 |   |   |   |-- cats.txt
 |   |   |   |-- /deeper_cats
 |   |   |   |   |-- cats.txt
 |   |   |   |   |-- more_cats.txt
 |   |   |   |-- more_cats.txt
 |   |   |-- more_cats.txt
 |   |-- do_swap.py
 |   |-- lab10_1.py
 |   |-- zen.story
 |-- /lab_11_Packages
 |   |-- /apple
 |   |   |-- /banana
 |   |   |   |-- total_text.py
 |   |   |-- /work_here
 |-- /lab_12_Function_Fancies
```


I want to `import` the `lab_08_Comprehensions/lab08_2.py` module while working in `lab_11_Packages`.

Three things must happen for this to work:

1. The directory I want to start from must be on the `sys.path`.

   ```
   >>> import sys
   >>> sys.path
   ['', '/usr/lib/python25.zip', '/usr/lib/python2.5', (more deleted) ]
   >>> sys.path.insert(0, '..')
   >>> sys.path
   ['..', '', '/usr/lib/python25.zip', '/usr/lib/python2.5', (more deleted) ]
   ```

   Now the `labs` directory is first on the search path for `import`s.

2. There must be an `__init__.py` file in every directory I want the `import` facility to descend into. __init__.py can be, and usually is, empty.

```
$ cd lab_11_Packages
$ touch ../lab_08_Comprehensions/__init__.py
```

3. In my new program, I must concoct an `import` line that starts with a subdirectory of something on my search path. It is dotted with any sibdirectores I want the `import` facility to descend into. Each of the subdirectories must have its own `__init__.py`. After the last dot comes the name of the module I want to import, but without the `.py`.

```
>>> import lab_08_Comprehensions.lab08_2
```

Note that all importable modules *must* end in `.py` and must not have any other dots in their name; and that the `import` line has no `.py`'s.

Now I can use it:

```
>>> dir(lab_08_Comprehensions.lab08_2)
['Cards', '__builtins__', '__doc__', '__file__', '__name__', 'main']
>>> help(lab_08_Comprehensions.lab08_2.Cards)
Help on function Cards in module lab_08_Comprehensions.lab08_2:

Cards()
    Return a deck of cards as a list of strings.

>>> lab_08_Comprehensions.lab08_2.Cards()[:5]
['2 of Clubs', '3 of Clubs', '4 of Clubs', '5 of Clubs', '6 of Clubs']
>>>
```


Here you might like the keyword `as`:

```
>>> import lab_08_Comprehensions.lab08_2 as cards
```

Then you can type `cards` instead of `lab_08_Comprehensions.cards`, and still keep your code clear about the fact that cards.Cards is defined in another module:

```
>>> cards.Cards()[36]
'Queen of Hearts'
>>>
```

```
Answers for Quiz (Lab 11)

>>> L = [1, 2, 3]
>>> T = (1, L, 3)
>>> print T
(1, [1, 2, 3], 3)
>>> L[1] = 0
>>> print T
(1, [1, 0, 3], 3)
>>> T[1] = 0
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object doesn't support item assignment
>>>
>>> friends = ['Abe', 'Bob', 'Carl']
>>> enemies = friends[:]
>>> enemies[1] = 'Brian'
>>> print friends
['Abe', 'Bob', 'Carl']
>>> print enemies
['Abe', 'Brian', 'Carl']

Bad way because you make your friends vulnerable:
>>> friends = ['Abe', 'Bob', 'Carl']
>>> enemies = friends
>>> enemies[1] = 'Brian'
>>> print enemies
['Abe', 'Brian', 'Carl']
>>> print friends
['Abe', 'Brian', 'Carl']
>>>
```

Draw a line from each object to its mutability:

numbers	immutable
strings	"
tuples	"
lists	mutable
dictionaries	"
dictionary keys	immutable

Lab 11

Collect and extract `labs.zip` from WebCT. You need:

- `/lab_11_Packages/apple` directory structure
- `labs/lab_11_Packages/numbers.txt`

1. The apple directory structure looks like:

```
/apple
 |-- /banana
 |   |-- total_text.py
 |-- /work_here
```

   Bring up the interpreter while the `work_here` directory is your current working directory. Do whatever is necessary: create a __init__.py in the right place; alter the `sys.path` to force the `import` facility to look into the `apple` directory; and use dot notation to import `total_text.py` into the interpreter and use `dir` and `help` to discover why it is useful to you for this exercise, and how to use it. The exercise:

   Now, do the import in a program in the `work_here`. Have your program take a file path as an argument on the command line, or get the file path interactively, and report the total of all the numbers in the file. My test file is `numbers.txt`. Put it anywhere you like.

   If you have time and interest, make your program robust, i.e., catch all possible errors and react appropriately.

2. Now, make a program in any directory you like that imports your previous module and walks a directory structure, adding up all the numbers in all the files in the directory structure.

3. (Optional) Use the walk command to make a tree.py, which emulates the old Linux tree command. Output from my solution is in the exercise above.

   You may not have time to get the output just right, but if you gather all all the files and directories and sort them, you've done enough.

   If you have time to present the data you collected, you'll find `os.path.join()` and `os.path.split()` useful. The help facility for `os.path.join()` is cryptic, but the online documentation is helpful.