04-files

February 3, 2017

1 Files

2 tempfile module

- will create a valid temporary file pathname on any OS

```
In [6]: import os
        import tempfile

        # does not create the file, just a pathname
        tp = tempfile.NamedTemporaryFile().name
        tp2 = tempfile.NamedTemporaryFile().name

        # os.path.exists(path) - True if file path exists

        [tp, tp2, os.path.exists(tp), os.path.exists(tp2)]
```

```
Out[6]: ['/var/folders/7p/8hg8wwy575z8m7n_bc4mjsdc0000gn/T/tmp9z9t0pqd',
         '/var/folders/7p/8hg8wwy575z8m7n_bc4mjsdc0000gn/T/tmpfejfhmnr',
         False,
         False]
```

3 Getting file status

```
In [11]: # os.path.exists and os.access reports file status without throwning erro
         # os.stat throws an error if the path doesn't exist.

         # similar to touch command - make an empty file

         open(tp, 'w').close()

         def ac(p):
             # can check exists, readable, writeable, executable
             return([ os.access(p, m) for m in [os.F_OK, os.R_OK, os.W_OK, os.X_OK]

         ac(tp)
```

```
Out[11]: [True, True, True, False]

In [13]: # last accessed time, last modified time

         [os.path.getatime(tp), os.path.getmtime(tp)]

Out[13]: [1486146685.0, 1486146971.0]

In [14]: [os.path.isfile(tp), os.path.isdir(tp)]

Out[14]: [True, False]

In [15]: [os.path.isfile(tp), os.path.isdir(tp)]

Out[15]: [True, False]

In [16]: os.stat(tp)

Out[16]: os.stat_result(st_mode=33206, st_ino=29977248, st_dev=16777220, st_nlink=1

In [17]: # removes a file, but throws error if it doesn't exist

         os.remove(tp)
         ac(tp)

Out[17]: [False, False, False, False]

In [18]: # file is gone

         os.path.exists(tp)

Out[18]: False

In [19]: # stat gets upset and throws an error if the file doesn't exist

         os.stat(tp)


        ---------------------------------------------------------------------------------

        FileNotFoundError                         Traceback (most recent call last)

        <ipython-input-19-9dca2872f6ed> in <module>()
          1 # stat gets upset and throws an error if the file doesn't exist
          2
    ----> 3 os.stat(tp)


        FileNotFoundError: [Errno 2] No such file or directory: '/var/folders/7p/8h
```

```
In [52]: # Returns list of files and dirs in a directory
         # can use isfile and isdir to figure out which is which

         fds = os.listdir( os.path.join(os.path.expanduser('~')))

         # find anaconda dir

         for fd in fds:
             if fd.startswith(('Anaconda', 'anaconda')):
                 anaconda = fd
         anaconda

Out[52]: 'anaconda'
```

4 'walk' and get all the files and dirs under a start dir

```
In [57]: # returns a generator...

         e = os.path.expanduser('~/' + anaconda + '/etc')
         print(e)
         g = os.walk(e)
         g

/Users/lstead/anaconda/etc


Out[57]: <generator object walk at 0x106be9a40>

In [58]: # nicer than os.listdir() in that files and dirs are in separate lists
         # returns (dirpath, dirs in dirpath, files in dir)

         [tup for tup in g]

Out[58]: [('/Users/lstead/anaconda/etc', ['fish', 'jupyter'], ['.DS_Store']),
          ('/Users/lstead/anaconda/etc/fish', ['conf.d'], []),
          ('/Users/lstead/anaconda/etc/fish/conf.d', [], ['conda.fish']),
          ('/Users/lstead/anaconda/etc/jupyter',
           ['nbconfig'],
           ['jupyter_notebook_config.json']),
          ('/Users/lstead/anaconda/etc/jupyter/nbconfig',
           [],
           ['notebook.json', 'tree.json'])]
```

5 open function

- used to open files for reading and writing

6 Writing files

- no automatic newlines

```
In [28]: # open file, write to file descriptor, close file descriptor
         # can be error prone - easy to forget to close. also, if there
         # is an error, the close call could be skipped
         # not closing file descriptors can cause a server to crash
         # 'w' is the 'open mode' - tells 'open' to open the file for writing

         fd = open(tp, 'w')
         for e in ['one', 'two', 'three', 'four']:
             fd.write(e + '\n')
         fd.close()
```

7 with

- 'with' is a 'context manager'
- binds return value from open to 'fd'
- 'with' will automatically close the file when the 'with' block is exited, even if by error
- note ':' and indenting defines a statement block over which 'fd' will be bound

```
In [29]: with open(tp, 'w') as fd:
             for e in ['one', 'two', 'three', 'four']:
                 fd.write(e + '\n')
```

```
In [30]: # could do one write with join

         with open(tp, 'w') as fd:
             fd.write('\n'.join(['one', 'two', 'three', 'four']))
```

```
In [32]: # or write out the string with newlines

         with open(tp, 'w') as fd:
             fd.write("one\ntwo\nthree\nfour\n")
```

```
In [33]: # can append(open mode 'a') to an existing file

         with open(tp, 'a') as f:
             for l in ['five', 'six']:
                 f.write(l + '\n')
```

8 print function output can goto a file

```
In [34]: with open(tp2, "w") as f:
             print(1,2,3,4,sep='\n', file=f)

         with open(tp2, 'r') as f:
             print(f.read())
```

```
1
2
3
4
```

9 Reading files - eager

- read the entire file immediately

```
In [35]: # eager read - read the entire file into one string
         # 'r' tells 'open' to open the file for reading

         with open(tp, 'r') as fd:
             print( fd.read())
```

```
one
two
three
four
five
six
```

```
In [36]: # eager read - get a list of all the lines

         with open(tp,'r') as fd:
             print(fd.readlines())
```

```
['one\n', 'two\n', 'three\n', 'four\n', 'five\n', 'six\n']
```

10 Reading files - lazy

- suppose you are looking for a substring in a huge unsorted file of text lines
  - lazy read probably wins
  - don't have to read in entire file before you can start search
  - don't have to allocate memory to hold the whole file
  - once you find the substring, you don't have to read the rest of the file

```
In [37]: # read one line at a time

         with open(tp, 'r') as fd:
             while True:
                 x = fd.readline()
```

```
                # returns empty string when finished
                if x == '':
                    break;
                print(x)
```

```
one

two

three

four

five

six
```

```
In [38]: # note double spacing
         # each line in the file has a newline, plus print is adding one
         # can turn off the print newline with keyword arg 'end'

         with open(tp, 'r') as fd:
             while True:
                 x = fd.readline()
                 # returns empty string when finished
                 if x == '':
                     break;
                 print(x, end='')
```

```
one
two
three
four
five
six
```

```
In [39]: fd = open(tp, 'r')
         fd

Out[39]: <_io.TextIOWrapper name='/var/folders/7p/8hg8wwy575z8m7n_bc4mjsdc0000gn/T/

In [40]: # a file descriptor is an iterator over the file lines

         [fd, iter(fd), fd is iter(fd)]

Out[40]: [<_io.TextIOWrapper name='/var/folders/7p/8hg8wwy575z8m7n_bc4mjsdc0000gn/T
          <_io.TextIOWrapper name='/var/folders/7p/8hg8wwy575z8m7n_bc4mjsdc0000gn/T
          True]
```

```
In [41]: next(fd)

Out[41]: 'one\n'

In [42]: # don't have to finish iterator...

         next(fd)

Out[42]: 'two\n'

In [43]: # note with readline and readlines each line has a trailing '\n',
         # which you usually don't want
         # use strip() to remove
         # can this cause a problem?

         'one\n'.strip()

Out[43]: 'one'

In [44]: # read N chars at a time

         with open(path, 'r')  as f:
             while True:
                 s = f.read(3)
                 if s == '':
                     break;
                 print(s)


one

tw
o
t
hre
e
f
our




In [45]: # ... or can finish iterator later on

         [next(fd), next(fd), next(fd), next(fd)]

Out[45]: ['three\n', 'four\n', 'five\n', 'six\n']

In [46]: # exhausted, can not be used again

         next(fd)
```

```
    ---------------------------------------------------------------------------

    StopIteration                             Traceback (most recent call last)

    <ipython-input-46-af6cbe3afd01> in <module>()
      1 # exhausted, can not be used again
      2
----> 3 next(fd)


    StopIteration:
```

11 In memory "files"

- very useful
- doc

```
In [47]: import io

         ios = io.StringIO()

         print('one', file=ios)
         ios.write('two')

         ios.getvalue()

Out[47]: 'one\ntwo'

In [48]: ios = io.StringIO('asdfasdf')

         ios.read()

Out[48]: 'asdfasdf'
```

12 shutil module

- has utilities for reading and writing:
  - compressed files - 'gzip', 'bz2'
  - file archives - 'zip', 'tar', 'hdf5'
- move and copy files
- doc