Modules

- Modules are usually .py files. A module contains Python code - typically function, class and constant value defintions that are generally useful.
- Not all modules are .py files. For example the `sys` module is builtin. And some modules are built in C. But all behave and are used in the same way.
- Modules are used by importing them using import statements usually at the beginning of a .py file. Syntax:

```
import mod1
import mod1, mod2, ..., modn
import mod as modname
# Using the syntax below can lead to name clashes
from mod import object as modname
from mod import object1, object2,..., objectN
from mod import (object1, object2,..., objectN)
from mod import *
```

Packages

- Packages organize modules that belong together. The organization can use sub-packages.
- Consider the example of an audio processing package.

```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
              __init__.py
              wavread.py        Individual module in sub-package
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/                  Subpackage for sound effects
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
      filters/                  Subpackage for filters
              __init__.py
              equalizer.py
              vocoder.py
              karaoke.py
              ...
Source: Python tutorial (at www.python.org)
```

Example: package, module

```
import os
print(os.path.basename(filename))
# safe fully qualified access

import os.path as path
print(path.basename(filename))
# risk of name collision with path

from os import path
print(path.basename(filename))
# risk of name collision with path

from os.path import basename
print(basename(filename))
# risk of name collision with basename

from os.path import *
print(basename(filename)) # risk of many name collisions
```

Files

Files are sequences of objects typically stored on disk. These can be read and/or written.

```
open(file, mode='r', buffering=-1, encoding=None,
errors=None, newline=None, closefd=True, opener=None)
```

open returns a file object (called descriptor). It has one mandatory argument and 7 default arguments. The only important one for common usage is mode. Default mode is 'rt' that is read and text.

For details of other arguments see Python documentation.

Commonly used file functions

open(fn, 'w') fn is a string representing a file name. Creates a file for writing and returns a file object/descriptor.

open(fn, 'r') fn is a string representing a file name. Opens an existing file for reading and returns a file object.

open(fn, 'a') fn is a string representing a file name. Opens an existing file for appending and returns a file object.

fd.read() returns a string containing the contents of the file associated with the file object fd.

fd.readline() returns the next line in the file associated with the file object fd.

fd.readlines() returns a list each element of which is one line of the file associated with the file handle fd.

fd.write(s) writes the strings to the end of the file associated with the file handle fd.

fd.writeLines(S) S is a sequence of strings. Writes each element of S as a separate line to the file associated with the file object fd.

fd.close() closes the file associated with the file object fd.

File: example

Program to read a text file (of lines) and reverse each line in the file and the whole file.

File: example

Program to read a text file (of lines) and reverse each line in the file and the whole file.

```
def revFile(fn): #fn is the file name
    rfd=open(fn) #file for reading
    wfd=open('R'+fn,'w') # output file has reversed line
    def rev():
        nonlocal rfd, wfd
        line=rfd.readline()
        if line=='':
            return
        else:
            rev()
            wfd.write(line[::-1])
    rev()
    rfd.close()
    wfd.close()
```