Content Delivered in class_10_18-August-2016

- Chapter 6: Modules
    - Methods of importing
        - Full module import
        - selective import
        - wild card import
    - sys module
    - os module

Assignments Given:

Assignment 1: exceute this, and observe the output

Modules

- Both buitin (ex: os), installed (ex: django) or user-defined
- It is a collection of functions, to serve a particular purpose
- imported in the functions, using 'import'
- Not all modules will be part of basic distribution
- To install a new module, **pip install moduleName**
- To search a module, **pip search moduleName**

In [2]:
```
import sys
```

In [3]:
```
print dir(sys)
```

```
['__displayhook__', '__doc__', '__excepthook__', '__name__', '__package__',
 '__stderr__', '__stdin__', '__stdout__', '_clear_type_cache', '_current_fram
es', '_getframe', '_mercurial', 'api_version', 'argv', 'builtin_module_name
s', 'byteorder', 'call_tracing', 'callstats', 'copyright', 'displayhook', 'dl
lhandle', 'dont_write_bytecode', 'exc_clear', 'exc_info', 'exc_type', 'except
hook', 'exec_prefix', 'executable', 'exit', 'exitfunc', 'flags', 'float_inf
o', 'float_repr_style', 'getcheckinterval', 'getdefaultencoding', 'getfilesys
temencoding', 'getprofile', 'getrecursionlimit', 'getrefcount', 'getsizeof',
 'gettrace', 'getwindowsversion', 'hexversion', 'long_info', 'maxint', 'maxsi
ze', 'maxunicode', 'meta_path', 'modules', 'path', 'path_hooks', 'path_import
er_cache', 'platform', 'prefix', 'ps1', 'ps2', 'ps3', 'py3kwarning', 'setchec
kinterval', 'setprofile', 'setrecursionlimit', 'settrace', 'stderr', 'stdin',
 'stdout', 'subversion', 'version', 'version_info', 'warnoptions', 'winver']
```

```
In [4]: sys.version
```

```
Out[4]: '2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Int
        el)]'
```

```
In [5]: sys.version_info
```

```
Out[5]: sys.version_info(major=2, minor=7, micro=12, releaselevel='final', serial=0)
```

```
In [6]: sys.winver
```

```
Out[6]: '2.7'
```

```
In [7]: sys.path
```

```
Out[7]: ['',
         'c:\\python27\\python27.zip',
         'c:\\python27\\DLLs',
         'c:\\python27\\lib',
         'c:\\python27\\lib\\plat-win',
         'c:\\python27\\lib\\lib-tk',
         'c:\\python27',
         'c:\\python27\\lib\\site-packages',
         'c:\\python27\\lib\\site-packages\\IPython\\extensions',
         'C:\\Users\\upethakamsetty\\.ipython']
```

User-defined modules are prioritized to builtin(or installed) modules

Methods of importing

```
import sys
from sys import *            # Not recommended by PEP 8
from sys import version
from sys import version as vr    # alias importing
```

```
In [10]: sys.version
```

```
Out[10]: '2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Int
         el)]'
```

```
In [11]: from sys import version
         version
```

```
Out[11]: '2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Int
         el)]'
```

```
In [12]: from sys import version as vr

         vr

Out[12]: '2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Int
         el)]'
```

```
In [13]: del vr
```

```
In [14]: vr

         ---------------------------------------------------------------------------
         NameError                                 Traceback (most recent call last)
         <ipython-input-14-8264846d88d6> in <module>()
         ----> 1 vr

         NameError: name 'vr' is not defined
```

Note: Selective importoptimizes the memory usage; but care should be taken when user-defined identifiers in the script/project have the same names as these functions of modules

```
In [2]: import os
```

```
In [5]: os.system("echo 'Hello World!'")

Out[5]: 0
```

```
In [6]: combinedDir = os.path.join('first', 'second', 'third', 'fourth')
```

```
In [7]: print 'combinedDir is ', combinedDir

        combinedDir is  first\second\third\fourth
```

```
In [8]: print os.path.exists(combinedDir)

        False
```

```
In [9]: os.getcwd()

Out[9]: 'C:\\Users\\upethakamsetty\\Google Drive\\python\\tut\\kt Group Batch 2\\clas
        s_10_18-August-2016'
```

```
In [10]: os.mkdir('newFolder')
```

```
In [11]: os.listdir(os.getcwd())

Out[11]: ['.ipynb_checkpoints', 'class_10_18-August-2016.ipynb', 'newFolder']
```

```
In [12]: os.makedirs(combinedDir)
In [13]: os.listdir(os.getcwd())

Out[13]: ['.ipynb_checkpoints', 'class_10_18-August-2016.ipynb', 'first', 'newFolder']

In [15]: os.listdir(os.getcwd())

Out[15]: ['.ipynb_checkpoints',
          'class_10_18-August-2016.ipynb',
          'first',
          'myTest.txt',
          'newFolder']

In [16]: os.rename('myTest.txt', 'myNewFile.txt')

In [17]: os.listdir(os.getcwd())

Out[17]: ['.ipynb_checkpoints',
          'class_10_18-August-2016.ipynb',
          'first',
          'myNewFile.txt',
          'newFolder']

In [23]: os.chdir('first/')

In [24]: os.listdir(os.getcwd())

Out[24]: ['second']

In [25]: os.chdir('..')    # changing to previous directories

In [26]: os.listdir(os.getcwd())

Out[26]: ['.ipynb_checkpoints',
          'class_10_18-August-2016.ipynb',
          'first',
          'myNewFile.txt',
          'newFolder']

In [27]: os.stat('myNewFile.txt')

Out[27]: nt.stat_result(st_mode=33206, st_ino=0L, st_dev=0L, st_nlink=0, st_uid=0, st_
         gid=0, st_size=0L, st_atime=1471487918L, st_mtime=1471487918L, st_ctime=14714
         87918L)

In [28]: modifiedTime = os.stat('myNewFile.txt').st_mtime

         print "myNewFile.txt was last modified on ", modifiedTime  # epoch time

         myNewFile.txt was last modified on  1471487918.46
```

In [29]:

```
from datetime import datetime
print datetime.fromtimestamp(modifiedTime)
```

```
2016-08-18 08:08:38.464941
```

In [30]:

```
print "myNewFile.txt was last modified on ", datetime.fromtimestamp(modifiedTi
me)
```

```
myNewFile.txt was last modified on  2016-08-18 08:08:38.464941
```

In [31]:
```
for dirpath, dirnames, filenames in os.walk('C:\Python27\Tools'):
        print 'Current Path:', dirpath
        print 'Directories:', dirnames
        print 'Files:', filenames
        print '-'*50
```

```
Current Path: C:\Python27\Tools
Directories: ['i18n', 'pynche', 'Scripts', 'versioncheck', 'webchecker']
Files: []
--------------------------------------------------
Current Path: C:\Python27\Tools\i18n
Directories: []
Files: ['makelocalealias.py', 'msgfmt.py', 'pygettext.py']
--------------------------------------------------
Current Path: C:\Python27\Tools\pynche
Directories: ['X']
Files: ['ChipViewer.py', 'ColorDB.py', 'DetailsViewer.py', 'html40colors.tx
t', 'ListViewer.py', 'Main.py', 'namedcolors.txt', 'pyColorChooser.py', 'pync
he.pyw', 'PyncheWidget.py', 'README.txt', 'StripViewer.py', 'Switchboard.py',
 'TextViewer.py', 'TypeinViewer.py', 'webcolors.txt', 'websafe.txt', '__init_
_.py']
--------------------------------------------------
Current Path: C:\Python27\Tools\pynche\X
Directories: []
Files: ['rgb.txt', 'xlicense.txt']
--------------------------------------------------
Current Path: C:\Python27\Tools\Scripts
Directories: []
Files: ['2to3.py', 'analyze_dxp.py', 'byext.py', 'byteyears.py', 'checkappen
d.py', 'checkpip.py', 'checkpyc.py', 'classfix.py', 'cleanfuture.py', 'combin
erefs.py', 'copytime.py', 'crlf.py', 'cvsfiles.py', 'db2pickle.py', 'diff.p
y', 'dutree.py', 'eptags.py', 'finddiv.py', 'findlinksto.py', 'findnocoding.p
y', 'find_recursionlimit.py', 'fixcid.py', 'fixdiv.py', 'fixheader.py', 'fixn
otice.py', 'fixps.py', 'google.py', 'gprof2html.py', 'h2py.py', 'hotshotmain.
py', 'ifdef.py', 'lfcr.py', 'linktree.py', 'lll.py', 'logmerge.py', 'mailerda
emon.py', 'md5sum.py', 'methfix.py', 'mkreal.py', 'ndiff.py', 'nm2def.py', 'o
bjgraph.py', 'parseentities.py', 'patchcheck.py', 'pathfix.py', 'pdeps.py',
 'pickle2db.py', 'pindent.py', 'ptags.py', 'pydocgui.pyw', 'pysource.py', 'RE
ADME.txt', 'redemo.py', 'reindent-rst.py', 'reindent.py', 'rgrep.py', 'serve.
py', 'setup.py', 'suff.py', 'svneol.py', 'texcheck.py', 'texi2html.py', 'tree
sync.py', 'untabify.py', 'which.py', 'win_add2path.py', 'xxci.py']
--------------------------------------------------
Current Path: C:\Python27\Tools\versioncheck
Directories: []
Files: ['checkversions.py', 'pyversioncheck.py', 'README.txt', '_checkversio
n.py']
--------------------------------------------------
Current Path: C:\Python27\Tools\webchecker
Directories: []
Files: ['README.txt', 'tktools.py', 'wcgui.py', 'wcmac.py', 'webchecker.py',
 'websucker.py', 'wsgui.py']
--------------------------------------------------
```

Assignment 1: exceute this, and observe the output

`

```
for i in os.environ:  # To get the environmental variables
    print i
```

In [35]:
```
print "os.environ.get('TMP')", os.environ.get('TMP')
```
```
os.environ.get('TMP') C:\Users\UPETHA~1\AppData\Local\Temp
```

In [36]:
```
l = os.environ
```

In [37]:
```
type(l)
```
Out[37]:
```
instance
```

In [41]:
```
filePath = os.path.join(os.environ.get('TMP'), 'test.txt')
print filePath
```
```
C:\Users\UPETHA~1\AppData\Local\Temp\test.txt
```

In [42]:
```
print os.path.exists(filePath)
```
```
False
```

In [43]:
```
os.getcwd()
```
Out[43]:
```
'C:\\Users\\upethakamsetty\\Google Drive\\python\\tut\\kt Group Batch 2\\clas
s_10_18-August-2016'
```

In [44]:
```
os.listdir(os.getcwd())
```
Out[44]:
```
['.ipynb_checkpoints',
 'class_10_18-August-2016.ipynb',
 'first',
 'myNewFile.txt',
 'newFolder']
```

In [46]:
```
os.path.basename('C:\\Users\\upethakamsetty\\Google Drive\\python\\tut\\kt Gro
up Batch 2\\class_10_18-August-2016\\myNewFile.txt')
```
Out[46]:
```
'myNewFile.txt'
```

In [47]:
```
os.path.dirname('C:\\Users\\upethakamsetty\\Google Drive\\python\\tut\\kt Grou
p Batch 2\\class_10_18-August-2016\\myNewFile.txt')
```
Out[47]:
```
'C:\\Users\\upethakamsetty\\Google Drive\\python\\tut\\kt Group Batch 2\\clas
s_10_18-August-2016'
```

```
In [48]: os.path.splitext('C:\\Users\\upethakamsetty\\Google Drive\\python\\tut\\kt Gro
         up Batch 2\\class_10_18-August-2016\\myNewFile.txt')

Out[48]: ('C:\\Users\\upethakamsetty\\Google Drive\\python\\tut\\kt Group Batch 2\\cla
         ss_10_18-August-2016\\myNewFile',
          '.txt')
```

```
In [49]: os.path.splitext('myNewFile.txt')

Out[49]: ('myNewFile', '.txt')
```

```
In [50]: os.listdir(os.getcwd())

Out[50]: ['.ipynb_checkpoints',
          'class_10_18-August-2016.ipynb',
          'first',
          'myNewFile.txt',
          'newFolder']
```

```
In [51]: os.listdir('.')

Out[51]: ['.ipynb_checkpoints',
          'class_10_18-August-2016.ipynb',
          'first',
          'myNewFile.txt',
          'newFolder']
```

```
In [53]: os.chdir('newFolder/')
         os.listdir('..')

Out[53]: ['.ipynb_checkpoints',
          'class_10_18-August-2016.ipynb',
          'first',
          'myNewFile.txt',
          'newFolder']
```

```
In [56]: os.chdir('..')
```

```
In [57]: os.rmdir('newFolder/')
```

```
In [58]: os.listdir('.')

Out[58]: ['.ipynb_checkpoints',
          'class_10_18-August-2016.ipynb',
          'first',
          'myNewFile.txt']
```

time related modules - time, datetime, pytz, ...

```
In [1]: import time
```

In [2]:
```
print time.__doc__
```

```
This module provides various functions to manipulate time values.

There are two standard representations of time.  One is the number
of seconds since the Epoch, in UTC (a.k.a. GMT).  It may be an integer
or a floating point number (to represent fractions of seconds).
The Epoch is system-defined; on Unix, it is generally January 1st, 1970.
The actual value can be retrieved by calling gmtime(0).

The other representation is a tuple of 9 integers giving local time.
The tuple items are:
  year (four digits, e.g. 1998)
  month (1-12)
  day (1-31)
  hours (0-23)
  minutes (0-59)
  seconds (0-59)
  weekday (0-6, Monday is 0)
  Julian day (day in the year, 1-366)
  DST (Daylight Savings Time) flag (-1, 0 or 1)
If the DST flag is 0, the time is given in the regular time zone;
if it is 1, the time is given in the DST time zone;
if it is -1, mktime() should guess based on the date and time.

Variables:

timezone -- difference in seconds between UTC and local standard time
altzone -- difference in  seconds between UTC and local DST time
daylight -- whether local time should reflect DST
tzname -- tuple of (standard time zone name, DST time zone name)

Functions:

time() -- return current time in seconds since the Epoch as a float
clock() -- return CPU time since process start as a float
sleep() -- delay for a number of seconds given as a float
gmtime() -- convert seconds since Epoch to UTC tuple
localtime() -- convert seconds since Epoch to local time tuple
asctime() -- convert time tuple to string
ctime() -- convert time in seconds to string
mktime() -- convert local time tuple to seconds since Epoch
strftime() -- convert time tuple to string according to format specification
strptime() -- parse string to time tuple according to format specification
tzset() -- change the local timezone
```

In [3]:
```
print dir(time)
```

```
['__doc__', '__name__', '__package__', 'accept2dyear', 'altzone', 'asctime',
 'clock', 'ctime', 'daylight', 'gmtime', 'localtime', 'mktime', 'sleep', 'str
ftime', 'strptime', 'struct_time', 'time', 'timezone', 'tzname']
```

In [4]:
```
time.tzname
```

Out[4]:
```
('India Standard Time', 'India Daylight Time')
```

```
In [5]: time.tzname[0]

Out[5]: 'India Standard Time'

In [6]: time.daylight # Results in boolean result of existence or absence of DST, in t
        hat time zone

Out[6]: 0

In [7]: time.timezone

Out[7]: -19800

In [8]: time.time()    #seconds past from epoch time, till now

Out[8]: 1471572115.805

In [9]: time.ctime()

Out[9]: 'Fri Aug 19 07:32:35 2016'

In [10]: time.asctime()

Out[10]: 'Fri Aug 19 07:33:01 2016'

In [11]: time.gmtime()

Out[11]: time.struct_time(tm_year=2016, tm_mon=8, tm_mday=19, tm_hour=2, tm_min=3, tm_
         sec=10, tm_wday=4, tm_yday=232, tm_isdst=0)

In [12]: time.localtime()

Out[12]: time.struct_time(tm_year=2016, tm_mon=8, tm_mday=19, tm_hour=7, tm_min=36, tm
         _sec=17, tm_wday=4, tm_yday=232, tm_isdst=0)

In [13]: t = time.localtime()
         print type(t)

         <type 'time.struct_time'>

In [14]: time.clock()

Out[14]: 8.924961600352715e-07

In [15]: time.sleep(6)  # To let the interpreter to sleep for 6 seconds

In [16]: time.strptime('Fri Aug 19 07:33:01 2016')

Out[16]: time.struct_time(tm_year=2016, tm_mon=8, tm_mday=19, tm_hour=7, tm_min=33, tm
         _sec=1, tm_wday=4, tm_yday=232, tm_isdst=-1)
```

```
In [17]: time.strptime(time.asctime())

Out[17]: time.struct_time(tm_year=2016, tm_mon=8, tm_mday=19, tm_hour=7, tm_min=44, tm
         _sec=47, tm_wday=4, tm_yday=232, tm_isdst=-1)

In [18]: time.strptime(time.ctime())

Out[18]: time.struct_time(tm_year=2016, tm_mon=8, tm_mday=19, tm_hour=7, tm_min=45, tm
         _sec=1, tm_wday=4, tm_yday=232, tm_isdst=-1)

In [21]: time.strptime("8/4/1988", "%d/%m/%Y")

Out[21]: time.struct_time(tm_year=1988, tm_mon=4, tm_mday=8, tm_hour=0, tm_min=0, tm_s
         ec=0, tm_wday=4, tm_yday=99, tm_isdst=-1)

In [22]: time.strptime("8/4/1988", "%m/%d/%Y")

Out[22]: time.struct_time(tm_year=1988, tm_mon=8, tm_mday=4, tm_hour=0, tm_min=0, tm_s
         ec=0, tm_wday=3, tm_yday=217, tm_isdst=-1)

In [23]: time.strptime("08 Apr 1988", "%d %b %Y")

Out[23]: time.struct_time(tm_year=1988, tm_mon=4, tm_mday=8, tm_hour=0, tm_min=0, tm_s
         ec=0, tm_wday=4, tm_yday=99, tm_isdst=-1)

In [25]: epochTime = time.time()
         print epochTime
         newCreatedTime = time.mktime(time.strptime(epochTime))
         print newCreatedTime

         1471573345.55

         ---------------------------------------------------------------------------
         TypeError                                 Traceback (most recent call last)
         <ipython-input-25-3bc0e7e6a05a> in <module>()
               1 epochTime = time.time()
               2 print epochTime
         ----> 3 newCreatedTime = time.mktime(time.strptime(epochTime))
               4 print newCreatedTime

         c:\python27\lib\_strptime.pyc in _strptime_time(data_string, format)
             476
             477 def _strptime_time(data_string, format="%a %b %d %H:%M:%S %Y"):
         --> 478     return _strptime(data_string, format)[0]

         c:\python27\lib\_strptime.pyc in _strptime(data_string, format)
             327                 raise ValueError("stray %% in format '%s'" % format)
             328             _regex_cache[format] = format_regex
         --> 329     found = format_regex.match(data_string)
             330     if not found:
             331         raise ValueError("time data %r does not match format %r" %

         TypeError: expected string or buffer

In [26]: import datetime
```

In [27]:
```
print datetime.__doc__
```

```
Fast implementation of the datetime type.
```

In [28]:
```
print dir(datetime)
```

```
['MAXYEAR', 'MINYEAR', '__doc__', '__name__', '__package__', 'date', 'datetim
e', 'datetime_CAPI', 'time', 'timedelta', 'tzinfo']
```

In [30]:
```
print datetime.datetime.__doc__
```

```
datetime(year, month, day[, hour[, minute[, second[, microsecond[,tzinf
o]]]]])

The year, month and day arguments are required. tzinfo may be None, or an
instance of a tzinfo subclass. The remaining arguments may be ints or longs.
```

In [31]:
```
print dir(datetime.datetime)
```

```
['__add__', '__class__', '__delattr__', '__doc__', '__eq__', '__format__', '_
_ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__le__', '__lt
__', '__ne__', '__new__', '__radd__', '__reduce__', '__reduce_ex__', '__repr_
_', '__rsub__', '__setattr__', '__sizeof__', '__str__', '__sub__', '__subclas
shook__', 'astimezone', 'combine', 'ctime', 'date', 'day', 'dst', 'fromordina
l', 'fromtimestamp', 'hour', 'isocalendar', 'isoformat', 'isoweekday', 'max',
 'microsecond', 'min', 'minute', 'month', 'now', 'replace', 'resolution', 'se
cond', 'strftime', 'strptime', 'time', 'timetuple', 'timetz', 'today', 'toord
inal', 'tzinfo', 'tzname', 'utcfromtimestamp', 'utcnow', 'utcoffset', 'utctim
etuple', 'weekday', 'year']
```

In [32]:
```
datetime.datetime.now()   # Local time
```

Out[32]:
```
datetime.datetime(2016, 8, 19, 7, 55, 48, 51000)
```

In [33]:
```
datetime.datetime.utcnow()
```

Out[33]:
```
datetime.datetime(2016, 8, 19, 2, 26, 33, 432000)
```

In [36]:
```
datetime.date.today()
```

Out[36]:
```
datetime.date(2016, 8, 19)
```

In [37]:
```
tdy = datetime.date.today()
```

In [38]:
```
print tdy.strftime("four-digit year: %Y, two-digit year: %y, month: %m, day: %
d")
```

```
four-digit year: 2016, two-digit year: 16, month: 08, day: 19
```

```
In [39]: print tdy.strftime("four-digit year: %Y, two-digit year: %y, month: %m, monthI
         nWords: %b, day: %d")
```

```
four-digit year: 2016, two-digit year: 16, month: 08, monthInWords: Aug, day:
 19
```

Both time and datetime modules can be used together

```
In [40]: t = datetime.datetime.now() # For the local timezone
         print t

         print "Epoch Seconds:", time.mktime(t.timetuple())
```

```
2016-08-19 08:02:52.502000
Epoch Seconds: 1471573972.0
```

```
In [41]: t = datetime.datetime.utcnow()   # For UTC
         print t
         print "Epoch Seconds:", time.mktime(t.timetuple())
```

```
2016-08-19 02:33:41.204000
Epoch Seconds: 1471554221.0
```

timeit moule

```
In [42]: import timeit
```

```
In [43]: logic = '[x for x in xrange(10) if x%2 != 0]'

         eval(logic)   # builtin function to execute a statement
```

```
Out[43]: [1, 3, 5, 7, 9]
```

```
In [44]: t = timeit.Timer(logic)
         print t
```

```
<timeit.Timer instance at 0x048205A8>
```

```
In [45]: print "1000 repeats of this logic takes :", t.timeit(1000), " seconds"
```

```
1000 repeats of this logic takes : 0.00263464866453  seconds
```

```
In [46]: print "10,00,000 repeats of this logic takes :", t.timeit(1000000), " seconds"
```

```
10,00,000 repeats of this logic takes : 4.16658887075  seconds
```

In [47]:
```
code = 'import random; l = random.sample(xrange(10000000), 1000); l.sort()'
eval(code)

t = timeit.Timer(code)

print "Create a list of a thousand random numbers. Sort the list. Repeated a t
housand times."
print "Average Time:", t.timeit(1000) / 1000
```

```
  File "<string>", line 1
    import random; l = random.sample(xrange(10000000), 1000); l.sort()
         ^
SyntaxError: invalid syntax
```

In [49]:
```
timeit.timeit('range(12)')
```

Out[49]:
```
1.2337965090905527
```

In [50]:
```
timeit.timeit('xrange(12)')
```

Out[50]:
```
0.6855097893439961
```

Python file types

.pyw - This is windows executable

.pyc - compiled python bytecode file, for a particular .py file.

.pyd - python dll file

.pyc file is platform-independent, yet interpreter dependent. The interpreter checks for the *.py* file last modified time stamp with that of *.pyc* file. If there is a mismatch, then that *.pyc* file will be discarded, and a new *.pyc* file will be created.

It is created either

```
1. when a particular _.py_ file is imported in another python script and/or in pyth
on interpreter.
2. Manually _.pyc_ file can be created, when the _.py_ file is compiled using py_co
mpile
    python -m py_compile fileName.py
```

In [51]:
```
import os; os.listdir(os.getcwd())
```
Out[51]:
```
['.ipynb_checkpoints',
 'class_10_18-August-2016.ipynb',
 'class_10_18-August-2016.pdf',
 'first',
 'myNewFile.txt',
 'newScript.py']
```

In [52]:
```
import newScript    #.py extension is not required
```

In [53]:
```
os.listdir(os.getcwd())    # observe the .pyc file
```
Out[53]:
```
['.ipynb_checkpoints',
 'class_10_18-August-2016.ipynb',
 'class_10_18-August-2016.pdf',
 'first',
 'myNewFile.txt',
 'newScript.py',
 'newScript.pyc']
```

In [54]:
```
print newScript.__doc__
```
```
        Purpose: module importing demonstration
```

In [55]:
```
print dir(newScript)
```
```
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'additio
n', 'firstFunction', 'multiplication', 'subtraction']
```

In [56]:
```
newScript.firstFunction()
```
```
This is first function
```

In [60]:
```
help(newScript.addition)
```
```
Help on function addition in module newScript:

addition(a, b)
    performs addition operation
```

In [61]:
```
newScript.addition(99, 67)
```
Out[61]:
```
166
```

In [62]:
```
newScript.addition(12.23, 6.07)
```
Out[62]:
```
18.3
```

```
In [63]: newScript.subtraction(12.23, 6.07)

Out[63]: 6.16
```

The newScript.py file was modified with better help for functions. ALso, two static variables were added.

```
In [64]: help(newScript.addition)

         Help on function addition in module newScript:

         addition(a, b)
             performs addition operation
```

Notice that the changes were not reflected.

Modifying the script of load modules needs reloading the module to get the changes to be affected in the working script, or interpreter.

```
In python 2.x,
    reload(<user-defined Module name>)
    ex: reload(newScript)
In python 3.x,
    import imp
    imp.reload(<user-defined Module name>)
    imp.reload(newScript)

    or

    import importlib
    importlib.reload(<user-defined Module name>)
    importlib.reload(newScript)
```

There are various other modules like xreload, reimport with additional functionalities, for reloading the modules.

```
In [66]: reload(newScript)   # To get the changes

Out[66]: <module 'newScript' from 'newScript.py'>

In [67]: help(newScript.addition)

         Help on function addition in module newScript:

         addition(a, b)
             performs addition operation
             ex: addition(12, 34)
             returns: a+b
```

```
In [68]: print dir(newScript)

         ['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'additio
         n', 'firstFunction', 'luckyNumber', 'multiplication', 'subtraction', 'vowel
         s']
```

```
In [69]: newScript.luckyNumber

Out[69]: 1321
```

To ensure that certain part of logic should be executed only when the script is independently called, write that logic under `if __name__ == '__main__'` condition

```
In [70]: reload(newScript)

         This script is imported from another module

Out[70]: <module 'newScript' from 'newScript.py'>
```

At times, if the imported module has any dependencies on other imported modules, those functionality will not get refreshed.

In that case, it would be better to delete that imported module object.

```
#del <importedModuleName>
del newScript

or
# using sys module
import sys
#del sys.modules[<importedModuleName>]
del sys.modules[newScript]
```

```
In [71]: del newScript   # deletes the imported object from the interpreter cache
```

```
In [72]: newScript # results in error, as that object is no more present in interpreter
          cache

         ---------------------------------------------------------------------------
         NameError                                 Traceback (most recent call last)
         <ipython-input-72-67d7c038759c> in <module>()
         ----> 1 newScript

         NameError: name 'newScript' is not defined
```

In [73]:
```
import newScript
print dir(newScript)
```

```
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'additio
n', 'firstFunction', 'luckyNumber', 'multiplication', 'subtraction', 'vowel
s']
```