Lecture 6 Class and Modules

A possibly overlooked point: Modules and Class in Python share many similaries at the basic level. They both define some names (attributes) and functions (methods) for the convenience of users -- and the codes to call them are also similar. Of course, Class also serves as the blue prints to generate instances, and supports more advanced functions such as Inheritance.

Class and Instance

Simple Example of Vector ¶

Let's first define the simplest class in Python

```
In [ ]: class VectorV0:
            '''The simplest class in python'''  # this is the document string
            
            pass
```

and create two instances `v1` and `v2`

```
In [ ]: v1 = VectorV0()  # note the parentheses here
        v2 = VectorV0()
```

Now `v1` and `v2` are the objects in Python

```
In [ ]: type(v1)
```

```
In [ ]: dir(v1)
```

We can manually assign the attributes to instance `v1` and `v2`

```
In [ ]: v1.x = 1.0
        v1.y = 2.0
        v2.x = 2.0
        v2.y = 3.0
```

```
In [ ]: dir(v1)
```

We don't want to create the instance or define the coordinates seperately. Can we do these in one step, when initializing the instance?

```
In [ ]: class VectorV1:
            '''define the vector'''  # this is the document string
            dim = 2   # this is the attribute
            def __init__(self, x=0.0, y=0.0):  # any method in Class requires the first param
        eter to be self!
                self.x = x
                self.y = y
```

```
In [ ]: v1 = VectorV1(1.0,2.0)
In [ ]: dir(v1)
```

```
In [ ]: print(v1.dim)
        print(v1.x)
        print(v1.y)
```

Btw, there is nothing mysterious about the `__init__` : you can just assume it is a function (method) stored in v1, and you can always call it if you like!

When you write `v1.__init__()` , you can equivalently think that you are calling a function with "ugly function name" `__init__` , and the parameter is `v1` (self), i.e. you are writing `__init__(v1)` . It is just a function updating the attributes of instance objects!

More generally, for the method `method(self, params)` you can call it by `self.method(params)` .

```
In [ ]: print(v1.x)
        print(id(v1))
        v1.__init__()
        print(v1.x)
        print(id(v1))
```

Another secret uncovered: `v1` is just a mutable object, and the "function" `__init__( )` just change `v1` in place!

Now we move on to update our vector class by defining more functions. Since you may not like ugly names here with dunder, let's just begin with normal function names.

```
In [ ]: class VectorV2:
            '''define the vector'''  # this is the document string
            dim = 2   # this is the attribute

            def __init__(self, x=0.0, y=0.0):  # any method in Class requires the first param
        eter to be self!
                '''initialize the vector by providing x and y coordinate'''
                self.x = x
                self.y = y

            def norm(self):
                '''calculate the norm of vector'''
                return math.sqrt(self.x**2+self.y**2)

            def vector_sum(self, other):
                '''calculate the vector sum of two vectors'''
                return VectorV2(self.x + other.x, self.y + other.y)

            def show_coordinate(self):
                '''display the coordinates of the vector'''
                return 'Vector(%r, %r)' % (self.x, self.y)
```

```
In [ ]: help(VectorV2)
```

```
In [ ]: import math
        v1 = VectorV2(1.0,2.0)
        v2 = VectorV2(2.0,3.0)
```

```
In [ ]: v1.norm()
```

```
In [ ]: v3 = v1.vector_sum(v2)
        v3.show_coordinate()
```

```
In [ ]: v1+v2 # will it work?
In [ ]: print(v3)
```

Something that we are still not satisfied:

- By typing v3 or using `print()` in the code, we cannot show its coordinates directly
- We cannot use the `+` operator to calculate the vector sum

Special (Magic) Methods

Here's the magic: by merely changing the function name, we can realize our goal!

```
In [ ]: class VectorV3:
            '''define the vector'''  # this is the document string
            dim = 2   # this is the attribute

            def __init__(self, x=0.0, y=0.0):  # any method in Class requires the first param
        eter to be self!
                '''initialize the vector by providing x and y coordinate'''
                self.x = x
                self.y = y

            def norm(self):
                '''calculate the norm of vector'''
                return math.sqrt(self.x**2+self.y**2)

            def __add__(self, other):
                '''calculate the vector sum of two vectors'''
                return VectorV3(self.x + other.x, self.y + other.y)

            def __repr__(self):   #special method of string representation
                '''display the coordinates of the vector'''
                return 'Vector(%r, %r)' % (self.x, self.y)
```

```
In [ ]: help(VectorV3)
```

```
In [ ]: v1 = VectorV3(1.0,2.0)
        v2 = VectorV3(2.0,3.0)
```

```
In [ ]: v3 = v1.__add__(v2)
        v3.__repr__()
```

```
In [ ]: v1 +v2
```

```
In [ ]: v3
```

Special methods are just like VIP admissions to take full use of the built-in operators in Python. With other special methods, you can even get elements by index `v3[0]` , or iterate through the object you created. For more advanced usage, you can see here (https://rszalski.github.io/magicmethods/).

Inheritance

Now we want to add another scalar production method to Vector, but we're tired of rewriting all the other methods. A good way is to create new Class VectorV4 (Child Class) by inheriting from VectorV3 (Parent Class) that we have already defined.

In [ ]:
```
class VectorV4(VectorV3): # Note the class VectorV3 in parentheses here
    '''define the vector'''  # this is the document string
    def __mul__(self, scalar):
        '''calculate the scalar product'''
        return VectorV4(self.x * scalar, self.y * scalar)
```

In [ ]:
```
help(VectorV4)
```

In [ ]:
```
v1 = VectorV4(1.0,2.0)
v2 = VectorV4(2.0,3.0)
```

In [ ]:
```
v1+v2
```

In [ ]:
```
v1*2
```

Modules and Packages

In Python, Functions (plus Classes, Variables) are contained in Modules, and Modules are organized in directories of Packages.

Now we have the `Vector.py` file in the folder.

In [2]:
```
import Vector
dir(Vector)
```

Out[2]:
```
['VectorV5',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'print_hello',
 'string']
```

In [5]:
```
Vector.string
```

Out[5]:
```
'Python'
```

In [6]:
```
Vector.print_hello()
```

```
Hello
```

In [7]:
```
v5 = Vector.VectorV5(1.0,2.0)
v5
```

In [9]:
```
import Vector as vc
vc.string
```

Out[9]:
```
'Python'
```

In [11]:
```
from Vector import print_hello # it's not a good habit to do this though, because of
 name conflicts
print_hello()
```

```
Hello
```

To import the modules, you must ensure that they are in your system paths.

In [13]:
```python
import sys
sys.path
```

Out[13]:
```
['/Users/cliffzhou/Documents/GitHub/UCI_MATH_10/lecture/lec_6',
 '/Users/cliffzhou/opt/anaconda3/lib/python37.zip',
 '/Users/cliffzhou/opt/anaconda3/lib/python3.7',
 '/Users/cliffzhou/opt/anaconda3/lib/python3.7/lib-dynload',
 '',
 '/Users/cliffzhou/opt/anaconda3/lib/python3.7/site-packages',
 '/Users/cliffzhou/opt/anaconda3/lib/python3.7/site-packages/aeosa',
 '/Users/cliffzhou/opt/anaconda3/lib/python3.7/site-packages/IPython/extensions',
 '/Users/cliffzhou/.ipython']
```

In [14]:
```
sys.modules.keys()
```

```
Out[14]: dict_keys(['sys', 'builtins', '_frozen_importlib', '_imp', '_thread', '_warnings',
         '_weakref', 'zipimport', '_frozen_importlib_external', '_io', 'marshal', 'posix', 'e
         ncodings', 'codecs', '_codecs', 'encodings.aliases', 'encodings.utf_8', '_signal',
         '__main__', 'encodings.latin_1', 'io', 'abc', '_abc', '_bootlocale', '_locale', 'sit
         e', 'os', 'stat', '_stat', 'posixpath', 'genericpath', 'os.path', '_collections_ab
         c', '_sitebuiltins', 'types', 'importlib', 'importlib._bootstrap', 'importlib._boots
         trap_external', 'warnings', 'importlib.util', 'importlib.abc', 'importlib.machiner
         y', 'contextlib', 'collections', 'operator', '_operator', 'keyword', 'heapq', '_heap
         q', 'itertools', 'reprlib', '_collections', 'functools', '_functools', 'mpl_toolkit
         s', 'sphinxcontrib', 'runpy', 'pkgutil', 'weakref', '_weakrefset', 'ipykernel', 'ipy
         kernel._version', 'ipykernel.connect', '__future__', 'json', 'json.decoder', 're',
         'enum', 'sre_compile', '_sre', 'sre_parse', 'sre_constants', 'copyreg', 'json.scanne
         r', '_json', 'json.encoder', 'subprocess', 'time', 'signal', 'errno', '_posixsubproc
         ess', 'select', 'selectors', 'collections.abc', 'math', 'threading', 'traceback', 'l
         inecache', 'tokenize', 'token', 'IPython', 'IPython.core', 'IPython.core.getipytho
         n', 'IPython.core.release', 'IPython.core.application', 'atexit', 'copy', 'glob', 'f
         nmatch', 'logging', 'string', '_string', 'shutil', 'zlib', 'bz2', '_compression', '_
         bz2', 'lzma', '_lzma', 'pwd', 'grp', 'traitlets', 'traitlets.traitlets', 'inspect',
         'dis', 'opcode', '_opcode', 'six', 'struct', '_struct', 'traitlets.utils', 'traitlet
         s.utils.getargspec', 'traitlets.utils.importstring', 'ipython_genutils', 'ipython_ge
         nutils._version', 'ipython_genutils.py3compat', 'ipython_genutils.encoding', 'local
         e', 'platform', 'traitlets.utils.sentinel', 'traitlets.utils.bunch', 'traitlets._ver
         sion', 'traitlets.config', 'traitlets.config.application', 'decorator', 'traitlets.c
         onfig.configurable', 'traitlets.config.loader', 'argparse', 'gettext', 'ast', '_as
         t', 'ipython_genutils.path', 'random', 'hashlib', '_hashlib', '_blake2', '_sha3', 'b
         isect', '_bisect', '_random', 'ipython_genutils.text', 'textwrap', 'ipython_genutil
         s.importstring', 'IPython.core.crashhandler', 'pprint', 'IPython.core.ultratb', 'pyd
         oc', 'urllib', 'urllib.parse', 'IPython.core.debugger', 'bdb', 'IPython.utils', 'IPy
         thon.utils.PyColorize', 'IPython.utils.coloransi', 'IPython.utils.ipstruct', 'IPytho
         n.utils.colorable', 'pygments', 'pygments.util', 'IPython.utils.py3compat', 'IPytho
         n.utils.encoding', 'IPython.core.excolors', 'IPython.testing', 'IPython.testing.skip
         doctest', 'pdb', 'cmd', 'code', 'codeop', 'IPython.core.display_trap', 'IPython.util
         s.path', 'IPython.utils.process', 'IPython.utils._process_posix', 'pexpect', 'pexpec
         t.exceptions', 'pexpect.utils', 'pexpect.expect', 'pexpect.pty_spawn', 'pty', 'tty',
         'termios', 'ptyprocess', 'ptyprocess.ptyprocess', 'fcntl', 'resource', 'ptyprocess.u
         til', 'pexpect.spawnbase', 'pexpect.run', 'IPython.utils._process_common', 'shlex',
         'IPython.utils.decorators', 'IPython.utils.data', 'IPython.utils.terminal', 'IPytho
         n.utils.sysinfo', 'IPython.utils._sysinfo', 'IPython.core.profiledir', 'IPython.path
         s', 'tempfile', 'IPython.utils.importstring', 'IPython.terminal', 'IPython.terminal.
         embed', 'IPython.core.compilerop', 'IPython.core.magic_arguments', 'IPython.core.err
         or', 'IPython.utils.text', 'pathlib', 'ntpath', 'IPython.core.magic', 'getopt', 'IPy
         thon.core.oinspect', 'typing', 'typing.io', 'typing.re', 'IPython.core.page', 'IPyth
         on.core.display', 'binascii', 'mimetypes', 'IPython.lib', 'IPython.lib.security', 'g
         etpass', 'IPython.lib.pretty', 'datetime', '_datetime', 'IPython.utils.openpy', 'IPy
         thon.utils.dir2', 'IPython.utils.wildcard', 'pygments.lexers', 'pygments.lexers._map
         ping', 'pygments.modeline', 'pygments.plugin', 'pygments.lexers.python', 'pygments.l
         exer', 'pygments.filter', 'pygments.filters', 'pygments.token', 'pygments.regexopt',
         'pygments.unistring', 'pygments.formatters', 'pygments.formatters._mapping', 'pygmen
         ts.formatters.html', 'pygments.formatter', 'pygments.styles', 'IPython.core.inputtra
         nsformer2', 'IPython.core.interactiveshell', 'pickleshare', 'pickle', '_compat_pickl
         e', '_pickle', 'IPython.core.prefilter', 'IPython.core.autocall', 'IPython.core.macr
         o', 'IPython.core.splitinput', 'IPython.core.alias', 'IPython.core.builtin_trap', 'I
         Python.core.events', 'backcall', 'backcall.backcall', 'IPython.core.displayhook', 'I
         Python.core.displaypub', 'IPython.core.extensions', 'IPython.core.formatters', 'IPyt
         hon.utils.sentinel', 'IPython.core.history', 'sqlite3', 'sqlite3.dbapi2', '_sqlite
         3', 'IPython.core.logger', 'IPython.core.payload', 'IPython.core.usage', 'IPython.di
         splay', 'IPython.lib.display', 'html', 'html.entities', 'IPython.utils.io', 'IPytho
         n.utils.capture', 'IPython.utils.strdispatch', 'IPython.core.hooks', 'IPython.utils.
         syspathcontext', 'IPython.utils.tempdir', 'IPython.utils.contexts', 'IPython.core.as
         ync_helpers', 'IPython.terminal.interactiveshell', 'asyncio', 'asyncio.base_events',
         'concurrent', 'concurrent.futures', 'concurrent.futures._base', 'socket', '_socket',
         'ssl', '_ssl', 'base64', 'asyncio.constants', 'asyncio.coroutines', 'asyncio.base_fu
         tures', 'asyncio.format_helpers', 'asyncio.log', 'asyncio.events', 'contextvars', '_
         contextvars', 'asyncio.base_tasks', '_asyncio', 'asyncio.futures', 'asyncio.protocol
         s', 'asyncio.sslproto', 'asyncio.transports', 'asyncio.tasks', 'asyncio.locks', 'asy
         ncio.runners', 'asyncio.queues', 'asyncio.streams', 'asyncio.subprocess', 'asyncio.u
         nix_events', 'asyncio.base_subprocess', 'asyncio.selector_events', 'prompt_toolkit',
         'prompt_toolkit.application', 'prompt_toolkit.application.application', 'prompt_tool
         kit.buffer', 'prompt_toolkit.application.current', 'prompt_toolkit.application.run_i
         n_terminal', 'prompt_toolkit.eventloop', 'prompt_toolkit.eventloop.async_generator',
'prompt_toolkit.eventloop.utils', 'prompt_toolkit.eventloop.inputhook', 'prompt_tool
kit.utils', 'wcwidth', 'wcwidth.wcwidth', 'wcwidth.table_wide', 'wcwidth.table_zer
o', 'prompt_toolkit.auto_suggest', 'prompt_toolkit.document', 'prompt_toolkit.clipbo
ard', 'prompt_toolkit.clipboard.base', 'prompt_toolkit.selection', 'prompt_toolkit.c
lipboard.in_memory', 'prompt_toolkit.filters', 'prompt_toolkit.filters.app', 'prompt
_toolkit.cache', 'prompt_toolkit.enums', 'prompt_toolkit.filters.base', 'prompt_tool
kit.filters.cli', 'prompt_toolkit.filters.utils', 'prompt_toolkit.completion', 'prom
pt_toolkit.completion.base', 'prompt_toolkit.formatted_text', 'prompt_toolkit.format
ted_text.ansi', 'prompt_toolkit.output', 'prompt_toolkit.output.base', 'prompt_toolk
it.data_structures', 'prompt_toolkit.styles', 'prompt_toolkit.styles.base', 'prompt_
toolkit.styles.defaults', 'prompt_toolkit.styles.named_colors', 'prompt_toolkit.styl
es.style', 'prompt_toolkit.styles.pygments', 'prompt_toolkit.styles.style_transforma
tion', 'colorsys', 'prompt_toolkit.output.color_depth', 'prompt_toolkit.output.defau
lts', 'prompt_toolkit.patch_stdout', 'prompt_toolkit.output.vt100', 'array', 'prompt
_toolkit.formatted_text.base', 'prompt_toolkit.mouse_events', 'prompt_toolkit.format
ted_text.html', 'xml', 'xml.dom', 'xml.dom.domreg', 'xml.dom.minidom', 'xml.dom.mini
compat', 'xml.dom.xmlbuilder', 'xml.dom.NodeFilter', 'prompt_toolkit.formatted_text.
pygments', 'prompt_toolkit.formatted_text.utils', 'prompt_toolkit.completion.filesys
tem', 'prompt_toolkit.completion.fuzzy_completer', 'prompt_toolkit.completion.word_c
ompleter', 'prompt_toolkit.completion.nested', 'prompt_toolkit.history', 'prompt_too
lkit.search', 'prompt_toolkit.key_binding', 'prompt_toolkit.key_binding.key_binding
s', 'prompt_toolkit.keys', 'prompt_toolkit.key_binding.key_processor', 'prompt_toolk
it.key_binding.vi_state', 'prompt_toolkit.validation', 'prompt_toolkit.input', 'prom
pt_toolkit.input.base', 'prompt_toolkit.input.defaults', 'prompt_toolkit.input.typea
head', 'prompt_toolkit.key_binding.bindings', 'prompt_toolkit.key_binding.bindings.p
age_navigation', 'prompt_toolkit.key_binding.bindings.scroll', 'prompt_toolkit.key_b
inding.defaults', 'prompt_toolkit.key_binding.bindings.basic', 'prompt_toolkit.key_b
inding.bindings.named_commands', 'prompt_toolkit.layout', 'prompt_toolkit.layout.con
tainers', 'prompt_toolkit.layout.controls', 'prompt_toolkit.lexers', 'prompt_toolki
t.lexers.base', 'prompt_toolkit.lexers.pygments', 'prompt_toolkit.layout.processor
s', 'prompt_toolkit.layout.utils', 'prompt_toolkit.layout.dimension', 'prompt_toolki
t.layout.margins', 'prompt_toolkit.layout.mouse_handlers', 'prompt_toolkit.layout.sc
reen', 'prompt_toolkit.layout.layout', 'prompt_toolkit.layout.menus', 'prompt_toolki
t.key_binding.bindings.completion', 'prompt_toolkit.key_binding.bindings.cpr', 'prom
pt_toolkit.key_binding.bindings.emacs', 'prompt_toolkit.key_binding.bindings.mouse',
'prompt_toolkit.key_binding.bindings.vi', 'prompt_toolkit.input.vt100_parser', 'prom
pt_toolkit.input.ansi_escape_sequences', 'prompt_toolkit.key_binding.digraphs', 'pro
mpt_toolkit.key_binding.emacs_state', 'prompt_toolkit.layout.dummy', 'prompt_toolki
t.renderer', 'prompt_toolkit.application.dummy', 'prompt_toolkit.shortcuts', 'prompt
_toolkit.shortcuts.dialogs', 'prompt_toolkit.key_binding.bindings.focus', 'prompt_to
olkit.widgets', 'prompt_toolkit.widgets.base', 'prompt_toolkit.widgets.toolbars', 'p
rompt_toolkit.widgets.dialogs', 'prompt_toolkit.widgets.menus', 'prompt_toolkit.shor
tcuts.progress_bar', 'prompt_toolkit.shortcuts.progress_bar.base', 'prompt_toolkit.s
hortcuts.progress_bar.formatters', 'prompt_toolkit.shortcuts.prompt', 'prompt_toolki
t.key_binding.bindings.auto_suggest', 'prompt_toolkit.key_binding.bindings.open_in_e
ditor', 'prompt_toolkit.shortcuts.utils', 'pygments.style', 'IPython.terminal.debugg
er', 'IPython.core.completer', 'unicodedata', 'IPython.core.latex_symbols', 'IPytho
n.utils.generics', 'jedi', 'jedi.api', 'parso', 'parso.parser', 'parso.tree', 'pars
o._compatibility', 'parso.utils', 'parso.pgen2', 'parso.pgen2.generator', 'parso.pge
n2.grammar_parser', 'parso.python', 'parso.python.tokenize', 'parso.python.token',
'parso.grammar', 'parso.python.diff', 'difflib', 'parso.python.parser', 'parso.pytho
n.tree', 'parso.python.prefix', 'parso.cache', 'gc', 'parso.python.errors', 'parso.n
ormalizer', 'parso.python.pep8', 'parso.file_io', 'jedi._compatibility', 'jedi.file_
io', 'queue', '_queue', 'jedi.parser_utils', 'jedi.debug', 'colorama', 'colorama.ini
tialise', 'colorama.ansitowin32', 'colorama.ansi', 'colorama.winterm', 'colorama.win
32', 'ctypes', '_ctypes', 'ctypes._endian', 'jedi.settings', 'jedi.cache', 'jedi.ap
i.classes', 'jedi.evaluate', 'jedi.evaluate.utils', 'jedi.evaluate.imports', 'jedi.e
valuate.sys_path', 'jedi.evaluate.cache', 'jedi.evaluate.base_context', 'jedi.commo
n', 'jedi.common.context', 'jedi.evaluate.helpers', 'jedi.common.utils', 'jedi.evalu
ate.compiled', 'jedi.evaluate.compiled.context', 'jedi.evaluate.filters', 'jedi.eval
uate.flow_analysis', 'jedi.evaluate.recursion', 'jedi.evaluate.names', 'jedi.evaluat
e.lazy_context', 'jedi.evaluate.compiled.access', 'jedi.evaluate.compiled.getattr_st
atic', 'jedi.evaluate.signature', 'jedi.evaluate.analysis', 'jedi.evaluate.gradual',
'jedi.evaluate.gradual.typeshed', 'jedi.evaluate.gradual.stub_context', 'jedi.evalua
te.context', 'jedi.evaluate.context.module', 'jedi.evaluate.context.klass', 'jedi.ev
aluate.arguments', 'jedi.evaluate.context.iterable', 'jedi.evaluate.param', 'jedi.ev
aluate.docstrings', 'jedi.evaluate.context.function', 'jedi.evaluate.parser_cache',
'jedi.evaluate.context.instance', 'jedi.evaluate.gradual.typing', 'jedi.evaluate.syn
tax_tree', 'jedi.evaluate.finder', 'jedi.evaluate.gradual.conversion', 'jedi.evaluat
e.gradual.annotation', 'jedi.api.keywords', 'pydoc_data', 'pydoc_data.topics', 'jed
```

```
i.api.interpreter', 'jedi.evaluate.compiled.mixed', 'jedi.api.helpers', 'jedi.api.co
mpletion', 'jedi.api.environment', 'filecmp', 'jedi.evaluate.compiled.subprocess',
'jedi.evaluate.compiled.subprocess.functions', 'jedi.api.exceptions', 'jedi.api.proj
ect', 'jedi.evaluate.usages', 'jedi.evaluate.gradual.utils', 'IPython.terminal.ptuti
ls', 'IPython.terminal.shortcuts', 'IPython.terminal.magics', 'IPython.lib.clipboar
d', 'IPython.terminal.pt_inputhooks', 'IPython.terminal.prompts', 'IPython.terminal.
ipapp', 'IPython.core.magics', 'IPython.core.magics.auto', 'IPython.core.magics.basi
c', 'IPython.core.magics.code', 'urllib.request', 'email', 'http', 'http.client', 'e
mail.parser', 'email.feedparser', 'email.errors', 'email._policybase', 'email.heade
r', 'email.quoprimime', 'email.base64mime', 'email.charset', 'email.encoders', 'quop
ri', 'email.utils', 'email._parseaddr', 'calendar', 'email.message', 'uu', 'email._e
ncoded_words', 'email.iterators', 'urllib.error', 'urllib.response', '_scproxy', 'IP
ython.core.magics.config', 'IPython.core.magics.display', 'IPython.core.magics.execu
tion', 'timeit', 'cProfile', '_lsprof', 'profile', 'pstats', 'IPython.utils.module_p
aths', 'IPython.utils.timing', 'IPython.core.magics.extension', 'IPython.core.magic
s.history', 'IPython.core.magics.logging', 'IPython.core.magics.namespace', 'IPytho
n.core.magics.osm', 'IPython.core.magics.packaging', 'IPython.core.magics.pylab', 'I
Python.core.pylabtools', 'IPython.core.magics.script', 'IPython.lib.backgroundjobs',
'IPython.core.shellapp', 'IPython.extensions', 'IPython.extensions.storemagic', 'IPy
thon.utils.frame', 'jupyter_client', 'jupyter_client._version', 'jupyter_client.conn
ect', 'zmq', 'zmq.backend', 'zmq.backend.select', 'zmq.backend.cython', 'zmq.backen
d.cython.constants', 'cython_runtime', 'zmq.backend.cython.error', '_cython_0_29_1
4', 'zmq.backend.cython.message', 'zmq.error', 'zmq.backend.cython.context', 'zmq.ba
ckend.cython.socket', 'zmq.backend.cython.utils', 'zmq.backend.cython._poll', 'zmq.b
ackend.cython._version', 'zmq.backend.cython._device', 'zmq.backend.cython._proxy_st
eerable', 'zmq.sugar', 'zmq.sugar.constants', 'zmq.utils', 'zmq.utils.constant_name
s', 'zmq.sugar.context', 'zmq.sugar.attrsettr', 'zmq.sugar.socket', 'zmq.sugar.pol
l', 'zmq.utils.jsonapi', 'zmq.utils.strtypes', 'zmq.sugar.frame', 'zmq.sugar.tracke
r', 'zmq.sugar.version', 'zmq.sugar.stopwatch', 'jupyter_client.localinterfaces', 'j
upyter_core', 'jupyter_core.version', 'jupyter_core.paths', 'jupyter_client.launche
r', 'traitlets.log', 'jupyter_client.client', 'jupyter_client.channels', 'jupyter_cl
ient.channelsabc', 'jupyter_client.clientabc', 'jupyter_client.manager', 'jupyter_cl
ient.kernelspec', 'jupyter_client.managerabc', 'jupyter_client.blocking', 'jupyter_c
lient.blocking.client', 'jupyter_client.blocking.channels', 'jupyter_client.multiker
nelmanager', 'uuid', '_uuid', 'ipykernel.kernelapp', 'tornado', 'tornado.ioloop', 'n
umbers', 'tornado.concurrent', 'tornado.log', 'logging.handlers', 'tornado.escape',
'tornado.util', 'tornado.speedups', 'curses', '_curses', 'zmq.eventloop', 'zmq.event
loop.ioloop', 'tornado.platform', 'tornado.platform.asyncio', 'tornado.gen', 'zmq.ev
entloop.zmqstream', 'ipykernel.iostream', 'imp', 'jupyter_client.session', 'hmac',
'jupyter_client.jsonutil', 'dateutil', 'dateutil._version', 'dateutil.parser', 'date
util.parser._parser', 'decimal', '_decimal', 'dateutil.relativedelta', 'dateutil._co
mmon', 'dateutil.tz', 'dateutil.tz.tz', 'six.moves', 'dateutil.tz._common', 'dateuti
l.tz._factories', 'dateutil.parser.isoparser', '_strptime', 'jupyter_client.adapte
r', 'ipykernel.heartbeat', 'ipykernel.ipkernel', 'IPython.utils.tokenutil', 'ipykern
el.comm', 'ipykernel.comm.manager', 'ipykernel.comm.comm', 'ipykernel.kernelbase',
'tornado.queues', 'tornado.locks', 'ipykernel.jsonutil', 'ipykernel.zmqshell', 'IPyt
hon.core.payloadpage', 'ipykernel.displayhook', 'ipykernel.eventloops', 'distutils',
'distutils.version', 'ipykernel.parentpoller', 'faulthandler', 'ipykernel.datapub',
'ipykernel.serialize', 'ipykernel.pickleutil', 'ipykernel.codeutil', 'IPython.core.c
ompleterlib', 'plistlib', 'xml.parsers', 'xml.parsers.expat', 'pyexpat.errors', 'pye
xpat.model', 'pyexpat', 'xml.parsers.expat.model', 'xml.parsers.expat.errors', 'appn
ope', 'appnope._nope', 'ctypes.util', 'ctypes.macholib', 'ctypes.macholib.dyld', 'ct
ypes.macholib.framework', 'ctypes.macholib.dylib', 'storemagic', 'Vector'])
```

We can import the `inspect` package and use `getsource` method to see the source codes of imported modules. Note that this does not work for built-in functions (https://github.com/python/cpython).

In [25]:
```
import inspect
lines = inspect.getsource(Vector.VectorV5)
print(lines)
```

```
class VectorV5:
    '''define the vector'''  # this is the document string
    dim = 2   # this is the attribute

    def __init__(self, x=0.0, y=0.0):  # any method in Class requires the first para
meter to be self!
        '''initialize the vector by providing x and y coordinate'''
        self.x = x
        self.y = y

    def norm(self):
        '''calculate the norm of vector'''
        return math.sqrt(self.x**2+self.y**2)

    def __add__(self, other):
        '''calculate the vector sum of two vectors'''
        return VectorV5(self.x + other.x, self.y + other.y)

    def __repr__(self):   #special method of string representation
        '''display the coordinates of the vector'''
        return 'Vector(%r, %r)' % (self.x, self.y)

    def __mul__(self, scalar):
        '''calculate the scalar product'''
        return VectorV5(self.x * scalar, self.y * scalar)
```

If we are interested in `numpy` ... (in fact `numpy` is a package rather than modules)

In [34]:
```
import numpy as np
lines = inspect.getsource(np.sum)
print(lines)
```

```
@array_function_dispatch(_sum_dispatcher)
def sum(a, axis=None, dtype=None, out=None, keepdims=np._NoValue,
        initial=np._NoValue, where=np._NoValue):
    """
    Sum of array elements over a given axis.

    Parameters
    ----------
    a : array_like
        Elements to sum.
    axis : None or int or tuple of ints, optional
        Axis or axes along which a sum is performed.  The default,
        axis=None, will sum all of the elements of the input array.  If
        axis is negative it counts from the last to the first axis.

        .. versionadded:: 1.7.0

        If axis is a tuple of ints, a sum is performed on all of the axes
        specified in the tuple instead of a single axis or all the axes as
        before.
    dtype : dtype, optional
        The type of the returned array and of the accumulator in which the
        elements are summed.  The dtype of `a` is used by default unless `a`
        has an integer dtype of less precision than the default platform
        integer.  In that case, if `a` is signed then the platform integer
        is used while if `a` is unsigned then an unsigned integer of the
        same precision as the platform integer is used.
    out : ndarray, optional
        Alternative output array in which to place the result. It must have
        the same shape as the expected output, but the type of the output
        values will be cast if necessary.
    keepdims : bool, optional
        If this is set to True, the axes which are reduced are left
        in the result as dimensions with size one. With this option,
        the result will broadcast correctly against the input array.

        If the default value is passed, then `keepdims` will not be
        passed through to the `sum` method of sub-classes of
        `ndarray`, however any non-default value will be.  If the
        sub-class' method does not implement `keepdims` any
        exceptions will be raised.
    initial : scalar, optional
        Starting value for the sum. See `~numpy.ufunc.reduce` for details.

        .. versionadded:: 1.15.0

    where : array_like of bool, optional
        Elements to include in the sum. See `~numpy.ufunc.reduce` for details.

        .. versionadded:: 1.17.0

    Returns
    -------
    sum_along_axis : ndarray
        An array with the same shape as `a`, with the specified
        axis removed.   If `a` is a 0-d array, or if `axis` is None, a scalar
        is returned.  If an output array is specified, a reference to
        `out` is returned.

    See Also
    --------
    ndarray.sum : Equivalent method.

    add.reduce : Equivalent functionality of `add`.

    cumsum : Cumulative sum of array elements.

    trapz : Integration of array values using the composite trapezoidal rule.

    mean, average
    Notes
    -----
    Arithmetic is modular when using integer types, and no error is
    raised on overflow.

    The sum of an empty array is the neutral element 0:

    >>> np.sum([])
    0.0

    For floating point numbers the numerical precision of sum (and
    ``np.add.reduce``) is in general limited by directly adding each number
    individually to the result causing rounding errors in every step.
    However, often numpy will use a  numerically better approach (partial
    pairwise summation) leading to improved precision in many use-cases.
    This improved precision is always provided when no ``axis`` is given.
    When ``axis`` is given, it will depend on which axis is summed.
    Technically, to provide the best speed possible, the improved precision
    is only used when the summation is along the fast axis in memory.
    Note that the exact precision may vary depending on other parameters.
    In contrast to NumPy, Python's ``math.fsum`` function uses a slower but
    more precise approach to summation.
    Especially when summing a large number of lower precision floating point
    numbers, such as ``float32``, numerical errors can become significant.
    In such cases it can be advisable to use `dtype="float64"` to use a higher
    precision for the output.

    Examples
    --------
    >>> np.sum([0.5, 1.5])
    2.0
    >>> np.sum([0.5, 0.7, 0.2, 1.5], dtype=np.int32)
    1
    >>> np.sum([[0, 1], [0, 5]])
    6
    >>> np.sum([[0, 1], [0, 5]], axis=0)
    array([0, 6])
    >>> np.sum([[0, 1], [0, 5]], axis=1)
    array([1, 5])
    >>> np.sum([[0, 1], [np.nan, 5]], where=[False, True], axis=1)
    array([1., 5.])

    If the accumulator is too small, overflow occurs:

    >>> np.ones(128, dtype=np.int8).sum(dtype=np.int8)
    -128

    You can also start the sum with a value other than zero:

    >>> np.sum([10], initial=5)
    15
    """
    if isinstance(a, _gentype):
        # 2018-02-25, 1.15.0
        warnings.warn(
            "Calling np.sum(generator) is deprecated, and in the future will give a
different result. "
            "Use np.sum(np.fromiter(generator)) or the python sum builtin instead.",
            DeprecationWarning, stacklevel=3)

        res = _sum_(a)
        if out is not None:
            out[...] = res
            return out
        return res

    return _wrapreduction(a, np.add, 'sum', axis, dtype, out, keepdims=keepdims,
                          initial=initial, where=where)
```

In [43]:
```python
lines = inspect.getsource(np.core.fromnumeric._wrapreduction)
print(lines)
```

```
def _wrapreduction(obj, ufunc, method, axis, dtype, out, **kwargs):
    passkwargs = {k: v for k, v in kwargs.items()
                  if v is not np._NoValue}

    if type(obj) is not mu.ndarray:
        try:
            reduction = getattr(obj, method)
        except AttributeError:
            pass
        else:
            # This branch is needed for reductions like any which don't
            # support a dtype.
            if dtype is not None:
                return reduction(axis=axis, dtype=dtype, out=out, **passkwargs)
            else:
                return reduction(axis=axis, out=out, **passkwargs)

    return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
```

You can view all the source code of numpy (https://github.com/numpy/numpy) on Github.

Beyond Basic Python: What's next?

- Knowledge and wisdom
- What we have not covered in basic python: other data types (dictionary, set, tuple), input/output, exceptions, -- consult a byte of python (https://python.swaroopch.com/), or programiz (https://www.programiz.com/python-programming)
- The systematic book (for example,Python Cookbook (https://www.oreilly.com/library/view/python-cookbook-3rd/9781449357337/))
- Practice!Practice!Practive! Useful websites such as Leetcode (https://leetcode.com/)