modules & packages

In [4]:

```
import sys
print(dir(sys))
```

```
['__breakpointhook__', '__displayhook__', '__doc__', '__excepthook__', '__interactivehook
__', '__loader__', '__name__', '__package__', '__spec__', '__stderr__', '__stdin__', '__s
tdout__', '__unraisablehook__', '_base_executable', '_clear_type_cache', '_current_frames
', '_debugmallocstats', '_framework', '_getframe', '_git', '_home', '_xoptions', 'abiflag
s', 'addaudithook', 'api_version', 'argv', 'audit', 'base_exec_prefix', 'base_prefix', 'b
reakpointhook', 'builtin_module_names', 'byteorder', 'call_tracing', 'callstats', 'copyri
ght', 'displayhook', 'dont_write_bytecode', 'exc_info', 'excepthook', 'exec_prefix', 'exe
cutable', 'exit', 'flags', 'float_info', 'float_repr_style', 'get_asyncgen_hooks', 'get_c
oroutine_origin_tracking_depth', 'getallocatedblocks', 'getandroidapilevel', 'getcheckint
erval', 'getdefaultencoding', 'getdlopenflags', 'getfilesystemencodeerrors', 'getfilesyst
emencoding', 'getprofile', 'getrecursionlimit', 'getrefcount', 'getsizeof', 'getswitchint
erval', 'gettrace', 'hash_info', 'hexversion', 'implementation', 'int_info', 'intern', 'i
s_finalizing', 'last_traceback', 'last_type', 'last_value', 'maxsize', 'maxunicode', 'met
a_path', 'modules', 'path', 'path_hooks', 'path_importer_cache', 'platform', 'prefix', 'p
s1', 'ps2', 'ps3', 'pycache_prefix', 'set_asyncgen_hooks', 'set_coroutine_origin_tracking
_depth', 'setcheckinterval', 'setdlopenflags', 'setprofile', 'setrecursionlimit', 'setswi
tchinterval', 'settrace', 'stderr', 'stdin', 'stdout', 'thread_info', 'unraisablehook', '
version', 'version_info', 'warnoptions']
```

In [8]:

```
print(sys.version) #version of python
```

```
3.8.3 (default, May 27 2020, 02:08:17)
[GCC 9.3.0]
```

In [9]:

```
print(sys.version_info)
```

```
sys.version_info(major=3, minor=8, micro=3, releaselevel='final', serial=0)
```

In [15]:

```
print (sys.stderr)
print (dir(sys.stderr))
```

```
<ipykernel.iostream.OutStream object at 0x71b984c100>
['__abstractmethods__', '__class__', '__del__', '__delattr__', '__dict__', '__dir__', '__
doc__', '__enter__', '__eq__', '__exit__', '__format__', '__ge__', '__getattribute__', '_
_gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__lt__', '__m
odule__', '__ne__', '__new__', '__next__', '__reduce__', '__reduce_ex__', '__repr__', '__
setattr__', '__sizeof__', '__str__', '__subclasshook__', '_abc_impl', '_buffer', '_checkC
losed', '_checkReadable', '_checkSeekable', '_checkWritable', '_flush', '_flush_buffer',
'_flush_pending', '_io_loop', '_is_master_process', '_master_pid', '_new_buffer', '_sched
ule_flush', '_subprocess_flush_pending', 'close', 'closed', 'detach', 'echo', 'encoding',
'errors', 'fileno', 'flush', 'flush_interval', 'flush_timeout', 'isatty', 'name', 'newlin
es', 'parent_header', 'pub_thread', 'read', 'readable', 'readline', 'readlines', 'seek',
'seekable', 'session', 'set_parent', 'tell', 'topic', 'truncate', 'writable', 'write', 'w
ritelines']
```

In [16]:

```
print(sys.stdout)
print(dir(sys.stdout))
```

```
<ipykernel.iostream.OutStream object at 0x71b984c070>
['__abstractmethods__', '__class__', '__del__', '__delattr__', '__dict__', '__dir__', '__
doc__', '__enter__', '__eq__', '__exit__', '__format__', '__ge__', '__getattribute__', '_
_gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__lt__', '__m
odule__', '__ne__', '__new__', '__next__', '__reduce__', '__reduce_ex__', '__repr__', '
odule__', '__ne__', '__new__', '__next__', '__reduce__', '__reduce_ex__', '__repr__', '__
setattr__', '__sizeof__', '__str__', '__subclasshook__', '_abc_impl', '_buffer', '_checkC
losed', '_checkReadable', '_checkSeekable', '_checkWritable', '_flush', '_flush_buffer',
'_flush_pending', '_io_loop', '_is_master_process', '_master_pid', '_new_buffer', '_sched
ule_flush', '_subprocess_flush_pending', 'close', 'closed', 'detach', 'echo', 'encoding',
'errors', 'fileno', 'flush', 'flush_interval', 'flush_timeout', 'isatty', 'name', 'newlin
es', 'parent_header', 'pub_thread', 'read', 'readable', 'readline', 'readlines', 'seek',
'seekable', 'session', 'set_parent', 'softspace', 'tell', 'topic', 'truncate', 'writable'
, 'write', 'writelines']
```

In [17]:

```
print(sys.stdin)
print (dir (sys.stdin))
```

```
<_io.TextIOWrapper name='<stdin>' mode='r' encoding='utf-8'>
['_CHUNK_SIZE', '__class__', '__del__', '__delattr__', '__dict__', '__dir__', '__doc__',
'__enter__', '__eq__', '__exit__', '__format__', '__ge__', '__getattribute__', '__gt__',
'__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__lt__', '__ne__', '_
_new__', '__next__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof_
_', '__str__', '__subclasshook__', '_checkClosed', '_checkReadable', '_checkSeekable', '_
checkWritable', '_finalizing', 'buffer', 'close', 'closed', 'detach', 'encoding', 'errors
', 'fileno', 'flush', 'isatty', 'line_buffering', 'mode', 'name', 'newlines', 'read', 're
adable', 'readline', 'readlines', 'reconfigure', 'seek', 'seekable', 'tell', 'truncate',
'writable', 'write', 'write_through', 'writelines']
```

- **python file is created**
  - **go to jupyter home page**
  - **click on new**
  - **select on text file**
  - **rename text file as module.py**
  - **(.py) is extension of python**
  - **module is the name of python file**
  - **PYTHON FILE is created.....**

In [32]:

```
ls
```

```
Alarms/          Fonts/           Pictures/        iLovePDF/
Android/         JioSwitch/       Podcasts/        inShare/
Audiobooks/      Movies/          Subtitles/       module.py
ColorOS/         Music/           Untitled.ipynb   oplus_log/
DCIM/            Notifications/   VidMate/         python/
Documents/       PDF'S/           __pycache__/
Download/        PicsArt/         apssdc/
```

In [33]:

```
import module
print(dir(module))
```

```
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__packag
e__', '__spec__', 'add', 'mul', 'sub']
```

In [34]:

```
print (module.sub(100,10))
```

```
90
```

In [35]:

```
print (module.add(100,20))
```

```
120
```

In [36]:

```
print (module.mul(10,5))
```

```
50
```

--------O----------

continution of another topic

PYTHON OOP'S

new concept

- object oriented programming language
- our login will implement based on classes and objects
- it is used to develop an application base
- concepts here are....

1. class
    - collection of objects
    - logic will have some objects & methods
    - ex:-student--->sname,semail, srollno
2. object
    - object is real world entity that has state and behaviour
3. method
    - method is a function
4. constructor
    - it is special function to interact with object
    - we can create a constructor by using---> *init*()
    - 3 types but mainly 2 are helpfull
5. inherritance
    - it will aquires all the parent class attributes in child class

1. CLASS

syntax of class

- class classname:
    - statement1
    - ....
    - statementn

2.OBJECT (object creation)

- objectname=classname

- def functionname():
    - statements

3.METHOD

- class classname:
    - def methodname():# defining a method

4.CONSTRUCTOR

- class classname:

- class classname:
    - def *init*():

5.INHERITANCE

- **class classname:**
    - **statements**
- **class classname1(classname):**
    - **statements**

In [12]:

```
# class
class student: #class
    srollno=542
    sname="harsha"
    sbranch="cse"

    def show(self): #method
        print("roll no",self.srollno)
        print("name",self.sname)
        print("branch",self.sbranch)
```

In [13]:

```
s1=student()
```

In [14]:

```
s1.show()
```

```
roll no 542
name harsha
branch cse
```

In [15]:

```
print(s1.sname)
```

```
harsha
```

In [16]:

```
print(s1.srollno)
```

```
542
```

In [17]:

```
print(s1.sbranch)
```

```
cse
```

constructor---> special method (function)

init()

```
1. parameterized-->a conductor with parameters
2. non parameterized--> conductors without parameters
```

In [18]:

```
# bASIC
class student: #class
    def __init__ (self,name,roll,branch):
        self.name=name
        self.roll=roll
        self.branch=branch
    def display(self): #method
        print("roll no",self.roll)
        print("name",self.name)
        print("branch",self.branch)
```

In [19]:

```
st=student("Harsha",542,"Cse")
```

In [20]:

```
st.display()
```

```
roll no 542
name Harsha
branch Cse
```

In [21]:

```
# parameterized
class A:
    def __init__(self,a,b):
        print("wel to parameterized constructor")
        self.a=a
        self.b=b
    def add(self):
        print("result is:",self.a+self.b)
```

In [22]:

```
a1=A(5,7)
```

```
wel to parameterized constructor
```

In [23]:

```
a1.add()
```

```
result is: 12
```

In [24]:

```
a2 = A("ap","python")
```

```
wel to parameterized constructor
```

In [25]:

```
a2.add()
```

```
result is: appython
```

In [26]:

```
# non parameterized
class B:
    def __init__(self):
        print("welcome to non parameterized")
    def mul(self,n,n1):
        print("mul is:",n*n1)
```

In [27]:

```
b=B()
```

```
welcome to non parameterized
```

In [28]:

```
b.mul(5,6)
```

```
mul is: 30
```

In [29]:

```
b.mul(3,45)
```

```
mul is: 135
```

In [30]:

```
# multiple constructors
class multiple:
    def __init__ (self):
        print("first cons")
    def __init__ (self):
        print("second cons")
```

In [31]:

```
m=multiple()
```

```
second cons
```

python built_in class functions

- **1.getattr()-->for getting attribute value from class**
    - **we have to give 2 parameters**
        - **1.objectname**
        - **2.attributename**
- **2.setattr()-->for assigning new value to attribute**
    - **we have 3 parameters**
        - **1.objectname**
        - **2.attribute**
        - **3.new value**
- **3.delattr()-->we can delete attribute from the class**
    - **we have 2 arguments**
        - **objectname**
        - **attributename**
- **4.hasattr()-->it returns TRUE if attribute is existed**
    - **2 arguments**
        - **objectname**
        - **attributename**

In [60]:

```
class college:
    def __init__ (self,c_name,c_code,c_loc):
        self.c_name=c_name
        self.c_code=c_code
        self.c_loc=c_loc
```

In [61]:

```
c=college("kits",1,"guntur")
```

In [62]:

```
print(getattr(c,"c_name"))
print(getattr(c,"c_loc"))
```

```
kits
guntur
```

In [63]:

```
setattr(c,"c_name","kkr & ksr")
```

In [64]:

```
print(getattr(c,"c_name"))
print(getattr(c,"c_loc"))
```

```
kkr & ksr
guntur
```

In [65]:

```
print(getattr(c,"c_code"))
```

```
1
```

In [66]:

```
delattr(c,"c_code")
# c_code is deleted so can be displayed if we use gerattr
```

In [71]:

```
print(hasattr(c,"c_loc"))# it is there so true
```

```
True
```

In [72]:

```
print(hasattr(c,"c_code"))# it is deleted so false
```

```
False
```

In [ ]:

```

```