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:**
    - **def *init*():**

- 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")
```

```
[illegible]
```

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
```

- **we are creating object for class**
- **object**
- **real world entity**
  - **state**
  - **behaviour**

5.INHERITANTS

- **IT IS ALSO A CLASS**
- **PARENT CLASS & CHILD CLASS**
- **IT IS REUSEABLE**
- **WE CAN MODIFY WITHOUT CLASS**
- **SUPER CLASSES & SUB CLASSES ADVANTAGES**
- **TIME IS LESS**
- **LESS MEMORY**
- **EXECUTION TIME IS LESS**

In [2]:

```
class superclass:
    def a():
        print ("i'm superclass")
class derivedclass(superclass):
    def b():
        print("i'm derivedclass")
```

In [3]:

```
obj_super=superclass
```

In [4]:

```
obj_super.a()
```

```
i'm superclass
```

In [5]:

```
#obj_super.b()
# gives error
```

In [7]:

```
obj_derived=derivedclass
obj_derived.b()
```

```
i'm derivedclass
```

In [8]:

```
obj_derived.a()
```

```
i'm superclass
```

In [9]:

```
class parentclass:
    var="initial"
    def base():
        print("i'm base class")
class childclass(parentclass):
    v=1000
    def child():
        print ("im child of parent")
```

In [14]:

```
obj_parent=parentclass
obj_parent.var
```

Out[14]:

```
'initial'
```

In [19]:

```
obj_parent.base()
```

```
i'm base class
```

In [21]:

```
#obj_parent.v
# gives error
```

In [12]:

```
obj_child=childclass
obj_child.v
```

Out[12]:

```
1000
```

In [17]:

```
obj_child.child()
```

```
im child of parent
```

In [22]:

```
obj_child.var
```

Out[22]:

```
'initial'
```

```
initial
```

In [24]:

```
obj_child.base()
```

```
i'm base class
```

TYPES OF INHERITANCE

1.SINGLE LEVEL INHERITENCE 2.MULTIPLE INHERITENCE 3.MULTI LEVEL INHERITENCE

SINGLE LEVEL

- **ONLY ONE CHILD WHICH IS INHERIT ONLY FROM ONE PARENT**

In [25]:

```
class super:
    def __init__(self):
        self.n1=100
        self.n2=200
        print ("im init constructor")
    def show(self):
        print(f"n1 and n2 are {self.n1,self.n2}")
class derived(super):
    def add(self):
        return "addition of 2 no's",self.n1+self.n2
obj_s=super()
```

```
im init constructor
```

In [26]:

```
obj_s.show()
```

```
n1 and n2 are (100, 200)
```

In [28]:

```
obj_c=derived()
```

```
im init constructor
```

In [29]:

```
obj_c.show()
```

```
n1 and n2 are (100, 200)
```

In [30]:

```
obj_c.add()
```

Out[30]:

```
("addition of 2 no's", 300)
```

MULTIPLE INHERI

- **MORE THAN 1 PARENT BUT ONLY 1 CHILD**

In [31]:

```
class parent1:
    def property1():
        print ("im property of parent1")
class parent2:
    def property2():
        print("im property of parent2")
class child(parent1,parent2):
    def property3():
        print ("im property of child class")
```

In [34]:

```
obj_p1=parent1
```

In [36]:

```
obj_p1.property1()#can
#obj_p1.property2()#cant
#obj_p1.property3()#cant
```

```
im property of parent1
```

In [33]:

```
obj_p2=parent2
```

In [40]:

```
obj_p2.property2()#can
#obj_p2.property1()#cant
#obj_p2.property3()#cant
```

```
im property of parent2
```

In [41]:

```
obj_c=child
```

In [43]:

```
obj_c.property1()
obj_c.property2()
obj_c.property3()
```

```
im property of parent1
im property of parent2
im property of child class
```

In [46]:

```
class father:
    def __init__(self):
        self.amount=500
class mother:
    def __init__(self):
        self.money=1000
class you(father,mother):
    def __init__(self):
        father.__init__(self)
        mother.__init__(self)
        print("my monthly pocket money is",self.amount+self.money)
```

In [48]:

```
obj_you=you()
```

```
my monthly pocket money is 1500
```

ASSIGNMENT

- **USE MULTIPLE INHERITENCE**
- **GET 3 DIGIT OTP FROM MAIL CLASS**
- **GET 3 DIGIT OTP FROM PHONE CLASS**

- **BY USING VERIFIED CLASS(CHILD)**
- **ACCESS BOTH 3 DIGIT OTP'S AND RETURN 6 DIGIT**

In [52]:

```
import random
class mail:
    def __init__(self):
        self.otp1=random.randint(100,999)  #int(input())
class phone:
    def __init__(self):
        self.otp2=random.randint(100,999)
class verified(mail,phone):
    def __init__(self):
        mail.__init__(self)
        phone.__init__(self)
        print("6 digit otp",str(self.otp1)+str(self.otp2))
```

In [53]:

```
v=verified()
```

```
6 digit otp 375467
```

MULTI LEVEL INHERITENCE

- **ONE GRAND PARENT,PARENT CLASS AND ONLY ONE CHILD.....**

In [58]:

```
class grandfather:
    def __init__(self):
        self.property1="1 acer land"
        print("grandfather property",self.property1)
class father(grandfather):
    def __init__(self):
        grandfather.__init__(self)
        self.property2="flat"
        print("father property",self.property2,"and",self.property1)
class child(father):
    def __init__(self):
        father.__init__(self)
        self.property3="study"
        print("my property",self.property1,self.property2,self.property3)
obj=child()
```

```
grandfather property 1 acer land
father property flat and 1 acer land
my property 1 acer land flat study
```

In [61]:

```
obj_father=father()
```

```
grandfather property 1 acer land
father property flat and 1 acer land
```

In [63]:

```
obj_grand=grandfather()
```

```
grandfather property 1 acer land
```

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

continuous of previous topic

modules & packages

modules & packages

In [65]:

```
import sys
print(dir(sys)) #system modules
```

```
['__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 [67]:

```
print (sys.version)
```

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

In [69]:

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

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

In [71]:

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

```
<ipykernel.iostream.OutStream object at 0x7451022100>
```

In [73]:

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

```
['__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 [75]:

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

```
['__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', 'softspace', 'tell', 'topic', 'truncate', 'writable'
, 'write', 'writelines']
```

In [77]:

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

```
['_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']
```

In [79]:

```
import math as mt
print(dir(mt))
```

```
['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'aco
sh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'comb', 'copysign', 'cos', 'cosh'
, 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'f
mod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'i
snan', 'isqrt', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'perm'
, 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau
', 'trunc']
```

In [81]:

```
print(mt.tan(45))
```

```
1.6197751905438615
```

In [84]:

```
print(mt.factorial(6))
```

```
720
```

In [86]:

```
print(mt.sin(45))
```

```
0.8509035245341184
```

In [88]:

```
a=mt.floor(9.25)
b=mt.floor(9.5)
c=mt.ceil(9.25)
d=mt.ceil(9.25)
print(a,b,c,d)
```

```
9 9 10 10
```

In [89]:

```
print(mt.gcd(120,180))
```

```
60
```

In [91]:

```
print(mt.sqrt(36))
```

```
6.0
```

In [93]:

```
print(mt.pow(2,3))
```

```
8.0
```

In [96]:

```
print(mt.pi)
```

```
3.141592653589793
```

In [97]:

```
# random module
import random as rd
print(dir(rd))
```

```
['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random', 'SG_MAGICCONST', 'SystemRandom',
'TWOPI', '_Sequence', '_Set', '__all__', '__builtins__', '__cached__', '__doc__', '__file
__', '__loader__', '__name__', '__package__', '__spec__', '_accumulate', '_acos', '_bisec
t', '_ceil', '_cos', '_e', '_exp', '_inst', '_log', '_os', '_pi', '_random', '_repeat', '
_sha512', '_sin', '_sqrt', '_test', '_test_generator', '_urandom', '_warn', 'betavariate'
, 'choice', 'choices', 'expovariate', 'gammavariate', 'gauss', 'getrandbits', 'getstate',
'lognormvariate', 'normalvariate', 'paretovariate', 'randint', 'random', 'randrange', 'sa
mple', 'seed', 'setstate', 'shuffle', 'triangular', 'uniform', 'vonmisesvariate', 'weibul
lvariate']
```

In [99]:

```
k=random.randint(1,100)
print(k)
```

```
69
```

In [101]:

```
nt=random.random()
print(nt)
```

```
0.9250486685865315
```

In [123]:

```
ch=random.choice([1,2,3,4,5,6])
print(ch)
if ch==6:
    ch=random.choice([1,2,3,4,5,6])
    print(ch)
```

```
6
3
```

In [147]:

```
rd=random.randrange(0,100,3)#used=3
#gives multiples of 3
print(rd)
```

```
60
```

- **python file is created as class1.py**

- **class kkrksr:**
  - **def even(eve):**
    - **if eve%2==0:**
      - **return True**

        - return False
    - **def odd(eve):**

        ```
        - if eve%2==1:#!=0
            - return True
        ```

        - **else:**
            - **return False**

In [165]:

```
ls# shows our file name i.e class1.py
```

```
__pycache__/   class-4.ipynb  class-8.ipynb  file3.txt      module.py
class-1.ipynb  class-5.ipynb  class1.py      file4.txt
class-2.ipynb  class-6.ipynb  file1.txt      file5.txt
class-3.ipynb  class-7.ipynb  file2.txt      file6.txt
```

In [166]:

```
from class1 import kkrksr
students=kkrksr
print(students.even(5))
print(students.even(4))
print(students.odd(3))
print(students.odd(2))
```

```
False
True
True
False
```

FILTER

syntax

- **(filter,range(10))**

In [1]:

```
# filter(filter,range(20))
def fliter(n):
    if n>5:
        return n
#data=filter(fliter,range(20))
data=filter(fliter,range(20))
print(list(data))
```

```
[6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```