In [ ]:
```
Python
======
1. Procedural style - direct approach style
2. functional style - Expression (map,filter)
3. Object oriented style programming- class---object(object based appproach)


-type           -class
-real entity    - object
-function       - method
                - inheritance
                -overloading
recap python native types:
    int, float, str, bytes, None, bool, list, tuple,dict, set

in C lang,
int c;
c=10;

c   |10   |0x11111
|----------------------> int

in python
c= 10
c---> 10
```

In [ ]:
```
Syntax:-
class className:
    <attritube>
    <attributes>
    <attributes>

className.attribute
```

In [1]:
```
'''eid=100
print (eid)
''' # procedural style

class Emp:
    eid=100  # attribute  oop style

print(Emp.eid)#className.attributename
```

```
100
```

In [4]:
```
print(Emp)
print(type(Emp),type(str),type(int),type(bool))
#Emp- user defined type
#str,bool,int,None,list-predefined python type
```

```
<class '__main__.Emp'>
<class 'type'> <class 'type'> <class 'type'> <class 'type'>
```

In [5]:
```
help(str)
```

```
Help on class str in module builtins:

class str(object)
 |  str(object='') -> str
```

```
|  str(bytes_or_buffer[, encoding[, errors]]) -> str
|
|  Create a new string object from the given object. If encoding or
|  errors is specified, then the object must expose a data buffer
|  that will be decoded using the given encoding and error handler.
|  Otherwise, returns the result of object.__str__() (if defined)
|  or repr(object).
|  encoding defaults to sys.getdefaultencoding().
|  errors defaults to 'strict'.
|
|  Methods defined here:
|
|  __add__(self, value, /)
|      Return self+value.
|
|  __contains__(self, key, /)
|      Return key in self.
|
|  __eq__(self, value, /)
|      Return self==value.
|
|  __format__(self, format_spec, /)
|      Return a formatted version of the string as described by format_spec.
|
|  __ge__(self, value, /)
|      Return self>=value.
|
|  __getattribute__(self, name, /)
|      Return getattr(self, name).
|
|  __getitem__(self, key, /)
|      Return self[key].
|
|  __getnewargs__(...)
|
|  __gt__(self, value, /)
|      Return self>value.
|
|  __hash__(self, /)
|      Return hash(self).
|
|  __iter__(self, /)
|      Implement iter(self).
|
|  __le__(self, value, /)
|      Return self<=value.
|
|  __len__(self, /)
|      Return len(self).
|
|  __lt__(self, value, /)
|      Return self<value.
|
|  __mod__(self, value, /)
|      Return self%value.
|
|  __mul__(self, value, /)
|      Return self*value.
|
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __repr__(self, /)
|      Return repr(self).
|
```

```
|  __rmod__(self, value, /)
|      Return value%self.
|
|  __rmul__(self, value, /)
|      Return value*self.
|
|  __sizeof__(self, /)
|      Return the size of the string in memory, in bytes.
|
|  __str__(self, /)
|      Return str(self).
|
|  capitalize(self, /)
|      Return a capitalized version of the string.
|
|      More specifically, make the first character have upper case and the rest lower
|      case.
|
|  casefold(self, /)
|      Return a version of the string suitable for caseless comparisons.
|
|  center(self, width, fillchar=' ', /)
|      Return a centered string of length width.
|
|      Padding is done using the specified fill character (default is a space).
|
|  count(...)
|      S.count(sub[, start[, end]]) -> int
|
|      Return the number of non-overlapping occurrences of substring sub in
|      string S[start:end].  Optional arguments start and end are
|      interpreted as in slice notation.
|
|  encode(self, /, encoding='utf-8', errors='strict')
|      Encode the string using the codec registered for encoding.
|
|      encoding
|        The encoding in which to encode the string.
|      errors
|        The error handling scheme to use for encoding errors.
|        The default is 'strict' meaning that encoding errors raise a
|        UnicodeEncodeError.  Other possible values are 'ignore', 'replace' and
|        'xmlcharrefreplace' as well as any other name registered with
|        codecs.register_error that can handle UnicodeEncodeErrors.
|
|  endswith(...)
|      S.endswith(suffix[, start[, end]]) -> bool
|
|      Return True if S ends with the specified suffix, False otherwise.
|      With optional start, test S beginning at that position.
|      With optional end, stop comparing S at that position.
|      suffix can also be a tuple of strings to try.
|
|  expandtabs(self, /, tabsize=8)
|      Return a copy where all tab characters are expanded using spaces.
|
|      If tabsize is not given, a tab size of 8 characters is assumed.
|
|  find(...)
|      S.find(sub[, start[, end]]) -> int
|
|      Return the lowest index in S where substring sub is found,
|      such that sub is contained within S[start:end].  Optional
|      arguments start and end are interpreted as in slice notation.
|
```

```
|      Return -1 on failure.
|
|  format(...)
|      S.format(*args, **kwargs) -> str
|
|      Return a formatted version of S, using substitutions from args and kwargs.
|      The substitutions are identified by braces ('{' and '}').
|
|  format_map(...)
|      S.format_map(mapping) -> str
|
|      Return a formatted version of S, using substitutions from mapping.
|      The substitutions are identified by braces ('{' and '}').
|
|  index(...)
|      S.index(sub[, start[, end]]) -> int
|
|      Return the lowest index in S where substring sub is found,
|      such that sub is contained within S[start:end].  Optional
|      arguments start and end are interpreted as in slice notation.
|
|      Raises ValueError when the substring is not found.
|
|  isalnum(self, /)
|      Return True if the string is an alpha-numeric string, False otherwise.
|
|      A string is alpha-numeric if all characters in the string are alpha-numeric and
|      there is at least one character in the string.
|
|  isalpha(self, /)
|      Return True if the string is an alphabetic string, False otherwise.
|
|      A string is alphabetic if all characters in the string are alphabetic and there
|      is at least one character in the string.
|
|  isascii(self, /)
|      Return True if all characters in the string are ASCII, False otherwise.
|
|      ASCII characters have code points in the range U+0000-U+007F.
|      Empty string is ASCII too.
|
|  isdecimal(self, /)
|      Return True if the string is a decimal string, False otherwise.
|
|      A string is a decimal string if all characters in the string are decimal and
|      there is at least one character in the string.
|
|  isdigit(self, /)
|      Return True if the string is a digit string, False otherwise.
|
|      A string is a digit string if all characters in the string are digits and there
|      is at least one character in the string.
|
|  isidentifier(self, /)
|      Return True if the string is a valid Python identifier, False otherwise.
|
|      Call keyword.iskeyword(s) to test whether string s is a reserved identifier,
|      such as "def" or "class".
|
|  islower(self, /)
|      Return True if the string is a lowercase string, False otherwise.
|
|      A string is lowercase if all cased characters in the string are lowercase and
|      there is at least one cased character in the string.
|
```

```
|  isnumeric(self, /)
|      Return True if the string is a numeric string, False otherwise.
|
|      A string is numeric if all characters in the string are numeric and there is at
|      least one character in the string.
|
|  isprintable(self, /)
|      Return True if the string is printable, False otherwise.
|
|      A string is printable if all of its characters are considered printable in
|      repr() or if it is empty.
|
|  isspace(self, /)
|      Return True if the string is a whitespace string, False otherwise.
|
|      A string is whitespace if all characters in the string are whitespace and there
|      is at least one character in the string.
|
|  istitle(self, /)
|      Return True if the string is a title-cased string, False otherwise.
|
|      In a title-cased string, upper- and title-case characters may only
|      follow uncased characters and lowercase characters only cased ones.
|
|  isupper(self, /)
|      Return True if the string is an uppercase string, False otherwise.
|
|      A string is uppercase if all cased characters in the string are uppercase and
|      there is at least one cased character in the string.
|
|  join(self, iterable, /)
|      Concatenate any number of strings.
|
|      The string whose method is called is inserted in between each given string.
|      The result is returned as a new string.
|
|      Example: '.'.join(['ab', 'pq', 'rs']) -> 'ab.pq.rs'
|
|  ljust(self, width, fillchar=' ', /)
|      Return a left-justified string of length width.
|
|      Padding is done using the specified fill character (default is a space).
|
|  lower(self, /)
|      Return a copy of the string converted to lowercase.
|
|  lstrip(self, chars=None, /)
|      Return a copy of the string with leading whitespace removed.
|
|      If chars is given and not None, remove characters in chars instead.
|
|  partition(self, sep, /)
|      Partition the string into three parts using the given separator.
|
|      This will search for the separator in the string.  If the separator is found,
|      returns a 3-tuple containing the part before the separator, the separator
|      itself, and the part after it.
|
|      If the separator is not found, returns a 3-tuple containing the original string
|      and two empty strings.
|
|  replace(self, old, new, count=-1, /)
|      Return a copy with all occurrences of substring old replaced by new.
|
|        count
```

```
|          Maximum number of occurrences to replace.
|          -1 (the default value) means replace all occurrences.
|
|      If the optional argument count is given, only the first count occurrences are
|      replaced.
|
|  rfind(...)
|      S.rfind(sub[, start[, end]]) -> int
|
|      Return the highest index in S where substring sub is found,
|      such that sub is contained within S[start:end].  Optional
|      arguments start and end are interpreted as in slice notation.
|
|      Return -1 on failure.
|
|  rindex(...)
|      S.rindex(sub[, start[, end]]) -> int
|
|      Return the highest index in S where substring sub is found,
|      such that sub is contained within S[start:end].  Optional
|      arguments start and end are interpreted as in slice notation.
|
|      Raises ValueError when the substring is not found.
|
|  rjust(self, width, fillchar=' ', /)
|      Return a right-justified string of length width.
|
|      Padding is done using the specified fill character (default is a space).
|
|  rpartition(self, sep, /)
|      Partition the string into three parts using the given separator.
|
|      This will search for the separator in the string, starting at the end. If
|      the separator is found, returns a 3-tuple containing the part before the
|      separator, the separator itself, and the part after it.
|
|      If the separator is not found, returns a 3-tuple containing two empty strings
|      and the original string.
|
|  rsplit(self, /, sep=None, maxsplit=-1)
|      Return a list of the words in the string, using sep as the delimiter string.
|
|        sep
|          The delimiter according which to split the string.
|          None (the default value) means split according to any whitespace,
|          and discard empty strings from the result.
|        maxsplit
|          Maximum number of splits to do.
|          -1 (the default value) means no limit.
|
|      Splits are done starting at the end of the string and working to the front.
|
|  rstrip(self, chars=None, /)
|      Return a copy of the string with trailing whitespace removed.
|
|      If chars is given and not None, remove characters in chars instead.
|
|  split(self, /, sep=None, maxsplit=-1)
|      Return a list of the words in the string, using sep as the delimiter string.
|
|      sep
|        The delimiter according which to split the string.
|        None (the default value) means split according to any whitespace,
|        and discard empty strings from the result.
|      maxsplit
```

```
|       Maximum number of splits to do.
|       -1 (the default value) means no limit.
|
|  splitlines(self, /, keepends=False)
|      Return a list of the lines in the string, breaking at line boundaries.
|
|      Line breaks are not included in the resulting list unless keepends is given and
|      true.
|
|  startswith(...)
|      S.startswith(prefix[, start[, end]]) -> bool
|
|      Return True if S starts with the specified prefix, False otherwise.
|      With optional start, test S beginning at that position.
|      With optional end, stop comparing S at that position.
|      prefix can also be a tuple of strings to try.
|
|  strip(self, chars=None, /)
|      Return a copy of the string with leading and trailing whitespace removed.
|
|      If chars is given and not None, remove characters in chars instead.
|
|  swapcase(self, /)
|      Convert uppercase characters to lowercase and lowercase characters to uppercase.
|
|  title(self, /)
|      Return a version of the string where each word is titlecased.
|
|      More specifically, words start with uppercased characters and all remaining
|      cased characters have lower case.
|
|  translate(self, table, /)
|      Replace each character in the string using the given translation table.
|
|        table
|          Translation table, which must be a mapping of Unicode ordinals to
|          Unicode ordinals, strings, or None.
|
|      The table must implement lookup/indexing via __getitem__, for instance a
|      dictionary or list.  If this operation raises LookupError, the character is
|      left untouched.  Characters mapped to None are deleted.
|
|  upper(self, /)
|      Return a copy of the string converted to uppercase.
|
|  zfill(self, width, /)
|      Pad a numeric string with zeros on the left, to fill a field of the given width.
|
|      The string is never truncated.
|
|  ----------------------------------------------------------------------
|  Static methods defined here:
|
|  __new__(*args, **kwargs) from builtins.type
|      Create and return a new object.  See help(type) for accurate signature.
|
|  maketrans(...)
|      Return a translation table usable for str.translate().
|
|      If there is only one argument, it must be a dictionary mapping Unicode
|      ordinals (integers) or characters to Unicode ordinals, strings or None.
|      Character keys will be then converted to ordinals.
|      If there are two arguments, they must be strings of equal length, and
|      in the resulting dictionary, each character in x will be mapped to the
|      character at the same position in y. If there is a third argument, it
```

```
|      must be a string, whose characters will be mapped to None in the result.
```

In [8]:
```
#task
#create a class student with sname,sid,dept,colleg
#display these details to monitor<STDOUT>
class student:
    sname="Kavin"
    sid=101
    sdept="EEE"
    college="BIT"
print("STUDENT DETAILS","\n","*"*15)
print("NAME : {}".format(student.sname))
print("ID : {}".format(student.sid))
print("DEPARTMENT : {}".format(student.sdept))
print("COLLEGE : {}".format(student.college))
```

```
STUDENT DETAILS
 ***************
NAME : Kavin
ID : 101
DEPARTMENT : EEE
COLLEGE : BIT
```

In [ ]:
```
class - type - mutable - we can add, delete, modify  the attributes
```

In [ ]:
```
Syntax:-
    className.attributeName=newValue
```

In [9]:
```
student.sname="Santhosh"
print(student.sname)
```

```
Santhosh
```

In [10]:
```
student.bloodgroup="A+"   # adding new attribute
print(student.bloodgroup)
```

```
A+
```

In [12]:
```
s="python"
#print(S)# NameError # procedural style
class A:
    s="python"

print(A.S)#AttributeError
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-12-aa51625911c0> in <module>
      4     s="python"
      5
----> 6 print(A.S)#AttributeError

AttributeError: type object 'A' has no attribute 'S'
```

In [13]:
```
print(Emp)
#object - real Entity
#str-class
#s="python"   s is str object
```

```
<class '__main__.Emp'>
```

In [16]:
```
#objectName=className() object creation in python
Emp()
```

Out[16]:
```
<__main__.Emp at 0x8130fd0>
```

In [21]:
```
obj1=Emp()
print(obj1)
obj2=Emp()
print(obj2)

print(Emp.eid)
print(obj1.eid)
print(obj2.eid)
```

```
<__main__.Emp object at 0x00000000087E86A0>
<__main__.Emp object at 0x00000000087E82E0>
100
100
100
```

In [23]:
```
obj1.eid=500# object based initialization
print(Emp.eid)
print(obj1.eid)
print(obj2.eid)
Emp.eid=1000
print(Emp.eid)
print(obj1.eid)
print(obj2.eid)
#change happens only with direct assignment

# else follow the blue
        #changing class attr changes other object attri
```

```
100
500
100
1000
500
1000
```

In [ ]:
```
#task
#create 2 student objects sobj1, sobj2
#chnage student specific details to sobj1 and sobj2
```

In [27]:
```
sobj1=student()
print("Before object based initialization")
print("( {},{},{},{},{})".format(sobj1.sname,sobj1.sid,sobj1.sdept,sobj1.college,sobj1.
sobj1.sname="Jeny"
sobj1.sid=111
sobj1.sdept="ECE"
sobj1.college="VIT"
sobj1.bloodgroup="B+"
print("After Object Based Initialization")
print("( {},{},{},{},{})".format(sobj1.sname,sobj1.sid,sobj1.sdept,sobj1.college,sobj1.

sobj2=student()
print("Before Object Based Initialization")
print("( {},{},{},{},{})".format(sobj2.sname,sobj2.sid,sobj2.sdept,sobj2.college,sobj2.
sobj2.sname="Joy"
sobj2.sid=123
```

```
sobj2.sdept="EEE"
sobj2.college="BIT"
sobj2.bloodgroup="AB+"
print("After Object Based Initialization")
print("( {},{},{},{},{})".format(sobj2.sname,sobj2.sid,sobj2.sdept,sobj2.college,sobj2.

Emp.sname="Krish"
Emp.sid=10000
Emp.sdept="CSE"

print(sobj1.sname,sobj2.sname)
```

```
Before object based initialization
( Santhosh,101,EEE,BIT,A+)
After Object Based Initialization
( Jeny,111,ECE,VIT,B+)
Before Object Based Initialization
( Santhosh,101,EEE,BIT,A+)
After Object Based Initialization
( Joy,123,EEE,BIT,AB+)
Jeny Joy
```

In [ ]:
```
#task
#create a server_infor class with(sname)
#create 3 objects  based on blueprint
#check object based initialization
```

In [29]:
```
class server_infor:
    sname="unix"

sobj1=server_infor()
sobj2=server_infor()
sobj3=server_infor()
sobj1.sname="Winx"   # object based initialization
sobj2.sname="OL7"
sobj3.sname="MACOS"
print(sobj1.sname,sobj2.sname,sobj3.sname)
server_infor.sname="Ubuntu"  # class based initialization
print(sobj1.sname,sobj2.sname,sobj3.sname)

sobj4=server_infor()
print(sobj4.sname)
```

```
Winx OL7 MACOS
Winx OL7 MACOS
Ubuntu
```

In [ ]:
```
#task
#create a class FilesystemInfo with attributes(fstype,partition Name, mountpoint,size)
#create 3 partition details(2 instance/objects)
#make Object based Initialization
#display each partition details
```

In [ ]:
```
class
object
|
method- function
```

In [31]:
```
def f1():
    print("Hello")
```

```
print(type(f1))

class Box:
    def f2():
        print("Hello")
obj=Box()
```

```
<class 'function'>
<class 'method'>
```

In [32]:
```
s="Hello"
s.upper() # str method
```

Out[32]:
```
'HELLO'
```

In [34]:
```
L=[]
L.append(10) # List method
print(L)
```

```
[10]
```

In [ ]:
```
class str:
    def upper():
        ...
class List:
    def append():
    def insert():
        ..
```

In [35]:
```
L.upper()# upper() is a method str class
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-35-42a2fdad2ec3> in <module>
----> 1 L.upper()# upper() is a method str class

AttributeError: 'list' object has no attribute 'upper'
```

In [36]:
```
def f1():
    print("Hello")
f1()
```

```
Hello
```

In [37]:
```
f1(10)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-37-ff691e6ce47c> in <module>
----> 1 f1(10)

TypeError: f1() takes 0 positional arguments but 1 was given
```

In [38]:
```
class box:
    def f2():
        print("Hello")
obj=box()
obj.f2()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-38-67846869461b> in <module>
      3         print("Hello")
      4 obj=box()
----> 5 obj.f2()

TypeError: f2() takes 0 positional arguments but 1 was given
```

In [39]:
```
class box:
    def f2(self):# self-> current object
        print("Hello")
obj=box()
obj.f2()
```

```
Hello
```

In [ ]:
```
#obj.f2()----> pvm----> f2(obj)
#obj2.f2()---> pvm---> f2(obj2)
#obj3.f2(10,20,30)---pvm---> f2(obj3,10,20,30)  internal conversion
```

In [43]:
```
class box:
    def f2(self):# self-> current object
        print("self value:{}".format(self))
obj1=box()
obj1.f2()
print("Obj1 value :{}".format(obj1))
print()

obj2=box()
obj2.f2()
print("Obj2 value :{}".format(obj2))
print()

obj3=box()
obj3.f2()
print("Obj3 value :{}".format(obj3))
```

```
self value:<__main__.box object at 0x00000000083119A0>
Obj1 value :<__main__.box object at 0x00000000083119A0>

self value:<__main__.box object at 0x00000000087E85E0>
Obj2 value :<__main__.box object at 0x00000000087E85E0>

self value:<__main__.box object at 0x00000000087E8EE0>
Obj3 value :<__main__.box object at 0x00000000087E8EE0>
```

In [46]:
```
class box:
    bid=100
    bname="box-1"

    def f1(self):
        print("hello Hai")

    def f2(self):
        #print(bname) NameError
        print("box name:",self.bname,"box-id:",self.bid)
obj=box()
obj.f1()
obj.f2()
```

```
hello Hai
box name: box-1 box-id: 100
```

In [48]:
```python
class server_info:
    sname="server"

    def f1(self,n): #object based initialization
        self.sname=n

    def f2(self): # atrribute display
        print("Server Name:{}".format(self.sname))
obj1=server_info()
obj2=server_info()

obj1.f1("Unix")
obj2.f1("Linux")

obj1.f2()
obj2.f2()
```

```
Server Name:Unix
Server Name:Linux
```

In [ ]:
```python
#task
#Modify the student class
    #1. create a method f1() to perform object based initialization
    #2. create a method f2() to display the attributes details to <STDOUT>
```

In [51]:
```python
class student:
    sname=""
    sid=0
    sdept=""
    college=""

    def f1(self,name,id,dept,col):
        self.sname= name
        self.sid= id
        self.sdept=dept
        self.college = col

    def f2(self):
        print("STUDENT DETAILS","\n","*"*15)
        print("NAME : {}".format(self.sname))
        print("ID : {}".format(self.sid))
        print("DEPARTMENT : {}".format(self.sdept))
        print("COLLEGE : {}".format(self.college))

obj1=student()
obj1.f1("Ram",101,"EEE","BIT")
obj2= student()
obj2.f1("Sai",102,"CSE","VIT")

obj1.f2()
obj2.f2()
```

```
STUDENT DETAILS
 ***************
NAME : Ram
ID : 101
DEPARTMENT : EEE
COLLEGE : BIT
```

```
STUDENT DETAILS
 ***************
NAME : Sai
ID : 102
DEPARTMENT : CSE
COLLEGE : VIT
```

In [ ]:
```
class classname:
    <attribute>
    <attribute>
    def methodName(self):
        ..
```

In [ ]:
```
class Vendor:
    vname=''
    vid=0
    product_list=[]

    def f1(self):
        ..
    def display():
        ..
```

In [ ]:
```
class db1:
    def f1(self, a1,a2):
        ...

    def f2(Self):
        query1

    def f3(self):
        query2

obj1=db1()
obj1.f1("dbconnectpara") - "OK"
obj1.f2()
obj1.f3()

# Db- rule---> connection---> query
obj2=db1()
obj2.f2()
obj2.f1()// python- correct- valid call
```

In [ ]:
```
#Constructor
|-----special method (or) buildin method - Object Initialization

--> called automatically when we create an object


__variable__    __methodName__() ----> special
```

In [56]:
```
#constructor---> __init__()
class Enrollement:
    def __init__(self,n,dob): #constructor
        self.sname=n
        self.sdob=dob

    def f2(self):
```

```
        print("About {} emp details:".format(self.sname))
        print("Name:{}\tDOB:{}".format(self.sname,self.sdob))

eobj1=Enrollement("Theeba","12Jan1999")
# object creation-constructor called automatically

eobj1.f2()
```

```
About Theeba emp details:
Name:Theeba     DOB:12Jan1999
```

In [58]:
```
eobj2=Enrollement() #TypeError
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-58-1b121d3dbf2b> in <module>
----> 1 eobj2=Enrollement() #TypeError

TypeError: __init__() missing 2 required positional arguments: 'n' and 'dob'
```

In [60]:
```
class Enroll:
    def f1(self):
        print("Inside a NON-CONSTRUCTOR")

obj=Enroll()
obj.f1()
```

```
Inside a NON-CONSTRUCTOR
```

In [63]:
```
class Enroll:
    def __init__(self):
        print("Inside a CONSTRUCTOR")

print(Enroll())
```

```
Inside a CONSTRUCTOR
<__main__.Enroll object at 0x0000000004E33BB0>
```

In [64]:
```
obj1=Enroll()
```

```
Inside a CONSTRUCTOR
```

In [67]:
```
class Enroll:
    def __init__(self,n,i):
        print("Inside a CONSTRUCTOR")

Enroll() # Enroll(1,2)-> valid
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-67-3e3b9b2ecff3> in <module>
      3         print("Inside a CONSTRUCTOR")
      4
----> 5 Enroll()

TypeError: __init__() missing 2 required positional arguments: 'n' and 'i'
```

In [71]:
```
i=10 # procedural style

i=int(10)# oop style

print(type(i))
help(int)
```

```python
class int:
    def __int__(self,value):
        self.value=value
```

```
<class 'int'>
Help on class int in module builtins:

class int(object)
 |  int([x]) -> integer
 |  int(x, base=10) -> integer
 |
 |  Convert a number or string to an integer, or return 0 if no arguments
 |  are given.  If x is a number, return x.__int__().  For floating point
 |  numbers, this truncates towards zero.
 |
 |  If x is not a number or if base is given, then x must be a string,
 |  bytes, or bytearray instance representing an integer literal in the
 |  given base.  The literal can be preceded by '+' or '-' and be surrounded
 |  by whitespace.  The base defaults to 10.  Valid bases are 0 and 2-36.
 |  Base 0 means to interpret the base from the string as an integer literal.
 |  >>> int('0b100', base=0)
 |  4
 |
 |  Built-in subclasses:
 |      bool
 |
 |  Methods defined here:
 |
 |  __abs__(self, /)
 |      abs(self)
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __and__(self, value, /)
 |      Return self&value.
 |
 |  __bool__(self, /)
 |      self != 0
 |
 |  __ceil__(...)
 |      Ceiling of an Integral returns itself.
 |
 |  __divmod__(self, value, /)
 |      Return divmod(self, value).
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __float__(self, /)
 |      float(self)
 |
 |  __floor__(...)
 |      Flooring an Integral returns itself.
 |
 |  __floordiv__(self, value, /)
 |      Return self//value.
 |
 |  __format__(self, format_spec, /)
 |      Default object formatter.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
```

```
 |      Return getattr(self, name).
 |
 |  __getnewargs__(self, /)
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __hash__(self, /)
 |      Return hash(self).
 |
 |  __index__(self, /)
 |      Return self converted to an integer, if self is suitable for use as an index int
o a list.
 |
 |  __int__(self, /)
 |      int(self)
 |
 |  __invert__(self, /)
 |      ~self
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __lshift__(self, value, /)
 |      Return self<<value.
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __mod__(self, value, /)
 |      Return self%value.
 |
 |  __mul__(self, value, /)
 |      Return self*value.
 |
 |  __ne__(self, value, /)
 |      Return self!=value.
 |
 |  __neg__(self, /)
 |      -self
 |
 |  __or__(self, value, /)
 |      Return self|value.
 |
 |  __pos__(self, /)
 |      +self
 |
 |  __pow__(self, value, mod=None, /)
 |      Return pow(self, value, mod).
 |
 |  __radd__(self, value, /)
 |      Return value+self.
 |
 |  __rand__(self, value, /)
 |      Return value&self.
 |
 |  __rdivmod__(self, value, /)
 |      Return divmod(value, self).
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __rfloordiv__(self, value, /)
 |      Return value//self.
 |
```

```
|  __rlshift__(self, value, /)
|      Return value<<self.
|
|  __rmod__(self, value, /)
|      Return value%self.
|
|  __rmul__(self, value, /)
|      Return value*self.
|
|  __ror__(self, value, /)
|      Return value|self.
|
|  __round__(...)
|      Rounding an Integral returns itself.
|      Rounding with an ndigits argument also returns an integer.
|
|  __rpow__(self, value, mod=None, /)
|      Return pow(value, self, mod).
|
|  __rrshift__(self, value, /)
|      Return value>>self.
|
|  __rshift__(self, value, /)
|      Return self>>value.
|
|  __rsub__(self, value, /)
|      Return value-self.
|
|  __rtruediv__(self, value, /)
|      Return value/self.
|
|  __rxor__(self, value, /)
|      Return value^self.
|
|  __sizeof__(self, /)
|      Returns size in memory, in bytes.
|
|  __sub__(self, value, /)
|      Return self-value.
|
|  __truediv__(self, value, /)
|      Return self/value.
|
|  __trunc__(...)
|      Truncating an Integral returns itself.
|
|  __xor__(self, value, /)
|      Return self^value.
|
|  as_integer_ratio(self, /)
|      Return integer ratio.
|
|      Return a pair of integers, whose ratio is exactly equal to the original int
|      and with a positive denominator.
|
|      >>> (10).as_integer_ratio()
|      (10, 1)
|      >>> (-10).as_integer_ratio()
|      (-10, 1)
|      >>> (0).as_integer_ratio()
|      (0, 1)
|
|  bit_length(self, /)
|      Number of bits necessary to represent self in binary.
|
```

```
|      >>> bin(37)
|      '0b100101'
|      >>> (37).bit_length()
|      6
|
|  conjugate(...)
|      Returns self, the complex conjugate of any int.
|
|  to_bytes(self, /, length, byteorder, *, signed=False)
|      Return an array of bytes representing an integer.
|
|      length
|        Length of bytes object to use.  An OverflowError is raised if the
|        integer is not representable with the given number of bytes.
|      byteorder
|        The byte order used to represent the integer.  If byteorder is 'big',
|        the most significant byte is at the beginning of the byte array.  If
|        byteorder is 'little', the most significant byte is at the end of the
|        byte array.  To request the native byte order of the host system, use
|        `sys.byteorder' as the byte order value.
|      signed
|        Determines whether two's complement is used to represent the integer.
|        If signed is False and a negative integer is given, an OverflowError
|        is raised.
|
|  ----------------------------------------------------------------------
|  Class methods defined here:
|
|  from_bytes(bytes, byteorder, *, signed=False) from builtins.type
|      Return the integer represented by the given array of bytes.
|
|      bytes
|        Holds the array of bytes to convert.  The argument must either
|        support the buffer protocol or be an iterable object producing bytes.
|        Bytes and bytearray are examples of built-in objects that support the
|        buffer protocol.
|      byteorder
|        The byte order used to represent the integer.  If byteorder is 'big',
|        the most significant byte is at the beginning of the byte array.  If
|        byteorder is 'little', the most significant byte is at the end of the
|        byte array.  To request the native byte order of the host system, use
|        `sys.byteorder' as the byte order value.
|      signed
|        Indicates whether two's complement is used to represent the integer.
|
|  ----------------------------------------------------------------------
|  Static methods defined here:
|
|  __new__(*args, **kwargs) from builtins.type
|      Create and return a new object.  See help(type) for accurate signature.
|
|  ----------------------------------------------------------------------
|  Data descriptors defined here:
|
|  denominator
|      the denominator of a rational number in lowest terms
|
|  imag
|      the imaginary part of a complex number
|
|  numerator
|      the numerator of a rational number in lowest terms
|
|  real
```

```
|      the real part of a complex number
```

In [ ]:
```
s1="Hello"# procedural

s1=str("Hello") # constructor call--> <class str>

L=[] # procedural

L=list()#oop
class list:
    def __init__(self..):
        ...

    def append(self,object):
        ..

    def insert(self,index,object):
        ..
```

In [72]:
```
a=10
b=20
a+b
```

Out[72]:
```
30
```

In [ ]:
```
special method (or)  dunder method  __methodNmae__()
```

In [73]:
```
a<b
```

Out[73]:
```
True
```

In [74]:
```
b>a
```

Out[74]:
```
True
```

In [75]:
```
#dir(className)->list of methods in this class
dir(str)
```

Out[75]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
```

```
'__mod__',
'__mul__',
'__ne__',
'__new__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__rmod__',
'__rmul__',
'__setattr__',
'__sizeof__',
'__str__',
'__subclasshook__',
'capitalize',
'casefold',
'center',
'count',
'encode',
'endswith',
'expandtabs',
'find',
'format',
'format_map',
'index',
'isalnum',
'isalpha',
'isascii',
'isdecimal',
'isdigit',
'isidentifier',
'islower',
'isnumeric',
'isprintable',
'isspace',
'istitle',
'isupper',
'join',
'ljust',
'lower',
'lstrip',
'maketrans',
'partition',
'replace',
'rfind',
'rindex',
'rjust',
'rpartition',
'rsplit',
'rstrip',
'split',
'splitlines',
'startswith',
'strip',
'swapcase',
'title',
'translate',
'upper',
'zfill']
```

In [77]:

```
class A:
    def __init__(self,a=0):
        self.a=a
```

```
obj=A()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-77-e52395e34615> in <module>
      3         self.a=a
      4 obj=A()
----> 5 len(obj)

TypeError: object of type 'A' has no len()
```

In [76]:
```
s="python"  # s-> obj of str class
len(s)
```

Out[76]:
```
6
```

In [78]:
```
class A:
    def __init__(self,a=0):
        self.a=a
    def __len__(self):
        return 10
obj=A()
len(obj)
```

Out[78]:
```
10
```

In [80]:
```
f=1.333
str(f)
```

Out[80]:
```
'1.333'
```

In [81]:
```
obj=A()
str(obj)
```

Out[81]:
```
'<__main__.A object at 0x00000000085B1790>'
```

In [82]:
```
dir(A)
```

Out[82]:
```
['__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__len__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
```

```
'__sizeof__',
'__str__',
'__subclasshook__',
'__weakref__']
```

In [84]:
```
a=10
b=20
a+b
dir(int)
```

Out[84]:
```
['__abs__',
 '__add__',
 '__and__',
 '__bool__',
 '__ceil__',
 '__class__',
 '__delattr__',
 '__dir__',
 '__divmod__',
 '__doc__',
 '__eq__',
 '__float__',
 '__floor__',
 '__floordiv__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__index__',
 '__init__',
 '__init_subclass__',
 '__int__',
 '__invert__',
 '__le__',
 '__lshift__',
 '__lt__',
 '__mod__',
 '__mul__',
 '__ne__',
 '__neg__',
 '__new__',
 '__or__',
 '__pos__',
 '__pow__',
 '__radd__',
 '__rand__',
 '__rdivmod__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rfloordiv__',
 '__rlshift__',
 '__rmod__',
 '__rmul__',
 '__ror__',
 '__round__',
 '__rpow__',
 '__rrshift__',
 '__rshift__',
 '__rsub__',
 '__rtruediv__',
 '__rxor__',
```

```
'__setattr__',
'__sizeof__',
'__str__',
'__sub__',
'__subclasshook__',
'__truediv__',
'__trunc__',
'__xor__',
'as_integer_ratio',
'bit_length',
'conjugate',
'denominator',
'from_bytes',
'imag',
'numerator',
'real',
'to_bytes']
```

In [85]:
```
a.__add__(b)
```

Out[85]:
```
30
```

In [86]:
```
a.__lt__(b)
```

Out[86]:
```
True
```

In [87]:
```
a.__ge__(b)
```

Out[87]:
```
False
```

In [89]:
```
class box:
    '''About this class is ljd;jd;jdsldj
    dksdkljdjd;sljds;ldjs;
    djs;ldjs;lksldkl
    lsjdsldsl'''
    var=100
    fname="khdldjs"
    def display(self):
        return 10

box.__dict__
```

Out[89]:
```
mappingproxy({'__module__': '__main__',
              '__doc__': 'About this class is ljd;jd;jdsldj\n    dksdkljdjd;sljds;ldj
s;\n    djs;ldjs;lksldkl\n    lsjdsldsl',
              'var': 100,
              'fname': 'khdldjs',
              'display': <function __main__.box.display(self)>,
              '__dict__': <attribute '__dict__' of 'box' objects>,
              '__weakref__': <attribute '__weakref__' of 'box' objects>})
```

In [90]:
```
box.__doc__
```

Out[90]:
```
'About this class is ljd;jd;jdsldj\n    dksdkljdjd;sljds;ldjs;\n    djs;ldjs;lksldkl\n
lsjdsldsl'
```

In [92]:
```
import cgi
print(cgi.FieldStorage.__doc__)
```

```
Store a sequence of fields, reading multipart/form-data.
```

```
    This class provides naming, typing, files stored on disk, and
    more.  At the top level, it is accessible like a dictionary, whose
    keys are the field names.  (Note: None can occur as a field name.)
    The items are either a Python list (if there's multiple values) or
    another FieldStorage or MiniFieldStorage object.  If it's a single
    object, it has the following attributes:

    name: the field name, if specified; otherwise None

    filename: the filename, if specified; otherwise None; this is the
        client side filename, *not* the file name on which it is
        stored (that's a temporary file you don't deal with)

    value: the value as a *string*; for file uploads, this
        transparently reads the file every time you request the value
        and returns *bytes*

    file: the file(-like) object from which you can read the data *as
        bytes* ; None if the data is stored a simple string

    type: the content-type, or None if not specified

    type_options: dictionary of options specified on the content-type
        line

    disposition: content-disposition, or None if not specified

    disposition_options: dictionary of corresponding options

    headers: a dictionary(-like) object (sometimes email.message.Message or a
        subclass thereof) containing *all* headers

    The class is subclassable, mostly for the purpose of overriding
    the make_file() method, which is called internally to come up with
    a file open for reading and writing.  This makes it possible to
    override the default choice of storing all files in a temporary
    directory and unlinking them as soon as they have been opened.
```

In [94]:

```
class login:
    username="admin"
    passwd="362525728"

login.username +" " + login.passwd
```

Out[94]:

```
'admin 362525728'
```

In [96]:

```
#Inside a class-> any variable (or) method starts with__, it private attributes

class login:
    username="admin"
    __passwd="758765"

login.__passwd
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-96-b1581399ed53> in <module>
      5     __passwd="758765"
      6
----> 7 login.__passwd
```

```
AttributeError: type object 'login' has no attribute '__passwd'
```

In [97]:
```
obj=login()
obj.__passwd
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-97-32c069e9b9f5> in <module>
      1 obj=login()
----> 2 obj.__passwd

AttributeError: 'login' object has no attribute '__passwd'
```

In [98]:
```
login.__dict__
```

Out[98]:
```
mappingproxy({'__module__': '__main__',
              'username': 'admin',
              '_login__passwd': '758765',
              '__dict__': <attribute '__dict__' of 'login' objects>,
              '__weakref__': <attribute '__weakref__' of 'login' objects>,
              '__doc__': None})
```

In [ ]:
```

```