24.OOps

June 8, 2020

OOPs

```
**encapsulation**
**data hiding**
**abstraction**
**inheritance**
**polymorphsim**
**shared memory**
**object**
**class**
**constructor**
**destructor**
```

function and methods

```
method is a python function which always takes first argument, a reference of object space

def func():
    pass

function can be called directly in global scope

def method(object_ref):
    pass

method can only be called by objects
```

0.0.1 Syntax

```
class class_name(parent1, parent2, ...):
    """
        doc string
    """
    class_variable = 'class variable which sharable across all object'
    def method1(object_ref):
        """
            doc string of method
        """
        body of method
    def method2(some_object):
        pass
    def method3(self):
        self represents a object space
```

[27]:
```python
class Human:
    type = "A selfish kind"
    def laugh(self):
        """
            Instance (object) methods
        """
        print(self, id(self))
        print("ha ha ha ah ha")
    def cry(self):
        print("aahu aahu aahu")
```

[11]:
```python
sachin = Human() # it returns self
```

[14]:
```python
print(sachin, id(sachin))
```

```
<__main__.Human object at 0x000001C1B3FC6C08> 1931459980296
```

[16]:
```python
sachin.laugh()
# sachin --> Human --> Human.laugh(sachin)
```

```
<__main__.Human object at 0x000001C1B3FC6C08> 1931459980296
ha ha ha ah ha
```

[17]:
```python
Human.laugh(sachin)
```

```
<__main__.Human object at 0x000001C1B3FC6C08> 1931459980296
ha ha ha ah ha
```

[4]:
```python
sachin.type
```

[4]:
```
'A selfish kind'
```

[5]:
```python
print(id(sachin))
```

```
1931459902728
```

[6]:
```python
print(type(Human))
```

```
<class 'type'>
```

[7]:
```python
print(Human)
```

```
<class '__main__.Human'>
```

[8]:
```
print(sachin) # object representation
```

```
<__main__.Human object at 0x000001C1B3FB3D08>
```

[9]:
```
sachin.laugh() # ? self
```

```
ha ha ha ah ha
```

[18]:
```
sachin.cry()
```

```
aahu aahu aahu
```

[19]:
```
l = []
```

[20]:
```
l.append(10)
```

[21]:
```
print(l)
```

```
[10]
```

[22]:
```
list.append(l, 20)
```

[23]:
```
l
```

[23]:
```
[10, 20]
```

[24]:
```
help(list.append)
```

```
Help on method_descriptor:

append(self, object, /)
    Append object to the end of the list.
```

[25]:
```
help(l.append)
```

```
Help on built-in function append:

append(object, /) method of builtins.list instance
    Append object to the end of the list.
```

[26]:
```
Human.laugh()
```

```
        ␣
↪---------------------------------------------------------------------------
```

```
        TypeError                                 Traceback (most recent call␣
 ↪last)

        <ipython-input-26-86e0db47740b> in <module>
    ----> 1 Human.laugh()


        TypeError: laugh() missing 1 required positional argument: 'self'
```

create a car class having thease methods

wheels

breaking_system

streaing

speed

object.method() -> class.method(object)

[32]:
```python
class A:
    def hello():
        print("Hello")
a = A()
```

[33]:
```python
a.hello()

# A.hello(a)
```

```
      ␣
 ↪---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call␣
 ↪last)

        <ipython-input-33-9537d85850c3> in <module>
    ----> 1 a.hello()
          2
          3 # A.hello(a)


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

[34]:
```python
A.hello()
```

```
Hello
```

[35]:
```python
class A:
    def hello(self):
        print("Hello")
```

[36]:
```python
a = A()
```

[37]:
```python
a.hello()
# A.hello(a)
```

```
Hello
```

[38]:
```python
A.hello()
```

```
        ␣
 ↪---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call␣
 ↪last)

        <ipython-input-38-18211e912635> in <module>
    ----> 1 A.hello()


        TypeError: hello() missing 1 required positional argument: 'self'
```

[28]:
```python
from tqdm import tqdm
from time import sleep
```

[30]:
```python
for _ in tqdm(range(900)):
    sleep(1)
```

```
100%|                                      |
900/900 [15:01<00:00,  1.00s/it]
```

[39]:
```python
class Car:
    def wheels(self):
        print('four wheel drive')
    def breaking(self, abs):
        print(f"I have {abs} breaking system")
```

methods are bounded to object space so first argument is always an object.

[40]:
```python
c = Car()
```

[41]:
```
c.wheels()
# Car.wheels(c)
```

```
four wheel drive
```

[43]:
```
c.breaking('normal')
```

```
I have normal breaking system
```

[44]:
```
c.breaking('abs')
```

```
I have abs breaking system
```

[46]:
```
c.name = 'nano'
# setting an attribute to an object
# Dynamic Binding
```

[47]:
```
print(c.name)
```

```
nano
```

What is a constructor ?

A special method which called whenever an object created, usually used to set default attribut

What are magic methods in Python ?

duck typing

__method__ magic methods or dunder methods usually universal methods and operators which are c

[48]:
```
### duck typing
l = [ 1, 2, 3, 4]
```

[52]:
```
print(len(l)) # magic methods
```

```
4
```

[50]:
```
s = "hello world"
```

[53]:
```
print(len(s))
```

```
11
```

len is duck type method supported on all iterable object

[54]:
```
l = [ 1, 2, 3, 4]

print(len(l))
print(list.__len__(l))
# class.__method__(self)
```

```
4
4
```

[55]:
```
s = "hello world"
print(len(s))

print(str.__len__(s))
# class.__method__(object/self)
```

```
11
11
```

[63]:
```
class Human:
    def __len__(self):
        return 5
    def laugh(self):
        print("ha ha ha ha ha")

class Animal:
    def __len__(self):
        return 10
    def laugh(self):
        print("aaaa aaaa aaaa aaaa")
```

[64]:
```
l = [ Human(), Animal() ]
```

[65]:
```
l
```

[65]:
```
[<__main__.Human at 0x1c1b4e5b748>, <__main__.Animal at 0x1c1b4e5b708>]
```

[66]:
```
# duck typing
for obj in l:
    obj.laugh()
```

```
ha ha ha ha ha
aaaa aaaa aaaa aaaa
```

[68]:
```
l = [ 'hello', [1, 2, 3, 4], Human(), Animal()]
for obj in l:
    print(len(obj), type(obj))
```

```
5 <class 'str'>
4 <class 'list'>
```

```
5 <class '__main__.Human'>
10 <class '__main__.Animal'>
```

```
[70]: l = list([1, 2, 3])
```

```
[71]: print(l)
```

```
[1, 2, 3]
```

```
[72]: s  = str('hello')
      print(s)
```

```
hello
```

```
[73]: h = Human()
```

```
[74]: print(h) # standard output
```

```
<__main__.Human object at 0x000001C1B4E33488>
```

```
[75]: h # shell output
```

```
[75]: <__main__.Human at 0x1c1b4e33488>
```

```
[76]: fp = open('output.txt', 'w')
```

```
[77]: print("Hello world", file=fp)
```

```
[78]: print(h, file=fp)
```

```
[79]: fp.close()
```

```
[80]: !type output.txt
```

```
Hello world
<__main__.Human object at 0x000001C1B4E33488>
```

```
[81]: print("hello world")
```

```
hello world
```

help(print)

```
strings ?
print(l) --> string


print(l) --> l ? type(l) --> list --> list.__str__(l) --> string representation of list --> wri
```

```
[84]: h.__str__() # string representation
```

```
[84]: '<__main__.Human object at 0x000001C1B4E33488>'
```

```
[85]: h.__repr__() # raw representation
```

```
[85]: '<__main__.Human object at 0x000001C1B4E33488>'
```

```
[92]: class A:
          def __str__(self):
              print("ohh i see this is how print works")
              return 'object representation'
```

```
[93]: a = A()
```

```
[94]: print(a)
```

```
ohh i see this is how print works
object representation
```

```
[95]: a # __repr__
```

```
[95]: <__main__.A at 0x1c1b4e76e88>
```

```
[96]: class A:
          pass
```

```
[97]: a = A()
      print(a)
```

```
<__main__.A object at 0x000001C1B4E71048>
```

```
[99]: issubclass(A, object)
      # object class is by bedefault extened in every class of python
```

```
[99]: True
```

```
[100]: 5 + 6
```

```
[100]: 11
```

```
[101]: int.__add__(5, 6)
```

```
[101]: 11
```

```
[102]: class A:
           pass
```

[103]:
```python
a = A()
b = A()
```

[104]:
```python
a + b
```

```
                ␣
 ↪---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call␣
 ↪last)

        <ipython-input-104-bd58363a63fc> in <module>
    ----> 1 a + b


        TypeError: unsupported operand type(s) for +: 'A' and 'A'
```

[105]:
```python
class A:
    def __add__(self, other):
        print("bhai bhai")
```

[106]:
```python
a = A()
```

[107]:
```python
b = A()
```

[108]:
```python
a + b
```

```
bhai bhai
```

[109]:
```python
class A:
    def __repr__(self):
        return "Ab apunich bhagwan hai, jo apan chahega vo hoga"
```

[110]:
```python
a = A()
```

[111]:
```python
print(a)
```

```
Ab apunich bhagwan hai, jo apan chahega vo hoga
```

[112]:
```python
a
```

[112]:
```
Ab apunich bhagwan hai, jo apan chahega vo hoga
```

Understand how an object is created

repr --> str

[113]:
```
class A:
    def __repr__(self):
        return "SHELL OUTPUT"
    def __str__(self):
        return "STANDARD OUTPUT"
```

[114]:
```
a = A()
```

[115]:
```
print(a)
```

```
STANDARD OUTPUT
```

[116]:
```
a
```

[116]:
```
SHELL OUTPUT
```

[129]:
```
class A:
    def __new__(cls):
        """Constructor which creates object"""
        print("__"*30)
        print('__new__ Creating Space / Memory Allocation for object of type␣
 ↪cls A')
        print(cls, type(cls))
        print("__"*30)
        return object.__new__(cls)
    def __init__(self):
        """Constructor which initlizes values in object"""
        print("**"*30)
        self.name = 'A class' # features
        self.value = "Default Value" # attribute
        print(self, type(self))
        print("__init__ Initlizing Default Values to obejct")
        print("**"*30)

    def __str__(self):
        return self.name
    def show_data(self):
        print("Name: ", self.name)
        print("Value: ", self.value)
```

[130]:
```
a = A()
```

```
____________________________________________________________
__new__ Creating Space / Memory Allocation for object of type cls A
<class '__main__.A'> <class 'type'>
____________________________________________________________
************************************************************
```

```
A class <class '__main__.A'>
__init__ Initlizing Default Values to obejct
************************************************************
```

[131]:
```
a.name
```

[131]:
```
'A class'
```

[132]:
```
a.value
```

[132]:
```
'Default Value'
```

[133]:
```
a.show_data()
```

```
Name:  A class
Value:  Default Value
```

[134]:
```
print(a)
```

```
A class
```

[135]:
```
class Human:
    def __init__(self, name, country, laugh_type):
        """
            self - instance itself (object)

            name, country those are local variables

            self.name, self.country - instance variables

            self.laugh_type instance variable
        """
        self.name = name
        self.country = country
        self.laugh_type = laugh_type
    def __str__(self):
        return self.name.upper()

    def laugh(self):
        print(f"{self.name}: {self.laugh_type}")
    def show_data(self):
        print("Name    : ", self.name)
        print("Country : ", self.country)
```

[136]:
```
sachin = Human('sachin yadav', 'India', 'ha ha ha ha haaa haa haa')
rajat = Human('rajat goyal', 'USA', 'he he he he ha ha he he ha ha')
```

[137]:
```
print(sachin)
```

```
SACHIN YADAV
```

[138]:
```python
print(rajat)
```

```
RAJAT GOYAL
```

[139]:
```python
sachin.show_data()
```

```
Name    :  sachin yadav
Country :  India
```

[140]:
```python
rajat.show_data()
```

```
Name    :  rajat goyal
Country :  USA
```

[141]:
```python
sachin.laugh()
```

```
sachin yadav: ha ha ha ha haaa haa haa
```

[142]:
```python
rajat.laugh()
```

```
rajat goyal: he he he he ha ha he he ha ha
```

[143]:
```python
class A:
    pass
```

[144]:
```python
a = A()
```

[145]:
```python
a
```

[145]:
```
<__main__.A at 0x1c1b3e1f288>
```

[146]:
```python
print(dir(A))
```

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__',
'__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__',
'__str__', '__subclasshook__', '__weakref__']
```

[147]:
```python
print(issubclass(A, object))
```

```
True
```

[151]:
```python
class A(object):
    def __init__(self): # over-riding
        self.name = 'Sachin'
    def __str__(self):
        return self.name.upper()
```

[152]:
```
a = A() #
```

[153]:
```
print(a)
```

```
SACHIN
```

[ ]:
```
```

[ ]:
```
```