In [ ]:
```
'''
dunder : double underscore

__init__
__str__
__len__
__getitem__
__setitem__
__repr__
__call__

__add__
__gt__
__lt__
__eq__

'''
```

In [ ]:
```
class A:
    def __init__(self):
        self.lst = [45, 89, 12]

ob = A()
print(ob) # <__main__.A object at 0x000002562E996F98>
```

In [ ]:
```
class A:
    def __init__(self):
        self.lst = [45, 89, 12]

    def __str__(self):
        return str(self.lst)

ob = A()
print(ob)       # [45, 89, 12]
print(ob.lst)   # [45, 89, 12]
```

In [ ]:
```
class A:
    def __init__(self):
        self.lst = [45, 89, 12]

    def __str__(self):
        return str(self.lst)

    def __len__(self):
        return len(self.lst)

ob = A()

print(len(ob))  # 3
```

In [ ]:
```
class A:
    def __init__(self):
        self.lst = [45, 89, 12]

    def __str__(self):
        return str(self.lst)

    def __getitem__(self, i):
        return self.lst[i]

ob = A()

print(ob[1]) # 89
```

In [ ]:
```
class A:
    def __init__(self):
        self.lst = [45, 89, 12]

    def __str__(self):
        return str(self.lst)

    def __getitem__(self, i):
        return self.lst[i]

    def __setitem__(self, i, v):
        self.lst[i] = v

ob = A()

ob[1] = 13
print(ob[1]) # 13
```

In [ ]:
```
class Clock:
    def __init__(self , h, m, s):
        self.h = h
        self.m = m
        self.s = s

    def __str__(self):
        return "{0:02d}:{1:02d}:{2:02d}".format(self.h , self.m, self.s)

ob = Clock(4, 26, 30)
print(ob)  # 4:26:30
```

In [ ]:
```
class Address:
    def __init__(self, c, s, z):
        self.city = c
        self.street = s
        self.zipcode = z

    def __str__(self):
        lst = []
        lst.append(f'{self.city} -{self.street} -{self.zipcode}')
        return ' '.join(lst)

a = Address('Hamedan', 'b' , '123')
print(a)  # Hamedan -b -123
```

In [ ]:
```
class Robot:
    def __init__(self, n, y):
        self.name = n
        self.build_year = y

    def __str__(self):
        return 'name:' + self.name + ',build year :'+str(self.build_year)

    def __repr__(self):
        return "Robot(\"" + self.name + "\" , " + str(self.build_year) + ")"

ob = Robot('rr', 1980)
print(ob)       # name : rr , build year : 1980

print(repr(ob))  # Robot("rr",1980)
```

In [ ]:
```
# __call__

class C:
    def __init__(self, size , x, y):
        self.size = size
        self.x = x
        self.y = y


    def __call__(self, x, y):
        self.x = x
        self.y = y

ob = C(300, 10 , 20)
print(ob.size) # 300
print(ob.x)    # 10
print(ob.y)    # 20

ob(30,50)

print(ob.size) # 300
print(ob.x)    # 30
print(ob.y)    # 50
```

In [ ]:
```
##################################
# overload an binary + operator
```

In [ ]:
```
# __add__

class Complex:

    def __init__(self, a, b):
        self.a = a
        self.b = b

    def __add__(self, o):
        x = self.a + o.a
        y = self.b + o.b
        return x, y

ob1 = Complex(1, 3)     # 1 + 3i
ob2 = Complex(2, 4)     # 2 + 4i
ob3 = ob1 + ob2
print(ob3)              # (3, 7)
```

In [ ]:
```
class Test:
    def __init__(self, a):
        self.a = a

    def __add__(self,o):
        return self.a + o.a

ob1 = Test(1)
ob2 = Test(4)
print(ob1 + ob2)   # 5

ob1 = Test('ali')
ob2 = Test('reza')
print(ob1 + ob2)   # alireza
```

In [ ]:

```python
class AB:
    def __init__(self, a):
        self.a = a

    def __gt__(self, o ):
        if(self.a > o.a):
            return True
        else:
            return False

ob1 = AB(2)
ob2 = AB(5)

if(ob1 > ob2):
  print('yes')
else:
  print('no')

print(ob1 > ob2) # False
```

In [ ]:

```python
class ABC:
    def __init__(self, a):
        self.a = a

    def __lt__(self, o ):
        if(self.a < o.a):
            return True
        else:
            return False

    def __eq__(self, o):
        if (self.a == o.a):
            return 'equal'
        else:
            return 'not equal'

ob1 = ABC(2)
ob2 = ABC(5)

print(ob1 < ob2)  # True

print(ob1 == ob2) # not equal
```

In [ ]:

```
# data descriptor

class A:
    def __init__(self, a=None):
        print('init')
        self.__set__(self, a)

    def __set__(self, i, v):
        print('set')
        self.v = v
        print(self.v)

    def __get__(self, i , o):
        print('get')
        return self.v + 1

class B:
    x = A(5)      # init  set


ob = B()
ob.x = 8     # set  8
print(ob.x) # get  9
```

دانشگاه شهید مدنی آذربایجان
برنامه نویسی پیشرفته با پایتون
امین گلزاری اسکوئی
۱۴۰۰-۱۴۰۱

[Codes and Projects (click here) (https://github.com/Amin-Golzari-Oskouei/Python-Programming-Course-Advanced-2021)](https://github.com/Amin-Golzari-Oskouei/Python-Programming-Course-Advanced-2021) [slides and videos (click here) (https://drive.google.com/drive/folders/1Dx3v7fD1QBWL-MNP2hd7iIxaRbeALkkA)](https://drive.google.com/drive/folders/1Dx3v7fD1QBWL-MNP2hd7iIxaRbeALkkA)