In [ ]:
```python
# decorator
def d(func):
    def wrapper():
        print('before')
        func()
        print('after')
    return wrapper


def f():
    print('hello')


x = d(f)
x()      # before hello after

print('----')

def d(func):
    def wrapper():
        print('before')
        func()
        print('after')
    return wrapper

@d
def f():
    print('hello')


f()      # before hello after
```

In [ ]:
```python
def d(func):
    def wrapper():
        print('before')
        func()
        func()
        print('after')
    return wrapper

@d
def f():
    print('python')


f()      # before  python python  after
```

In [ ]:
```
def d(func):
    def w(a):
        func(a + 3)
    return w

@d
def g(x):
    print(x)

g(5)  # 8
```

In [ ]:
```
'''
instance method:
      def f(self, a):
          pass

class method:
      @classmethod
      def g(cls, a):
         pass

static method:
      @staticmethod
      def h(a):
          pass
'''
```

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

    def f(self, i):
        print(self.a)
        return (i+3)

ob = C(1)
print(ob.a)       # 1
print(ob.f(3))    # 1  6

# print(C.a)       # error
# print(C.f(3))    # error
```

In [ ]:
```python
class A :
    def __init__(self, x):
        self.x = x

    @staticmethod
    def func_sum(m, n):
        print(m + n)
        # print(self.x)  error

A.func_sum(2, 3)   # 5

ob = A(8)
ob.func_sum(2, 3) # 5

print(A.func_sum == ob.func_sum )  # True
```

In [ ]:
```python
class D :
    def __init__(self, x):
        self.x = x

    @classmethod
    def h(cls, t):
        print(t + 2)
        return cls(t)

ob = D(0)
ob.h(2)    # 4
D.h(2)     # 4

print(D.h == ob.h )  # True

print(ob.h(15).x)   # 17  15
```

In [ ]:
```python
from datetime import date
class C:
    def __init__(self,name, age):
        self.name = name
        self.age = age

    @classmethod
    def f(cls,name, year):
        y = date.today().year - year
        return cls(name, y)

    @staticmethod
    def s(age):
        return age < 50

ob = C.f('Amin',1972)
print(ob.age)                  # 48
print(ob.name)                 # Amin

print(ob.s(48))                # True
```

In [ ]:

```python
class Date:
    def __init__(self, day=0 , month=0, year=0):
        self.day = day
        self.month = month
        self.year = year

    @classmethod
    def f(cls,d):
        day, month, year = map(int, d.split('-'))
        return cls(day,month,year)

    @staticmethod
    def g(d):
        day, month, year = map(int, d.split('-'))
        return day <= 31  and month <= 12


d = Date.f('11-09-2020')
print(d.year)             # 2020
print(d.month)            # 9
print(d.day)              # 11


v = Date.g('11-09-2020')
print(v)                  # True

v = Date.g('11-40-2020')
print(v)                  # False
```

In [ ]:

```python
class Person:
    def __init__(self, name, family):
        self.name = name
        self.family = family

    @property
    def f(self):
        return "%s %s" %(self.name , self.family)

    @f.setter
    def f(self, s):
        name , family = s.split(' ')
        self.name = name
        self.family = family

ob = Person('sara','rasti')
print(ob.f)                 # sara rasti

ob.f= 'amin golzari'
print(ob.name)            # amin
print(ob.family)          # golzari
```

In [ ]:
```
class Test:
     def f(k):
         return k.__name__
     f = classmethod(f)

print(Test.f())      # Test
```

In [ ]:
```
class  Numbers:
    a = 3

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

    def add(self):
        return self.x + self.y

    @classmethod
    def mul(cls, b):
        return cls.a * b

    @staticmethod
    def sub(b, c):
        return b - c

    @property
    def value(self):
       return(self.x, self.y)

    @value.setter
    def value(self, t):
        self.x, self.y = t

    @value.deleter
    def value(self):
        del self.x
        del self.y


ob = Numbers(2, 3)
print(ob.add())      # 5
print(ob.mul(4))     # 12
print(ob.sub(2,3))   # -1

print(ob.value)     # (2, 3)
ob.value = (6, 8)
print(ob.value)     # (6, 8)
```

In [ ]:
```
def d(func):
    def w():
        '''hello'''
        print(func.__name__)
        return func()
    return w

@d
def f(x):
    '''python'''
    return x+2

print(f.__name__)   # w
print(f.__doc__)    # hello
```

In [ ]:
```
from functools import wraps

def d(func):
    @wraps(func)
    def w():
        '''hello'''
        print(func.__name__)
        return func()
    return w

@d
def f(x):
    '''python'''
    return x+2

print(f.__name__)   # f
print(f.__doc__)    # python
```

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

    def f(self):
        return self.a + 2

class D(B):
      pass

ob = D(3)
print(ob.f())    # 5
```

In [ ]:

```python
from abc import ABC, abstractmethod

class B(ABC):
    def __init__(self, a):
        self.a = a
        super().__init__()

    @abstractmethod
    def f(self):
       pass


class D(B):
     def f(self):
        return self.a + 2

class E(B):
     def f(self):
        return self.a + 3

ob = D(3)
print(ob.f())    # 5

ob2 = E(4)
print(ob2.f())   # 7
```

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) slides and videos (click here) (https://drive.google.com/drive/folders/1Dx3v7fD1QBWL-MNP2hd7iIxaRbeALkkA)