In [1]:
```
class Myclass():
    def _______(self):
        return "Myclass"

a = Myclass()
print(a)
```

```
Myclass
```

In [2]:
```
def fun(s:str, ss:str=________):
    print (s, ss)

fun(12)
```

```
12 hello
```

In [3]:
```
class Myclass():
    name = "Li"
    def __init__(self, name):
        self.name = name

    def say_hello(self, name):
        print("Say hello to  %s." % name)

a = Myclass("Wei")
a.say_hello(________)

```

```
Say hello to  Li.
```

In [4]:
```
class A:
    def method(self):
        print("I am in parent")

class B(A):
    def method(self):
        print("I am in child.")
        ___.method(self)

b = B()
b.method()
```

```
I am in child.
I am in parent
```

In [5]:
```
class B(A):
    def method(self):
        print("I am in child.")
        ______().method()

b = B()
b.method()
```

```
I am in child.
I am in parent
```

In [6]:
```
class Myclass():
    def say_hello(self):
        print("hello.....")

a = Myclass()
Myclass.say_hello(____)
```

```
hello.....
```

In [7]:
```
class Myclass():
    @____________
    def class_say_hello(cls, s):
        print("class method ", s)

Myclass.class_say_hello("NEU2")
```

```
class method  NEU2
```

In [8]:
```
def myrepr(cls):
    cls.__repr__ = lambda self:super(cls, self).__repr__()[10:15]
    return cls

@_______
class classwithlonglonglongname():
    pass

c = classwithlonglonglongname()
print(c)
```

```
class
```

In [3]:
```
import copy
L = [1,2,3]
LC = L[:]
L2 = copy.copy(L)
print(LC is L2, LC == L2)
```

```
_______ _______
```

In [10]:
```
class A():
    def __eq__(self, other):
        return type(self) _____ type(other)

print(A() == 12)
```

```
False
```

In [11]:

```
class B(object):
    def __init__(self, x):
        self.__x = x

    def getx(self): return self.__x
    def setx(self, value): self.__x = value
    def delx(self): del self.__x

    x = ________(getx, setx, delx,  "x property")

b = B(10)
print(b.x)
b.x = 100
print(b.x)

```

```
10
100
```

In [12]:

```
class Myclass(object):
    __slots__ = ("say_hello", "data")

a = Myclass()
a.data = 12

def say_hello(obj):
    print("hello in outside {0.data}".format(obj))

a.say_hello = say_hello
a.say_hello(_____)

```

```
hello in outside 12
```

In [13]:

```
class A:
    pass

a = A()
print(________(a, A), isinstance(A, ________))
```

```
True True
```

In [14]:

```
def mymethod(self):
    return "hello in mymethod"

klass = type("Myclass", (object,), {________: mymethod})
inst = klass()
inst.method()
```

Out[14]: 'hello in mymethod'

In [15]:

```python
import abc

class A(abc.ABC):
    @abc.abstractmethod
    def a(self):pass

class B(A):
    def _____(self):
        print("hello in child class")

B().a()
```

```
hello in child class
```