

Множественное наследование. Шпаргалка

Абстрактный класс и абстрактный метод

```
from abc import ABC, abstractmethod

class Employee(ABC):

    @abstractmethod
    def work(self):
        pass

class Developer(Employee):

    def work(self):
        print("Пишет код")

class Accountant(Employee):

    def work(self):
        print("Считает зарплату")
```

Миксины

```
class Employee:

    def __init__(self, first, last):
        super().__init__()
        self.first = first
        self.last = last

class MixinLog:
    ID = 1

    def __init__(self):
        self.id = self.ID
        MixinLog.ID += 1

    def order_log(self):
        print(f'{self.id}-й сотрудник')

class Developer(Employee, MixinLog):
    pass

dev1 = Developer('Ivan', 'Ivanov')
dev1.order_log()

dev2 = Developer('Elena', 'Ivanova')
dev2.order_log()
```

Использование `__slots__` и сравнение с обычным классом

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def get_set_del(self):
        self.x += 1
        del self.y
        self.y = 0

class PointSlots:
    __slots__ = ('x', 'y')

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

    def get_set_del(self):
        self.x += 1
        del self.y
        self.y = 0

pt = Point(10, 20)          # Без __slots__
pts = PointSlots(10, 20)    # Co __slots__

import timeit

t1 = timeit.timeit(pt.get_set_del)    # Без __slots__
t2 = timeit.timeit(pts.get_set_del)   # Co __slots__

print(t1, t2)
print((t1-t2)/t1*100)
```