面向对象编程 Object-Oriented Programming(OOP)

OOA　OOD　OOP
分析　设计　编程

编程思想：

面向过程:

1 分析出解决问题的步骤，然后逐步解决。

例如：筹办婚礼

1、发请柬(贴照片 设计邀请语 制作请柬 发送请柬)

2、宴席 （找厨师 购买食材 设计菜品 准备餐具 桌椅板凳）

3、婚礼仪式 （找主持人 设计仪式流程 有哪些环节

2 公式：程序 = 算法 + 数据结构

3 优点：所有环节、细节自己掌控

4 缺点：面面俱到 要考虑所有细节、工作量大

面向对象:

1 指定解决问题的人，分配工作

例如：筹办婚礼

1、发请柬--->摄影公司 （对象）

2、宴席----->酒店（对象）

3、婚礼仪式->婚庆公司 （对象）

2 公式：程序 = 对象 + 交流（交互）

3 优点：分类解决问题 各司其职 、高复用

概念

类:是抽象的概念，即人类社会里的类别（具有相同属性和行为的 视为 同类）

属性=数据 行为=方法

对象：类的具体实例，归属某个类别的个体

类是创建对象的“模板”类里包括：个体的属性 和 个体的行为
数据 方法

- 类

> *拥有相同属性和行为的对象分为一组，即为一个类*

```
      /-------> BYD   E6(京A.88888)     实例(也叫对象)
车(类)
      \-------> BWM   X5(京B.66666)     实例(也叫对象)

       /-------> 100     (对象)
int(类)
       \-------> 200     (对象)

        /------> 张三    (对象)
人(类)
        \------> 李四    (对象)
```

创建类、创建对象

```
class person():
        money = '1亿'
        def run(self):
                print("今天跑步 5公里")

plj = person()
plj.run()
print(plj.money)
```

单继承

```
class shifu(): #师傅类
        def make_cake(self):
                print("独门配置 百年传承")

class tudi(shifu): #徒弟类
        pass

xiaojiu = tudi()  #创建徒弟对象
xiaojiu.make_cake() #可以调用父类的方法
```

多继承

```
class bb(): #爸爸类
        def car(self):
                print("BMW")

class mm(): #妈妈类
        def fly(self):
                print("私人飞机1架")

class erzi(bb,mm): #儿子类
        pass

daerzi = erzi()  #创建儿子对象
#调用基类方法
daerzi.fly()
daerzi.car()
```

魔法方法:具体特殊功能的函数,函数名都是事先定义好的

```
魔法方法 _ _init_ _()  #作用初始化对象，创建对象时默认调用
class person():

        def __init__(self):
                self.gender = 'boy'
                self.age = 1

        def myinfo(self,name):
                print(f"我叫{name} 今年{self.age} 是位{self.gender}")

        money = '1亿'
        def run(self):
                print("今天跑步 5公里")

plj = person()
plj.myinfo("plj")


#可以传参的
class person():

        def __init__(self,gender,age):
                self.gender = gender
                self.age = age

        def myinfo(self,name):
                print(f"我叫{name} 今年{self.age} 是位{self.gender}")

        money = '1亿'
        def run(self):
                print("今天跑步 5公里")

plj = person("girl",20)
plj.myinfo("plj")

魔法方法 _ _str_ _() #作用输出return 的返回值
默认输出对象的内存地址
plj = person("girl",20)
print(plj) #

[root@localhost ~]# python3 oop.py
```

```
<__main__.person object at 0x7fe8cf7ad400>

]# vim oop.py
class person():
        def __init__(self,gender,age):
                self.gender = gender
                self.age = age
        def __str__(self):
                return "我是person类的对象"

plj = person("girl",20)
print(plj)
:wq
[root@localhost ~]# python3 oop.py
我是person类的对象
[root@localhost ~]#
```

魔法方法 _ _call_ _() 将实例当成函数来调用，执行此函数内的代码

```
vim oop.py
class person():
        def __init__(self,gender,age):
                self.gender = gender
                self.age = age

        def __call__(self):
                print(self.age,self.gender)

plj = person("girl",20)
plj()
:wq

[root@localhost ~]# python3 oop.py
20 girl
[root@localhost ~]#
```

魔法方法 _ _del_ _() 删除对象时自动调用del() 方法

```
vim oop.py
class person():
        def __init__(self,gender,age):
                self.gender = gender
                self.age = age
         def __del__(self):
```

```
                print("对象被成功删除")
plj = person("girl",20)
del plj
:wq

[root@localhost ~]# python3 oop.py
对象被成功删除
```