

# BDA\_Ch2 part 1

2019 年 9 月 25 日

```
In [ ]: print('hello world')
```

```
In [ ]: word_count = 711
        if word_count > 800:
            print("字数达到要求")
        else:
            print("字数不达标")
```

```
In [ ]: # 封装
```

```
class Student:
    name = '' #name 属性是公开的
    __score = -1 #score 属性前面有 __ 符号，表示是私有属性

    # 初始化方法，在类实例化的时候会首先调用这个方法
    def __init__(self, name, score):
        self.name = name
        self.__score = score

    # 由于 score 是私有属性，因此需要通过类内部的方法来进行访问
    def get_score(self):
        return self.__score

    def is_qualified(self):
        if self.__score > 90:
            print('优秀')
        else:
            print('继续努力')
```

```
s1 = Student('张三', 85)
s1.is_qualified()
print('学生姓名为: ' + s1.name) #name 属性是公开的，所以可以直接访问
s1.name = '张麻子'
print('学生姓名改为: ' + s1.name) # 甚至可以直接改名字
print('学生成绩为: ' + str(s1.get_score())) #score 属性是私有的，可以通过类内部的方法来调用访问
print('学生成绩为: ' + str(s1.__score)) # 因为无法访问 score 属性，所以会报错
# 但是在 Python 中并没有办法真正限制，如下方式就可以直接调用（格式为：对象._ 类 __ 属性名）
print('学生成绩为: ' + str(s1._Student__score))
s1._Student__score = 90 # 修改值也没问题
print('学生成绩为: ' + str(s1._Student__score))
```

In [ ]: # 继承

```
class Student:
    _name = '' #name 前面加 _, 类和子类都可以调用

    def __init__(self, name):
        self._name = name

    def workday_act(self):
        print(self._name + '工作日上课')

    def weekend_act(self):
        print(self._name + '周末休息')

class UGStudent(Student):
    # 重写父类的方法
    def weekend_act(self):
        print(self._name + '周末出去玩')

class PhD(Student):
    # 重写父类的方法
    def weekend_act(self):
        print(self._name + '作为博士生, 周末看文献、跑数据当作休息')

ug1 = UGStudent('张三')
ug1.workday_act()
ug1.weekend_act()
phd1 = PhD('李四')
phd1.workday_act()
phd1.weekend_act()
```

In [ ]: # 多态

```
class PhD:
    _name = ''

    def __init__(self, name):
        self._name = name

    def research(self):
        pass

    @staticmethod # 静态方法, 需要通过类名来调用这个方法
    def phd_research(obj):
        obj.research()

class MathPhD(PhD):
```

```
def research(self):
    print(self._name + '正在推公式')

class ChemicalPhD(PhD):
    def research(self):
        print(self._name + '正在刷试管')

class ManagementPhD(PhD):
    def research(self):
        print(self._name + '正在编故事')

phd1 = MathPhD('李四')
phd2 = ChemicalPhD('王五')
phd3 = ManagementPhD('小六')

PhD.phd_research(phd1)
PhD.phd_research(phd2)
PhD.phd_research(phd3)
```