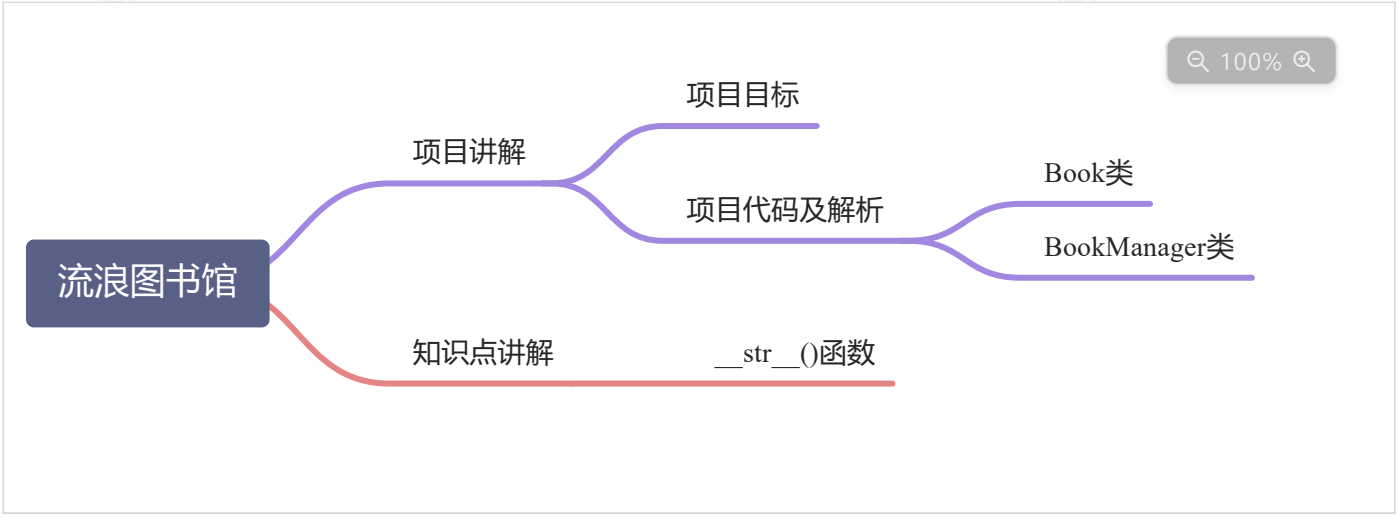
第14课 流浪图书馆

一、课程结构导图



二、项目讲解

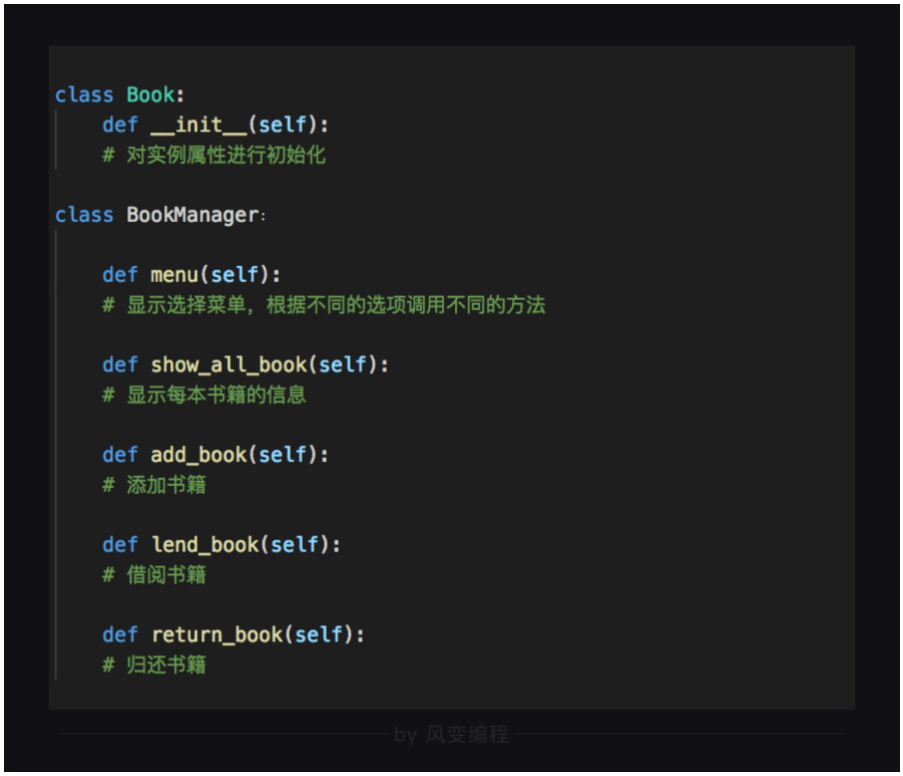
2.1 项目目标

项目目标：编写一个图书管理系统的程序。
实现功能：该程序可以对系统中的书籍进行查询、添加、借阅、归还这四步操作。

查询书籍：可以一键查询系统里所有书籍的基本信息和借阅状态。
添加书籍：往系统添加书籍时，需要输入书籍的基本信息（书名、作者、推荐语）
借阅书籍：当书籍的借阅状态是“未借出”的时候，书籍才可出借，借出后状态变成“已借出”。
归还书籍：归还成功后书籍的借阅状态会更改成“未借出”，可再次被借阅。

2.2 项目代码及解析

根据项目目标，可以把代码的框架梳理如下：



下面解析一下为什么要使用两个类?

Book 类就相当一个模型，而BookManager 类是一个控制器，通过BookManager 类来调用控制Book 类。

在操作的过程中，需要通过Book 类的实例对象调用相关的实例属性进行对比等操作，比如：借书的时候，需要通过输入的书名在“书库”中的查找是否有同名书籍。使用Book 类的实例对象，便可以直接调用实例属性name 和输入的书名进行匹配。

2.2.1 Book类

```
1 class Book:
2
3     def __init__(self, name, author, comment, state = 0):
4         self.name = name
5         self.author = author
```

```
        self.comment = comment
        self.state = state
    def __str__(self):
        status = '未借出'
        if self.state == 1:     # 当state 为1 时，修改借出状态status 为“已借出”
            status = '已借出'
        return '名称：《%s》 作者：%s 推荐语：%s\n状态：%s ' % (self.name,
self.author, self.comment, status)
```

代码解析：由于初始方法使用return 语句只能返回空值，不能返回字符串，所以不能把第12行直接放到初始方法中，需要另外创建一个方法来返回数据，如果是采用普通的方法，需要再进一步的调用才能执行，而__str__() 方法同初始方法，调用类的时候，便可自动执行，并返回return 的内容，不需要进一步的调用。

2.2.2 BookManager类

说明：以下代码把每一块的内容拆开，单独运行可能会报错，需要把本课所有代码串起来再运行。

1. **初始方法**

```
class BookManager:
    books = []      # 存储书籍的列表，每一个元素都是Book的实例对象
    def __init__(self):
        # 实例调用Book 类，state为默认参数，默认未借出，如无修改必要可以不用传递
        book1 = Book('惶然录', '费尔南多·佩索阿', '一个迷失方向且濒于崩溃的灵魂的自我启
示，一首对默默无闻、失败、智慧、困难和沉默的赞美诗。')
        book2 = Book('以箭为翅', '简媜', '调和空灵文风与禅宗境界，刻画人间之缘起缘灭。像
一条柔韧的绳子，情这个字，不知勒痛多少人的心肉。')
        book3 = Book('心是孤独的猎手', '卡森·麦卡勒斯', '我们渴望倾诉，却从未倾听。女孩、
黑人、哑巴、醉鬼、鳏夫的孤独形态各异，却从未退场。', 1)      # 传入借出状态为1，即“已借
出”。

        # 将Book 类的3个实例添加到books 列表中
        self.books.append(book1)
        self.books.append(book2)
        self.books.append(book3)
```

代码解析：调用Book 类，将返回值分别赋值给book1 、book2 、book3，并添加到列表books 中。
注意，将对象添加到列表中时，是添加对象本身，即Book 类的实例对象，而不是return 语句后面的字符串。
Book 类的实例对象，才具有Book 类中的属性和方法，在检查书籍、借阅书籍、归还书籍进行调用时，才可以调用相关的属性、方法进行相关的判断和修改。

2. 菜单 menu()

```
def menu(self):
    print('欢迎使用流浪图书管理系统，每本沉默的好书都是一座流浪的岛屿，希望你有缘发现并着陆，为精神家园找到一片栖息地。\n')
    while True:
        print('1.查询所有书籍\n2.添加书籍\n3.借阅书籍\n4.归还书籍\n5.退出系统\n')
        choice = input('请输入数字选择对应的功能：')
        if choice == '1':
            self.show_all_book()    # 功能1：调用show_all_book() 打印所有书籍
        elif choice == '2':
            self.add_book()         # 功能2：调用add_book() 添加书籍
        elif choice == '3':
            self.lend_book()        # 功能3：调用lend_book() 借阅书籍
        elif choice == '4':
            self.return_book()      # 功能4：调用return_book() 归还书籍
        elif choice == '5':
            print('感谢使用！愿你我成为爱书之人，在茫茫书海里相遇。')
            break                   # 功能5：跳出循环，结束程序
        else:                       # 当输入不是1~5这5个数字时，提示输出有误
            print('不能识别指令，重启程序！')
```

代码解析：定义菜单方法，设定五个指令供用户选择，五个指令对应五种不同的操作。另外，当用户输错时，则会执行else语句，提示错误。

3. 查询所有书籍 show_all_book()

```
def show_all_book(self):
    print('书籍信息如下：')
    for book in self.books:     # 遍历所有书籍
```

```
        print(book)
```

代码解析：书籍存放在books 列表中，故查询所有书籍就是遍历books 列表，打印出每一个元素。

4. **添加书籍 add_book()**

```
def add_book(self):
    new_name = input('请输入书籍名称：')
    new_author = input('请输入作者名称：')
    new_comment = input('请输入书籍推荐语：')
    new_book = Book(new_name, new_author, new_comment)  # 实例化调用Book 类，创建实例对象new_b
    self.books.append(new_book)     # 将实例对象new_book 添加到books 列表中
    print('书籍录入成功！\n')
```

代码解析：在Book 类中，包含书籍的书名、作者、书评和是否借出信息。因新添加的书籍，是未借出状态，所以借阅状态可设置为默认值，因此只需添加书名、作者、书评三个信息即可。使用这三个信息，调用Book 类创建一个新的实例对象，并添加到books列表中。

5. **检查书籍 check_book()**

```
def check_book(self, name):
    for book in self.books: # 遍历列表的每个元素，即每个Book实例
        if book.name == name:   # 如果存在有实例名称与输入书籍名称是一样的
            return book  # 返回该实例对象，遇到return语句方法停止执行
    else:               # 若for循环中，没有返回满足条件的对象，则执行else子句
        return None
```

代码解析：

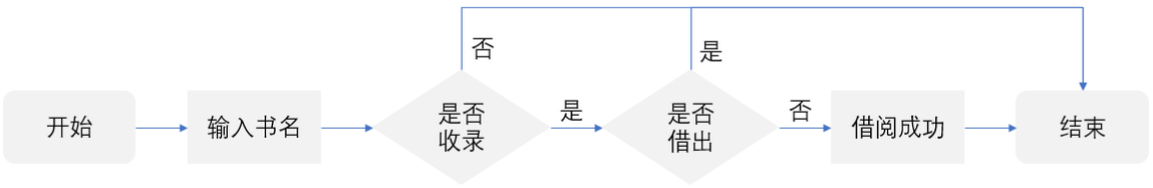
1. 借书和还书的时候，需要先对书库的内容进行检索，判断书籍是否在“书库”中。所以把检查书籍步骤单独提取出来，提高代码的复用性。在这个“书库”里，书名是唯一的ID ，所以在检查时使用书名来查找。
2. 遍历books 列表，分别取出列表中的Book 类的实例对象，然后调用实例属性name 判断和需要检查的书籍名称是否一致，如果一致，则返回对应的实例对象，如果不满足，返回空值None 。

6. 借阅书籍 lend_book()

```
def lend_book(self):
    name = input('请输入书籍的名称：')
    res = self.check_book(name) # 调用check_book方法，并将返回值赋值给变量res

    if res != None:             # 如果返回值不等于None值，即返回的是实例对象
        if res.state == 1:      # 如果实例对象的属性state为1，即借阅状态为“已借出”
            print('你来晚了一步，这本书已经被借走了噢')
        else:
            print('借阅成功，借了不看会变胖噢～')
            res.state = 1       # 书籍借出后属性state变为1
    else:
        print('这本书暂时没有收录在系统里呢')
```

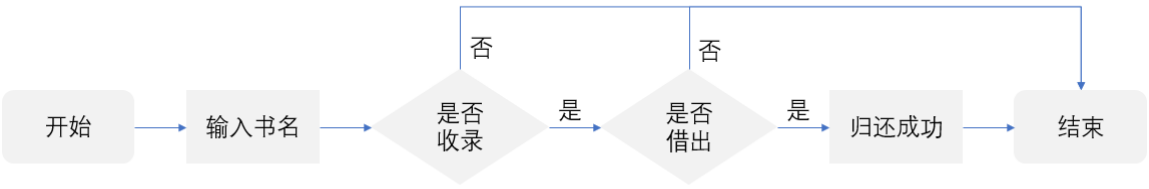
代码解析：借书的流程如下，首先输入书名，然后调用检查书籍的方法查看是否收录该书籍，有收录则判断借出状态state ，等于0则是未借出，所以可以借阅。如果没有收录，或者已借出则直接结束。



7. 归还书籍 return_book()

```
def return_book(self):
    name = input('请输入归还书籍的名称：')
    res = self.check_book(name) # 调用check_book方法，将返回值赋值给变量res
    if res == None:             # 如果返回的是空值，即这本书的书名不在系统里
        print('没有这本书噢，你恐怕输错了书名～')
    else:
        if res.state == 0:      # 如果实例属性state等于0，即这本书的借阅状态为“未借出”
            print('这本书没有被借走，在等待有缘人的垂青呢！')
        else:
            print('归还成功！')
            res.state = 0       # 归还后修改书籍借阅状态为0，即：“未借出”
```

代码解析：还书的流程如下，还书流程和结束流程差不多，差别在于，判断是否借出部分，借书是未借出可以借阅，而还书是已借出可以归还。



三、知识点讲解

3.1 __str__()函数

使用场景： 初始方法只能返回None，当需要返回字符串时，可以使用该方法。（注：两边有两个下划线的方法叫魔术方法。__init__() 和__str__() 都是魔术方法）。

用法： 使用__str__() 方法返回一个字符串，在打印实例对象时，会自动调用，并返回该字符串。

示例：

```
# 代码一：无__str__()
class Dog:
    def __init__(self):
        self.size = '小'

dog = Dog()
print(dog)
# 结果：<__main__.Ball object at 0x00000202AF84FA20>,0x00000202AF84FA20是所在的内存空间。

# 代码二：使用__str__()
class Dog:
    def __init__(self):
        self.size = '小型'
```

```
    def __str__(self):
        return '这条狗属于%s犬。'% self.size

dog = Dog()
print(dog)      # 结果：这条狗属于小型犬。
```