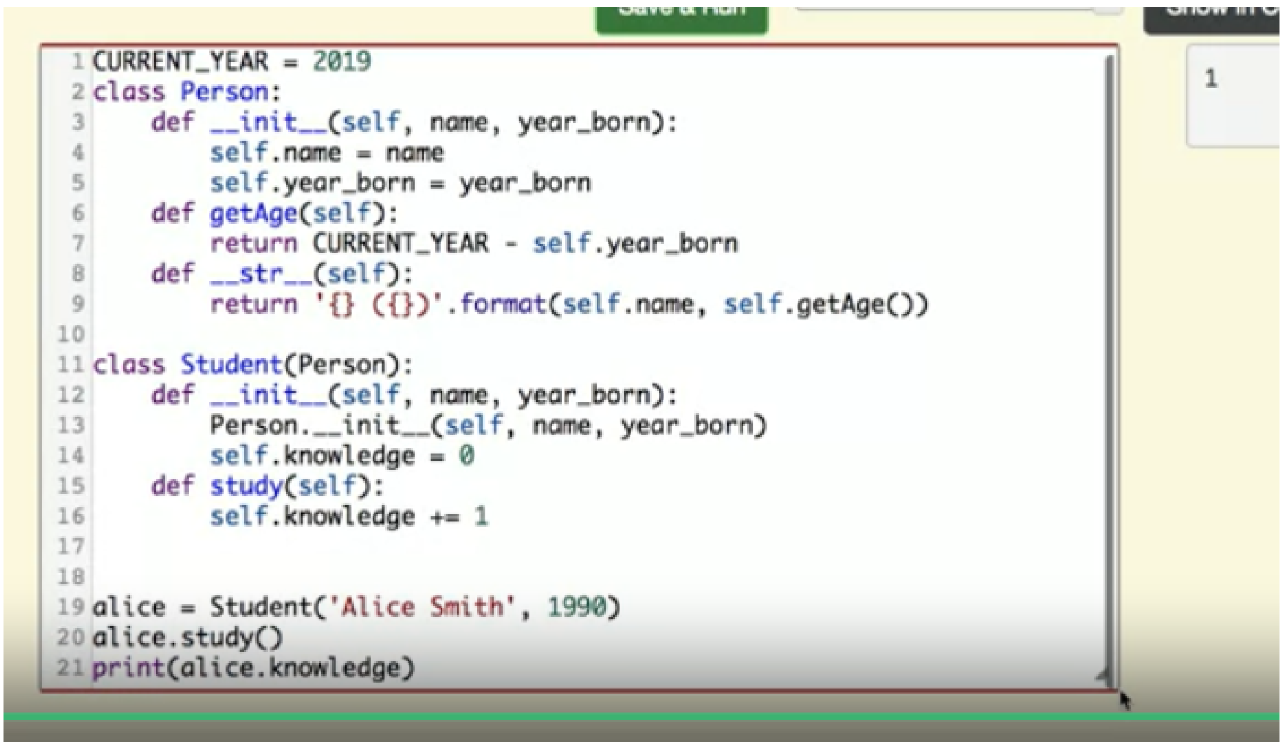
4.2 Inheritance

- 现在有两个class 一个是person一个是student 其实student是inherited from person的 所以相当于大类和小类 define了一个大类之后其他的可以不用那么麻烦
- 

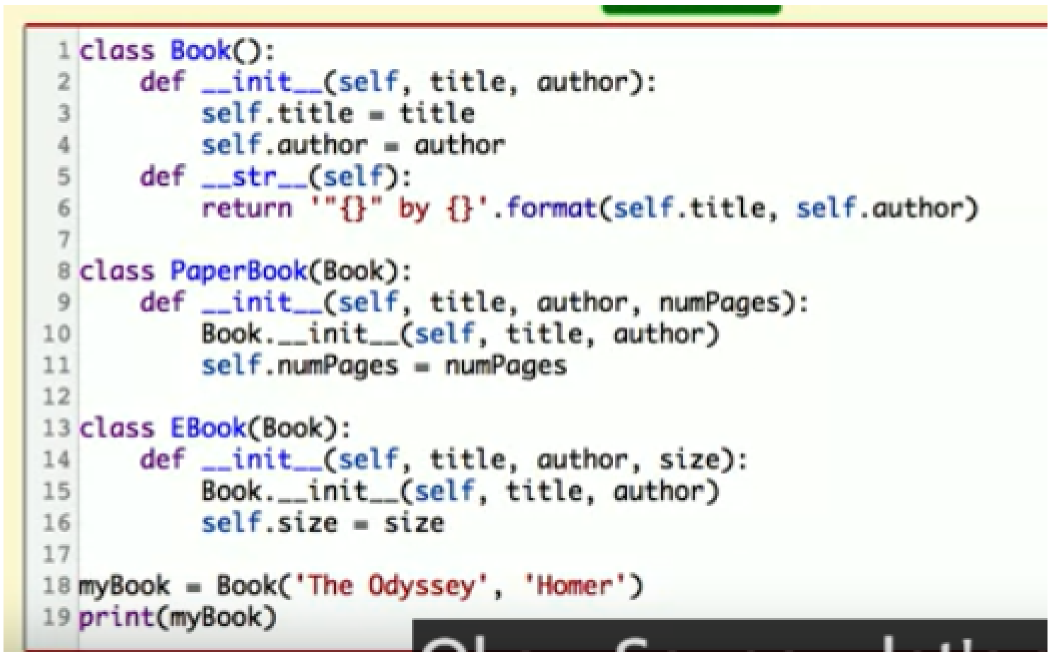
```
CURRENT_YEAR = 2019
class Person:
    def __init__(self, name, year_born):
        self.name = name
        self.year_born = year_born
    def getAge(self):
        return CURRENT_YEAR - self.year_born
    def __str__(self):
        return '{} ({})'.format(self.name, self.getAge())

class Student(Person):
    def __init__(self, name, year_born):
        Person.__init__(self, name, year_born)
        self.knowledge = 0
    def study(self):
        self.knowledge += 1


alice = Student('Alice Smith', 1990)
alice.study()
print(alice.knowledge)
```

```
1
```

  - student想要inherited from person的话 class的地方括号里就是要inherited的大class
  - 还有 命名和variable如果一样的话 就是def改成大class的名字.__init__()括号里是person的东西 然后student class可以有自己的特性
  - 就算有个人 用student去call 还是可以从person的性质中get到信息
  - subclass可以有一些自己的特性 做出相关的修改 比如这里的纸质书和ebook



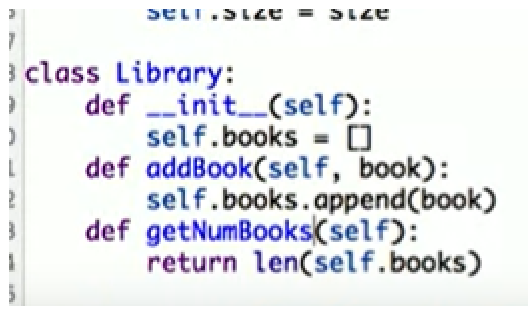
```
class Book():
    def __init__(self, title, author):
        self.title = title
        self.author = author
    def __str__(self):
        return '"{}" by {}'.format(self.title, self.author)

class PaperBook(Book):
    def __init__(self, title, author, numPages):
        Book.__init__(self, title, author)
        self.numPages = numPages

class EBook(Book):
    def __init__(self, title, author, size):
        Book.__init__(self, title, author)
        self.size = size

myBook = Book('The Odyssey', 'Homer')
print(myBook)
```

- 比如想说一个大的class可以包含好几类小class 这里只是contain而不是inherited——composition 因为create了一个list



```
        self.size = size

class Library:
    def __init__(self):
        self.books = []
    def addBook(self, book):
        self.books.append(book)
    def getNumBooks(self):
        return len(self.books)
```