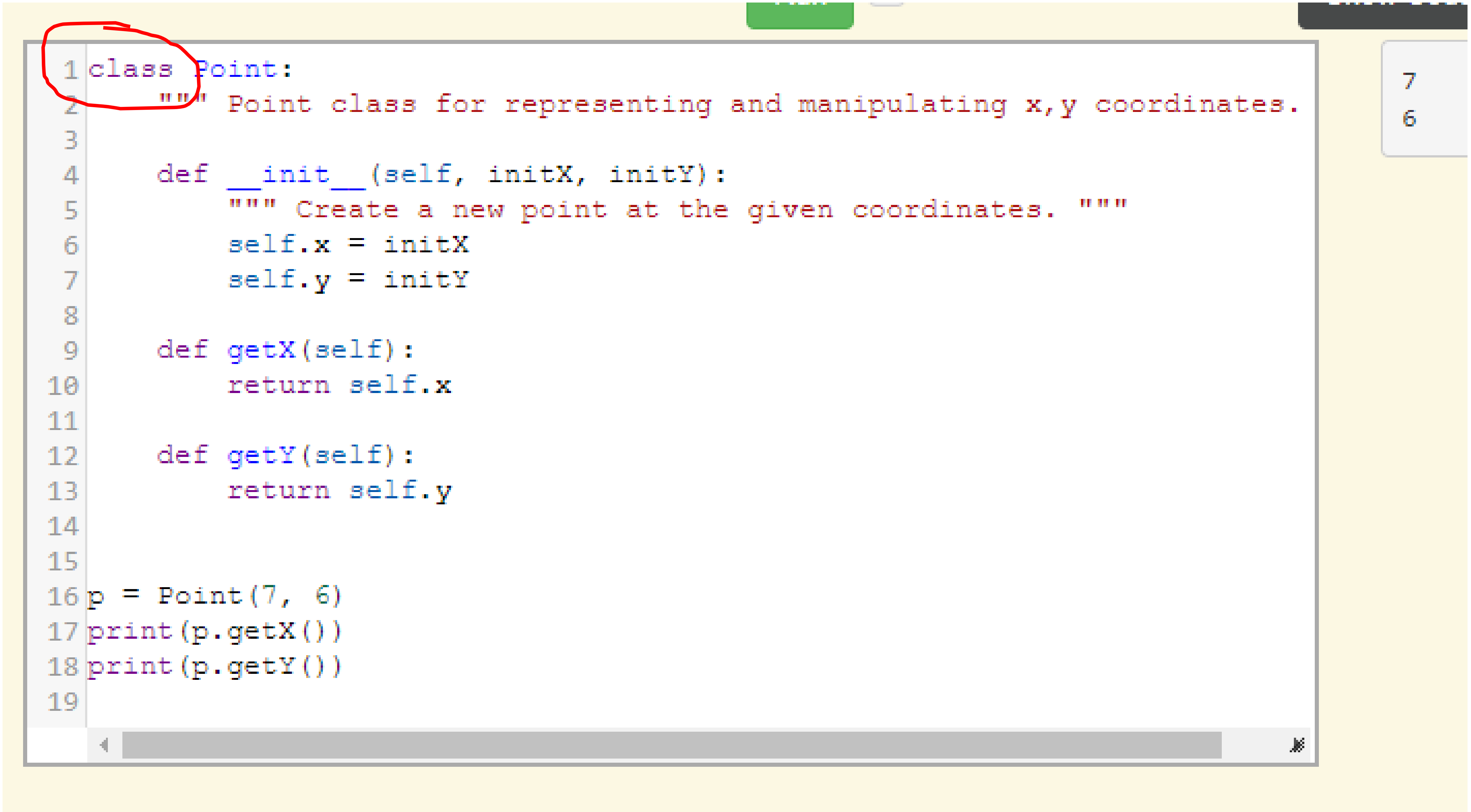
OOP continued

by Deborah R. Fowler

KEY CONCEPTS

- ✓ variables
- ✓ truth statements
- ✓ looping
- ✓ functions
- ✓ I/O
- ✓ lists
- ✓ classes/objects
- ✓ OOP

Recall from last day the basic structure



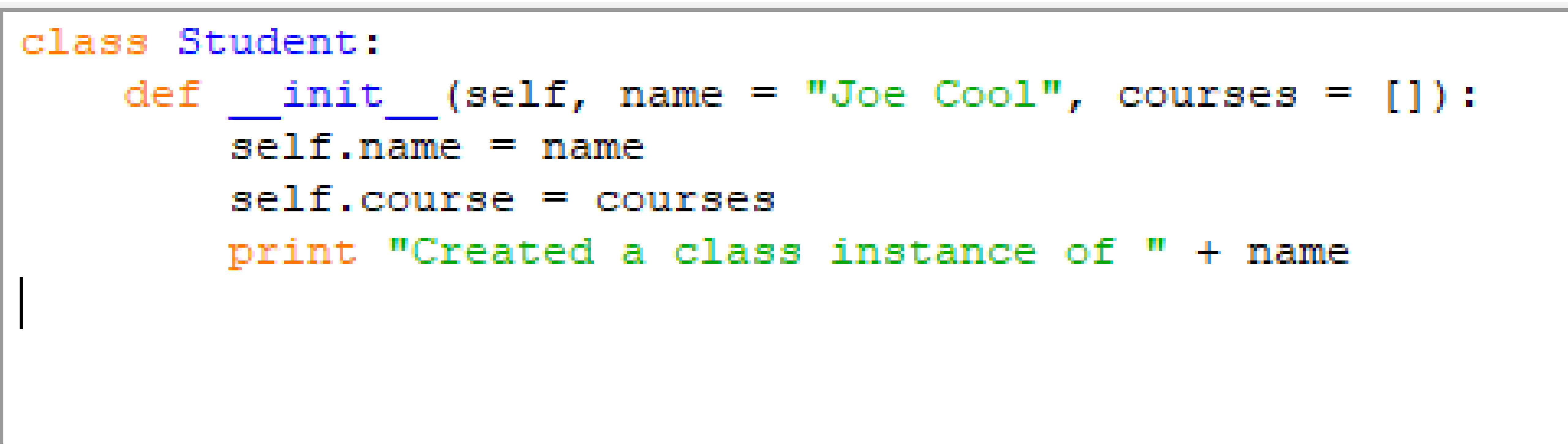
```
class Point:
    """ Point class for representing and manipulating x,y coordinates.

    def __init__(self, initX, initY):
        """ Create a new point at the given coordinates. """
        self.x = initX
        self.y = initY

    def getX(self):
        return self.x

    def getY(self):
        return self.y


p = Point(7, 6)
print(p.getX())
print(p.getY())
```

```
7
6
```

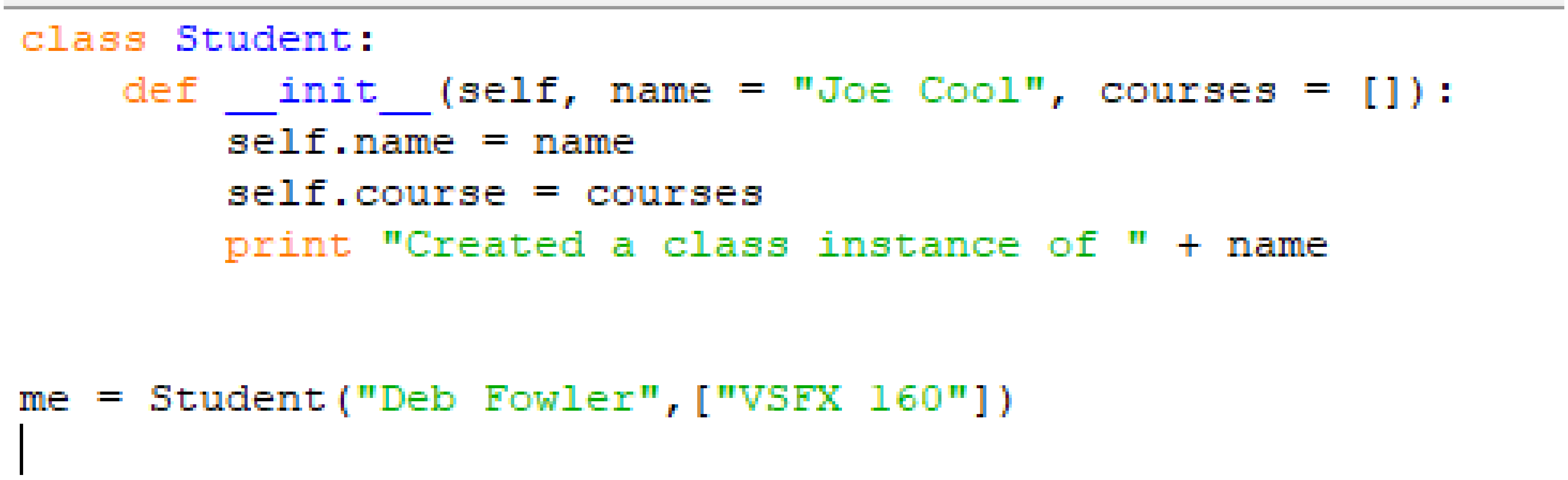
Suppose you wanted to keep track of students data

name, idnum, classes and so on

What would this look like?

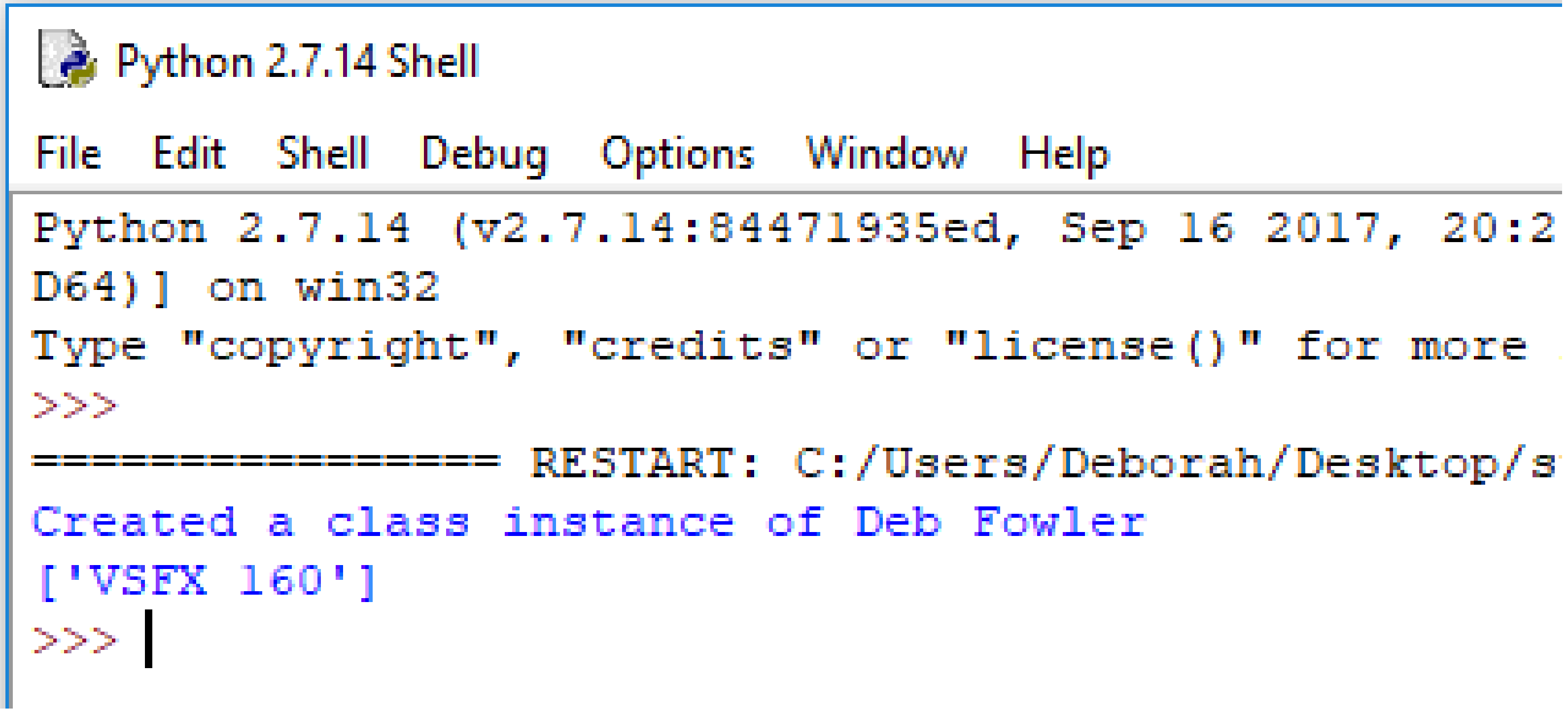


```
class Student:
    def __init__(self, name = "Joe Cool", courses = []):
        self.name = name
        self.course = courses
        print "Created a class instance of " + name
```

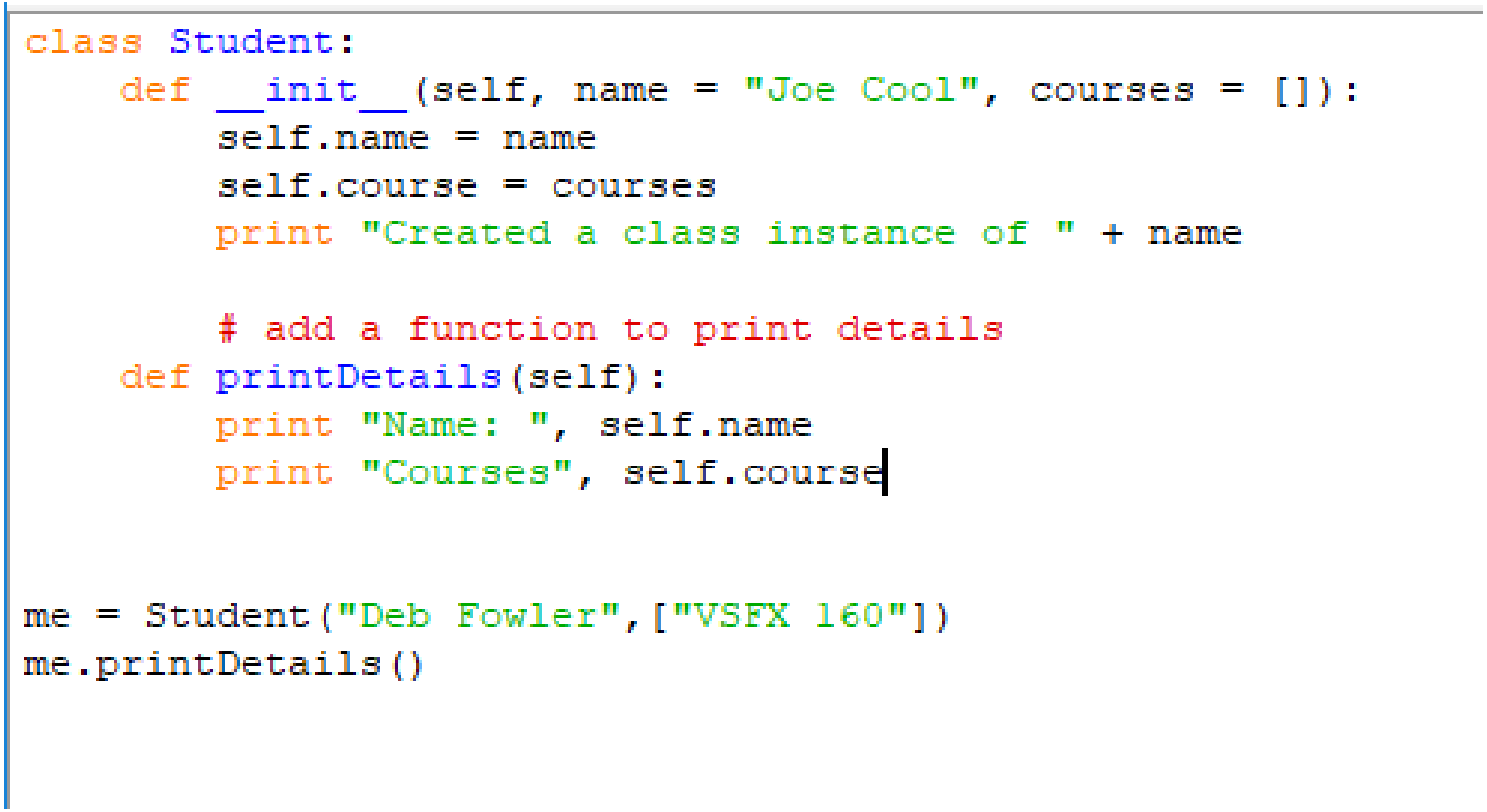


```python
class Student:
    def __init__(self, name = "Joe Cool", courses = []):
        self.name = name
        self.course = courses
        print "Created a class instance of " + name


me = Student("Deb Fowler",["VSFX 160"])
```

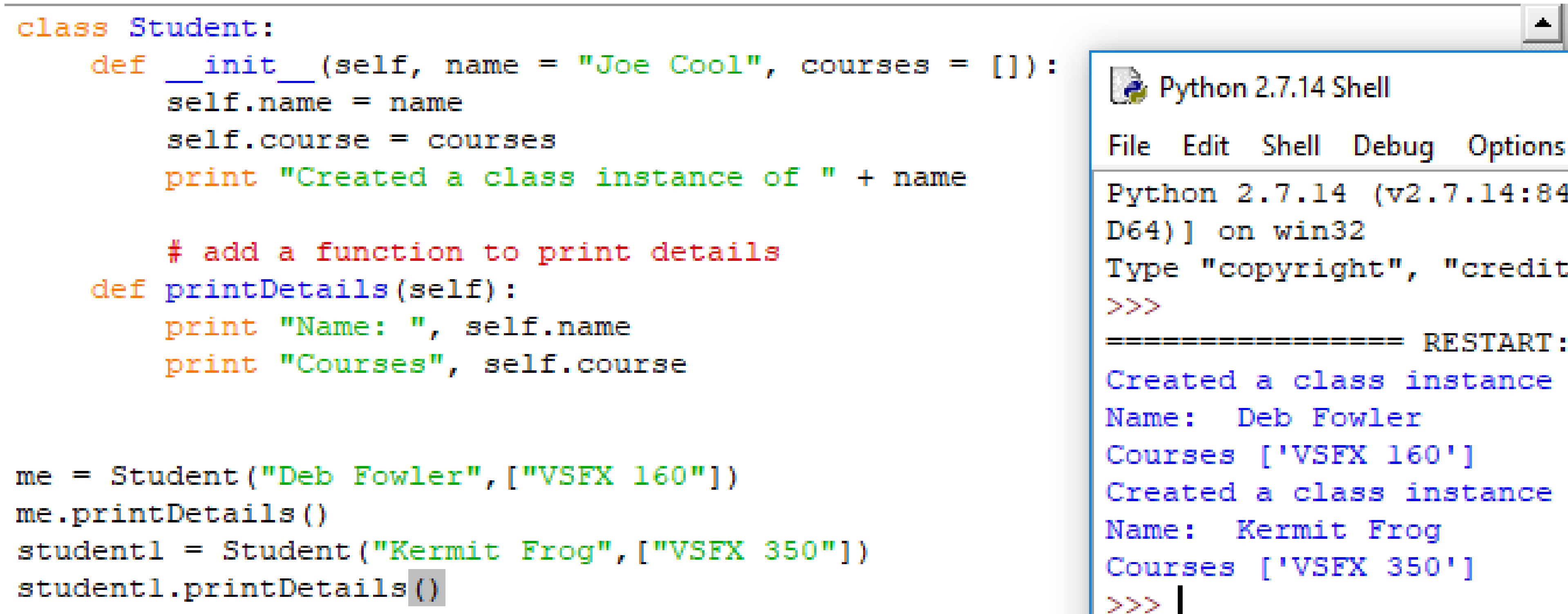
```
class Student:
    def __init__(self, name = "Joe Cool", courses = []):
        self.name = name
        self.course = courses
        print "Created a class instance of " + name


me = Student("Deb Fowler",["VSFX 160"])
print me.course
```



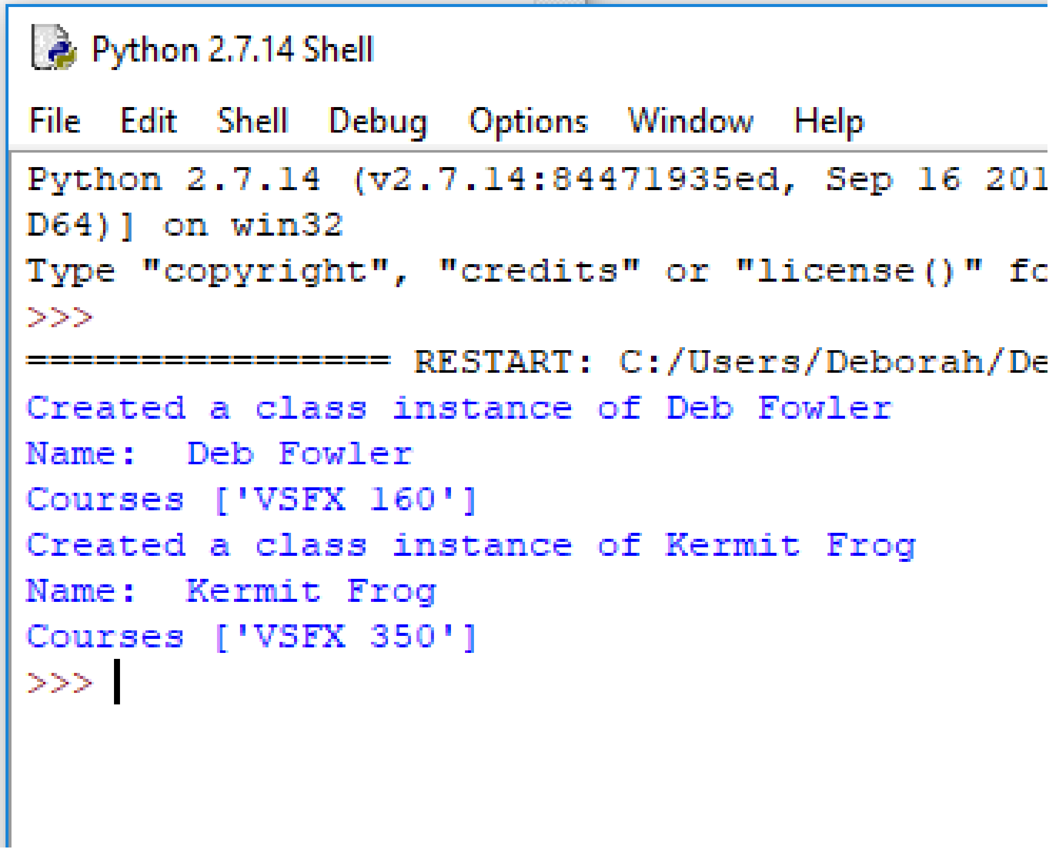
```
Python 2.7.14 (v2.7.14:84471935ed, Sep 16 2017, 20:2
D64)] on win32
Type "copyright", "credits" or "license()" for more
>>>
================= RESTART: C:/Users/Deborah/Desktop/s
Created a class instance of Deb Fowler
['VSFX 160']
>>>
```



```python
class Student:
    def __init__(self, name = "Joe Cool", courses = []):
        self.name = name
        self.course = courses
        print "Created a class instance of " + name

        # add a function to print details
    def printDetails(self):
        print "Name: ", self.name
        print "Courses", self.course


me = Student("Deb Fowler",["VSFX 160"])
me.printDetails()
```



```
class Student:
    def __init__(self, name = "Joe Cool", courses = []):
        self.name = name
        self.course = courses
        print "Created a class instance of " + name

        # add a function to print details
    def printDetails(self):
        print "Name: ", self.name
        print "Courses", self.course


me = Student("Deb Fowler",["VSFX 160"])
me.printDetails()
student1 = Student("Kermit Frog",["VSFX 350"])
student1.printDetails()
```



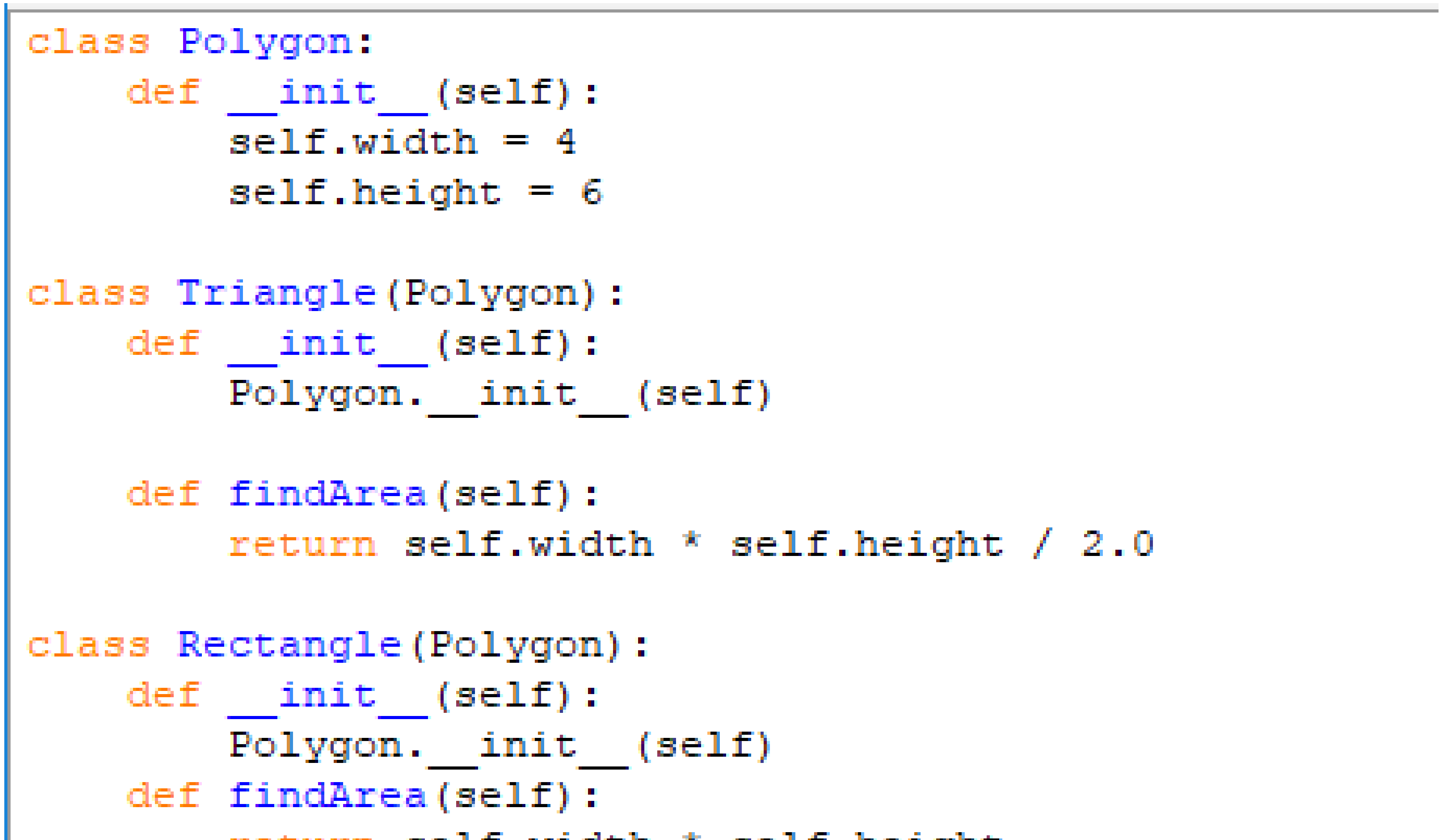
```
Python 2.7.14 (v2.7.14:84471935ed, Sep 16 201
D64)] on win32
Type "copyright", "credits" or "license()" fc
>>>
================ RESTART: C:/Users/Deborah/De
Created a class instance of Deb Fowler
Name:  Deb Fowler
Courses ['VSFX 160']
Created a class instance of Kermit Frog
Name:  Kermit Frog
Courses ['VSFX 350']
>>>
```

In-class Exercise

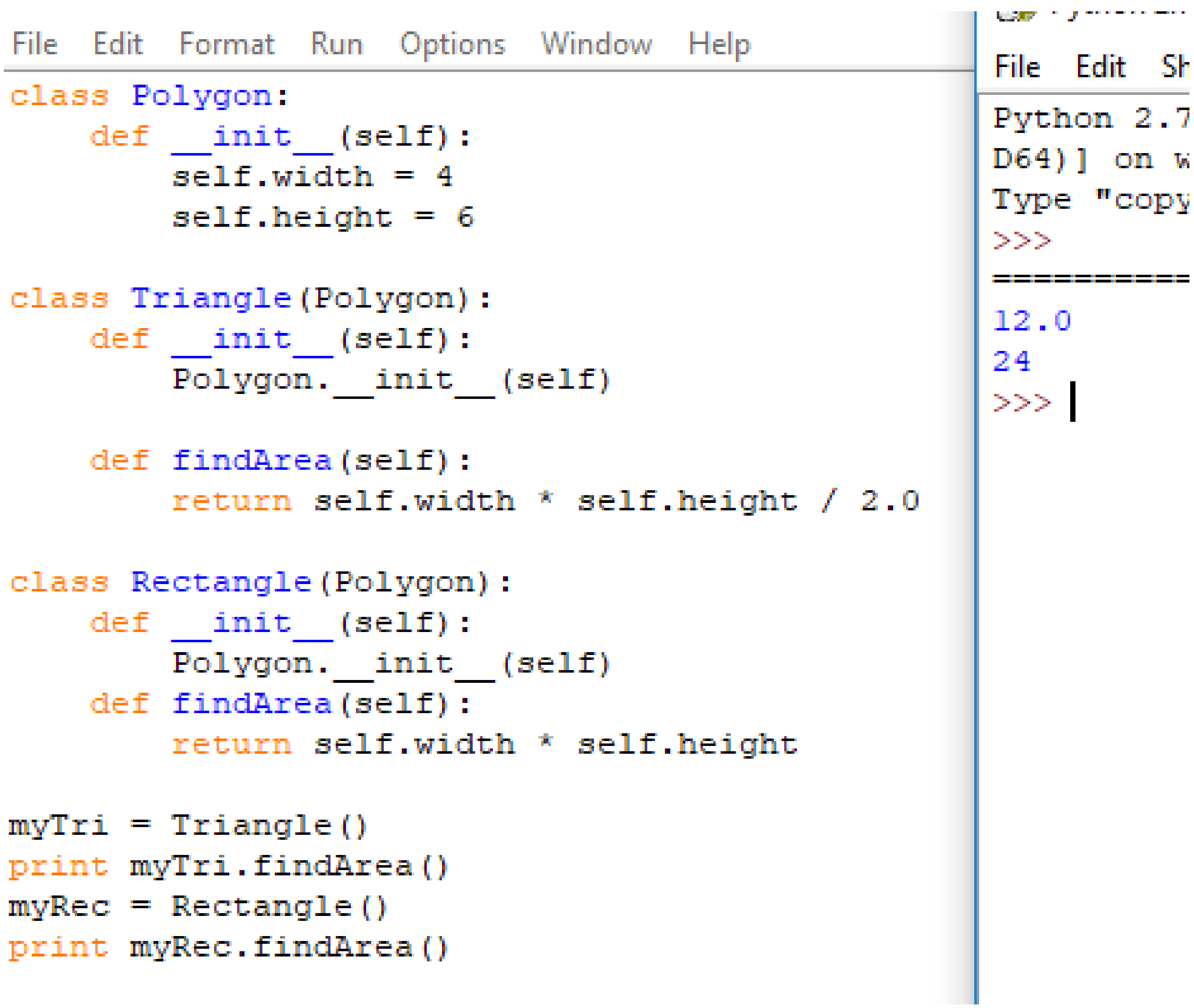
To become familiar with OOP you have a choice of two assignments. These will be completed in class.

1. OOP
2. OOP with Inheritance – defining classes to take the functionality of a parent class (so you have children of classes)

Example of Inheritance



```
class Polygon:
    def __init__(self):
        self.width = 4
        self.height = 6

class Triangle(Polygon):
    def __init__(self):
        Polygon.__init__(self)

    def findArea(self):
        return self.width * self.height / 2.0

class Rectangle(Polygon):
    def __init__(self):
        Polygon.__init__(self)
    def findArea(self):
        return self.width * self.height
```



```
class Polygon:
    def __init__(self):
        self.width = 4
        self.height = 6

class Triangle(Polygon):
    def __init__(self):
        Polygon.__init__(self)

    def findArea(self):
        return self.width * self.height / 2.0

class Rectangle(Polygon):
    def __init__(self):
        Polygon.__init__(self)
    def findArea(self):
        return self.width * self.height

myTri = Triangle()
print myTri.findArea()
myRec = Rectangle()
print myRec.findArea()
```

```
Python 2.7
D64)] on w
Type "copy
>>>
==========
12.0
24
>>>
```

In-class Exercise

See the links on the page for full descriptions

1. Start with the student Class we defined. Add

    grades

    average for the student

    letter grade for the student

To start, assume they only are in one course

In-class Exercise

Read in data from a file and print the students weighted grade average.

Get it working for one student, then expand it to many

In-class Exercise

Sample data:

Cool Joe

3

99 80 100 70

Brown Sally

1

80 99 70 100

In-class Exercise

Program should output

Student name and average