CMSC201 Computer Science I for Majors

Lecture 18 – Classes and Modules (Continued, Part 3)

Prof. Jeremy Dixon

Last Class We Covered

- Constructors
- Difference between
 - Data attributes
 - Class attributes
- Special built-in methods and attributes
- Creating and using a class

Any Questions from Last Time?

Today's Objectives

- Project 1 Considerations
- To harness the power of inheritance
 - To learn about subclasses and superclasses
 - To be able to redefine a method
 - To be able to extend a method
 - (Including ___init___)



Find the Errors in the Code Below

```
def student:
    def init(self, n, a, g):
        name = n
                                      There are at
        age = a
        qpa = q
                                      least seven
    def updateGPA(newGPA):
                                     unique errors
        qpa = newGPA
def main():
    val = new student("Alex", 21, 4.0)
    test = new student("Test", 18, 0)
    updateGPA(test, 3.26)
main()
```



Find the Errors in the Code Below

```
student:
    def(init(self, n, a, g):
        name =
    def updateGPA(newGPA):
        gpa = newGPA
def main():
         =(new)student("Alex", 21, 4.0)
    test = (new)student("Test", 18, 0)
    updateGPA (test, 3.26)
main()
```



Find the Errors in the Code Below

```
class student:
    def init (self, name, age, gpa):
        self.name = name
        self.age = age
        self.gpa = gpa
    def updateGPA(self, newGPA):
        self.gpa = newGPA
def main():
    val = student("Alex", 21, 4.0)
    test = student("Test", 18, 0)
    test.updateGPA(3.26)
main()
```

Inheritance

Inheritance

- Inheritance is when one class (the "child" class) is based upon another class (the "parent" class)
- The child class inherits most or all of its features from the parent class it is based on
- It is a very powerful tool available to you with Object-Oriented Programming

Inheritance Example

- For example: computer science students are a specific type of student
- They share attributes with every other student
- We can use inheritance to use those already defined attributes and methods of students for our computer science students

Inheritance Vocabulary

- The class that is inherited from is called the
 - Parent class
 - Ancestor
 - Superclass
- The class that does the inheriting is called a
 - Child class
 - Descendant
 - -Subclass

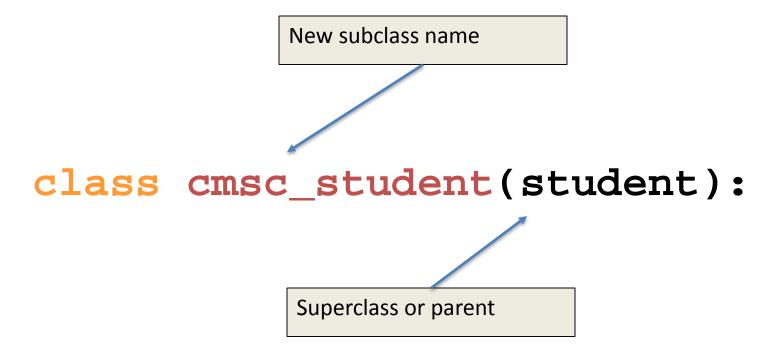
Inheritance Code

 To create a child class, put the name of the parent class in parentheses when you initially define the class

class cmscStudent(student):

 Now the child class cmscStudent has the properties and functions available to the parent class student

Subclass Example



Extending a Class

- We may also say that the child class is
 extending the functionality of the parent class
- Child class inherits all of the methods and data attributes of the parent class
 - Also has its own methods and data attributes
 - We can even redefine parent methods!



Inheritance Example

```
class student:
  """A class representing a student."""
  def __init__(self,n,a):
      self.full name = n
      self.age = a
  def get_age(self):
      return self.age
class cs student (student):
  """A class extending student."""
  def __init__(self,n,a,s):
      student. init (self,n,a) #Call init for student
      self.section num = s
  def get age(self): #Redefines get age method entirely
      print ("Age: " + str(self.age)
```

Redefining Methods

Redefining Methods

- Redefining a method is when a child class implements its own version of that method
- To redefine a method, include a new method definition – with the same name as the parent class's method – in the child class
 - Now child objects will use the new method



Redefining Example

 Here, we have an animal class as the parent and a dog class as the child

```
class animal:
    # rest of class definition
    def speak(self):
        print("\"" + self.species + " noise\"")

class dog(animal):
    def speak(self):
        print("Woof woof bark!")
```

Extending Methods

- Instead of completely overwriting a method,
 we can instead extend it for the child class
- When might we want to do this?
 - Constructor (___init___)
 - Print function (___repr___)
 - -When else?

Extending a Method

- Want to execute both the <u>original method</u> in the parent class and some <u>new code</u> in the child class
 - To do this, explicitly call the parent's version
- One major thing: you must pass in the self variable when you call a parent method
 - —This is the only time you should do this!



Extending Example

Now we have a cat class as the child, with an additional data attribute sleepsAllDay

```
class animal:
    def __init__(self, name, species):
        self.name = name
        self.species = species

class cat(animal):
    def __init__(self, name, sleepsAllDay):
        animal.__init__(self, name, "cat")
        self.sleepsAllDay = sleepsAllDay
```



Student Inheritance Example

```
class student:
  """A class representing a student."""
  def init (self, name, age):
      self.full name = name
      self.age = age
  def getAge(self):
      return self.age
class cmscStudent (student):
  """A class extending student class to CMSC students."""
  def __init__(self, name, age, section):
      # call __init__ for student
      student.__init__(self, name, age)
      self.section num = section
  def getAge(self): # redefines getAge method entirely
      print ("Age: " + str(self.age)
```

LIVECODING!!!

Any Other Questions?

Announcements

- Lab has been cancelled this week!
 - Work on your project instead

- Project 1 is out
 - Due by Tuesday, November 17th at 8:59:59 PM
 - Do NOT procrastinate!
- Next Class: Recursion