

CMSC201

Computer Science I for Majors

Lecture 22 – Classes and Modules (Final Continuation)

Prof. Katherine Gibson

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

- 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
    age = a
    gpa = g
  def updateGPA(newGPA) :
    gpa = newGPA

def main():
  val = new student("Alex", 21, 4.0)
  test = new student("Test", 18, 0)
  updateGPA(test, 3.26)

main()
```

There are at
least seven
unique errors

Find the Errors in the Code Below

```
def student:  
    def init(self, n, a, g):  
        name = n  
        age = a  
        gpa = g  
    def updateGPA(newGPA):  
        gpa = 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

```
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**

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!

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("\n" + self.species + " noise\n")

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__`)
 - Function used when printing (`__str__`)
 - When else?

Extending a Method

- Want to execute both the original method in the parent class and some new code 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

- Homework 8 is out
 - Due by Monday, April 25th at 8:59:59 PM
- Project 2 will be out soon
- Final exam: Common Final
 - Friday, May 13th from 6 to 8 PM
 - If you have religious/sports exemptions that prevent you from taking the exam then, let us know ASAP
 - (List when your other exams are being held)

Practice Problems

- Create a **food** class
 - Attributes: **foodName** and **calories**
 - Methods: **__init__()**, **__str__()**, and **cook()**
 - **cook()** should set **calories** to 0 (food burnt)
- Create a **fruit** class (inherits from **food**)
 - Methods: extend **__init__()** and **__str__()** and override **cook()** to drop **calories** by half
 - Additional attributes: **ripeness** (goes from 0 to 10)

Practice Problems: Sample `main ()`

- This `main ()` should work with your classes

```
def main():
```

```
    myFood = food("hot dog", 80)
```

```
    print(myFood)    # should print 80 calories
```

```
    myFood.cook()
```

```
    print(myFood)    # should print 0 calories
```

```
    myFruit = fruit("pear", 60, 8)
```

```
    print(myFruit)   # print 60 calories and ripeness level
```

```
    myFruit.cook()
```

```
    print(myFruit)   # print 30 calories and ripeness level
```