CS 150: Inheritance

Cynthia Taylor
Oberlin College
March 21st 2014

Inheritance

- Many times, a new class will have much in common with an existing class
- Inheritance allows a new class to specialize an existing class by specifying only what is different between them
- The class that inherits is called a *subclass*, and the class that is inherited from is its *superclass*

is-a

- inheritance models is-a relationships
- The subclass “is-a” subset of the superclass
  - The superclass could be `Ship`, representing all ships
  - The subclass could be `CloakedShip` that “is-a” Ship but cannot be detected with sensors

has-a

- Be careful not to use inheritance to model has-a relationships
- For example, consider a `Ship` and a `PhotonTorpedo`
  - A PhotonTorpedo is not a Ship
  - A Ship is not a PhotonTorpedo
  - Inheritance cannot be used
  - A Ship has a PhotonTorpedo, not is a PhotonTorpedo
- Has-a relationships are modeled with instance variables, not inheritance

In the following pairs of words, the first is the *subclass* and the second is the *superclass*. Which of them is a correct example of inheritance?

A. dog, cat
B. dog, animal
C. animal, dog
D. dog, tail
E. None of the above

is-a

has-a

Inheriting from a Class

- To inherit from another class, use that class' name in parentheses in the class declaration

```
class CloakedShip(Ship):
    ...
```

- When calling a method on a CloakedShip:
  - If the method exists in CloakedShip, it is called
  - Otherwise, the one in Ship is called

```
class A:
    def __init__(self, x):
        self.x = x
    def __str__(self):
        return str(self.x)

class B(A):
    def __init__(self, x):
        self.x = x * 2

b = B(5)
print(b)
```

What is the output of this code?

A. 5

B. 10

C. 510

D. This will cause an error

E. I don't know

Person Class

```
p = Person("George", "Williker")
s = Student("Buddy","Bob")
s.study()
p.study()
```

What is the output of this code?

A. "Need to study...
but...Internet!" will print twice
B. "Need to study... but...Internet!" will print once
C. Nothing will print
D. This will cause an error
E. I don't know

Type Checking

- Sometimes we need to check the *type* of an object before calling a method

```
if type(p) == Student:
  p.study()
else:
  p.work()
```

What if we want a variable that is shared by all members of a class?

- Say we want to count the total number of students

- `self.count` will be unique for each object, not shared by the whole class

- We need a ~~*global*~~ class variable

```
Class Student(Person):
  count = 0
def __init__(self, first, last):
  ....
  Student.count = Student.count + 1
```

- `Student.count` will be shared by all objects of type student

s = Student()
s.count +

- Can access either by using classname.variable name, or objectname.variablename

```
class A:
  count = 0
  def __init__(self, x):
      self.x = x+1
      A.count= A.count+1

A1 = A(2)
A2 = A(4)
A3 = A(5)

print(A1.x, A1.count,
  A.count)
```

What is the output of this code?

A. 3, 1, 1

B. 3, 1, 3

C. 3, 3, 3

D. 2, 1, 1

E. I don't know

Next Class

- Binary
- Lab 7 – Due Tuesday AFTER BREAK at 10 pm
- Prelab 8 – Due in class on Wednesday