Inheritance

CSCI120 Intro to Computing

Hierarchies

Hierarchies

class Animal
- self.age
- self.name

- def get_age(self)
- def get_name(self)
- def set_age(self, ag)
- def set_name(self, nm)

class People
- self. age
- self.name
- **self.height**

- def get_age(self)
- def get_name(self)
- def set_age(self, ag)
- def set_name(self, nm)
- **def speak(self)**

class Dog
- self. age
- self.name

- def get_age(self)
- def get_name(self)
- def set_age(self, ag)
- def set_name(self, nm)
- **def move(self)**
- **def woof(self)**

class Cat
- self. age
- self.name

- def get_age(self)
- def get_name(self)
- def set_age(self, ag)
- def set_name(self, nm)
- **def meow(self)**

class Student
- self. age
- self.name
- **self.height**
- **self.studentId**

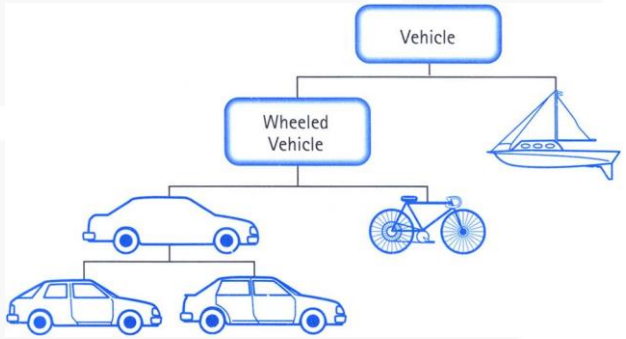
- def get_age(self)
- def get_name(self)
- def set_age(self, ag)
- def set_name(self, nm)
- **def speak(self)**
- **def do_homework(self)**

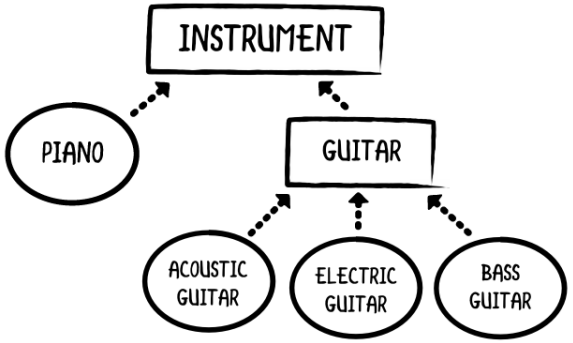
class Professor
- self. age
- self.name
- **self.height**
- **self.employeeId**

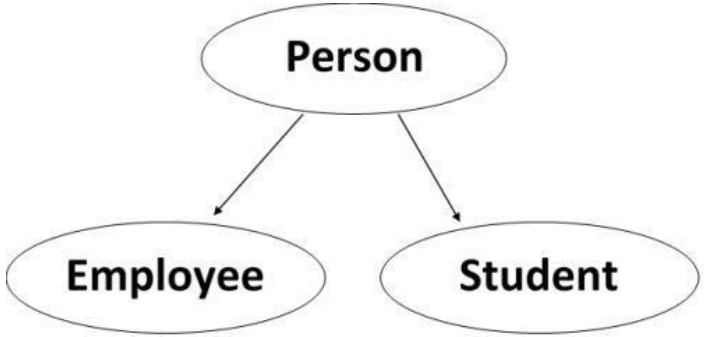
- def get_age(self)
- def get_name(self)
- def set_age(self, ag)
- def set_name(self, nm)
- **def speak(self)**
- **def teach(self)**

Inheritance

- Inheritance allows us to define a class that inherits all the methods and attributes from another class.
- Why we need inheritance:
  - Represent ancestral relationships







Class Implementation

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(object):
    def __init__(self, age, name, height):
        self.age = age
        self.name = name
        self.height = height

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag

    def speak(self):
        print("Hello")
```

```
class Student(object):
    def __init__(self, age, name, height, studentId):
        self.age = age
        self.name = name
        self.height = height
        self.studentId = studentId

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag

    def speak(self):
        print("Hello")

    def do_homework(self):
        print(f"student {self.name} is doing homework")
```

Duplicated and Growing Codes

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(object):
    def __init__(self, age, name, height):
        self.age = age
        self.name = name
        self.height = height

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag

    def speak(self):
        print("Hello")
```

```
class Student(object):
    def __init__(self, age, name, height, studentId):
        self.age = age
        self.name = name
        self.height = height
        self.studentId = studentId

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag

    def speak(self):
        print("Hello")

    def do_homework(self):
        print(f"student {self.name} is doing homework")
```

Duplicated and Growing Codes

```python
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```python
class People(object):
    def __init__(self, age, name, height):
        self.age = age
        self.name = name
        self.height = height

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag

    def speak(self):
        print("Hello")
```

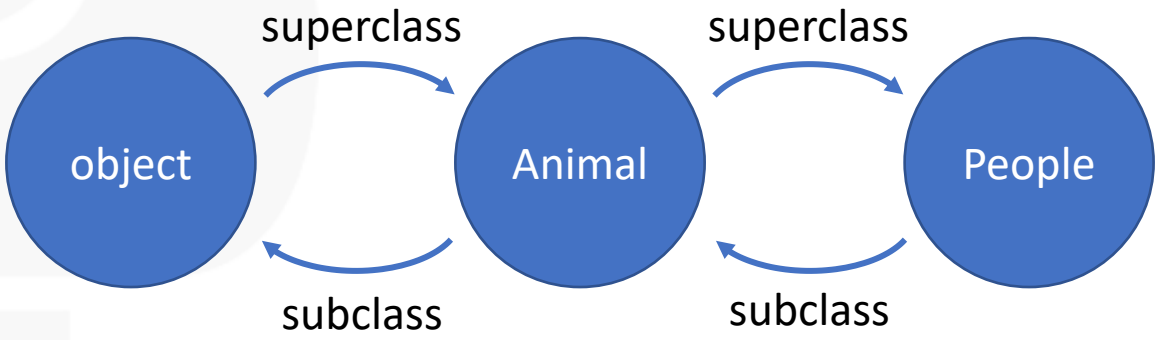
```python
class Student(object):
    def __init__(self, age, name, height, studentId):
        self.age = age
        self.name = name
        self.height = height
        self.studentId = studentId

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag

    def speak(self):
        print("Hello")

    def do_homework(self):
        print(f"student {self.name} is doing homework")
```

Inheritance

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

everything is an object
Implements basic operations in Python, like binding variables, etc.



- Parent class (superclass)
  - Parent class is the class being inherited from, also called base class
- Child class (subclass)
  - inherits all data attributes and behaviors of **parent** class
  - add more info
  - add more behavior
  - override behavior

Inheritance

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

a new attribute

a new method

super() – refers to the parent method

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

Child class can use parents' attributes and methods by **super()**

Inheritance

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
p = People(20, "Alex", 60)
print("name", p.name)
print("age", p.get_age())
```

- subclass can have attributes with same name as superclass
- subclass can have methods with same name as superclass

Student Classs

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")
```

Which line is wrong?

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")
```

```
s = Student(20, "Alex", 60, 12345)
print(s.age, s.name)  # 1
s.get_age()  # 2
s.speak()  # 3
s.do_homework()  # 4
```

- A. 1
- B. 2
- C. 1 and 2
- D. 3
- E. All correct

Which line is wrong?

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")
```

```
s = Student(20, "Alex", 60, 12345)
print(s.age, s.name)  # 1
s.get_age()  # 2
s.speak()  # 3
s.do_homework()  # 4
```

- A. 1
- B. 2
- C. 1 and 2
- D. 3
- **E. All correct**

Which line is wrong?

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")
```

```
s = Student(20, "Alex", 60, 12345)
print(s.age, s.name)  # 1
s.get_age()  # 2
s.speak()  # 3
s.do_homework()  # 4
```

- Inherited from Animal

Which line is wrong?

```python
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```python
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```python
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")
```

```python
s = Student(20, "Alex", 60, 12345)
print(s.age, s.name)  # 1
s.get_age()  # 2
s.speak()  # 3
s.do_homework()  # 4
```

- Inherited from Animal
- Inherited from People

Overriding

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

overrides

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")

    def speak(self):
        print(f"My name is {self.name}")
```

- Define the same method of parent class will override parent's method
- For an instance of a class, look for a method name in **current class definition**
  - student.speak()
- If not found, look for method name **up the hierarchy** (in parent, then grandparent, and so on)
  - Use first method up the hierarchy that you found with that method name
  - student.get_age(), student.set_age()

Overriding

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")

    def speak(self):
        super().speak()
        print(f"My name is {self.name}")
```

```
p = People(22, "Ben", 62)
p.speak()  # Hello
s = Student(20, "Alex", 60, 12345)
s.speak()
```

Overriding

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")

    def speak(self):
        super().speak()
        print(f"My name is {self.name}")
```

```
p = People(22, "Ben", 62)
p.speak()  # Hello
s = Student(20, "Alex", 60, 12345)
s.speak()
```

Overriding

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")

    def speak(self):
        super().speak()
        print(f"My name is {self.name}")
```

```
p = People(22, "Ben", 62)
p.speak()  # Hello
s = Student(20, "Alex", 60, 12345)
s.speak()
```

Overriding

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")

    def speak(self):
        super().speak()
        print(f"My name is {self.name}")
```

```
p = People(22, "Ben", 62)
p.speak()  # Hello
s = Student(20, "Alex", 60, 12345)
s.speak()  # Hello
```

Overriding

```python
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```python
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```python
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")

    def speak(self):
        super().speak()
        print(f"My name is {self.name}")
```

```python
p = People(22, "Ben", 62)
p.speak()  # Hello
s = Student(20, "Alex", 60, 12345)
s.speak()  # Hello
           # My name is Alex
```

Overriding __init__()

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")

    def speak(self):
        super().speak()
        print(f"My name is {self.name}")
```

```
p = People(22, "Ben", 62)
p.speak()  # Hello
s = Student(20, "Alex", 60, 12345)
s.speak()  # Hello
           # My name is Alex
```

Overriding __init__()

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")

    def speak(self):
        super().speak()
        print(f"My name is {self.name}")
```

```
p = People(22, "Ben", 62)
p.speak()  # Hello
s = Student(20, "Alex", 60, 12345)
s.speak()  # Hello
           # My name is Alex
```

Overriding __init__()

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")

    def speak(self):
        super().speak()
        print(f"My name is {self.name}")
```

```
p = People(22, "Ben", 62)
p.speak()  # Hello
s = Student(20, "Alex", 60, 12345)
s.speak()  # Hello
           # My name is Alex
```

Overriding __init__()

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")

    def speak(self):
        super().speak()
        print(f"My name is {self.name}")
```

```
p = People(22, "Ben", 62)
p.speak()  # Hello
s = Student(20, "Alex", 60, 12345)
s.speak()  # Hello
           # My name is Alex
```

Overriding __init__()

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")

    def speak(self):
        super().speak()
        print(f"My name is {self.name}")
```

```
p = People(22, "Ben", 62)
p.speak()  # Hello
s = Student(20, "Alex", 60, 12345)
s.speak()  # Hello
           # My name is Alex
```

Overriding __init__()

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")

    def speak(self):
        super().speak()
        print(f"My name is {self.name}")
```

```
p = People(22, "Ben", 62)
p.speak()  # Hello
s = Student(20, "Alex", 60, 12345)
s.speak()  # Hello
           # My name is Alex
```

Overriding __init__()

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")

    def speak(self):
        super().speak()
        print(f"My name is {self.name}")
```

```
p = People(22, "Ben", 62)
p.speak()  # Hello
s = Student(20, "Alex", 60, 12345)
s.speak()  # Hello
           # My name is Alex
```

Overriding __init__()

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")

    def speak(self):
        super().speak()
        print(f"My name is {self.name}")
```

```
p = People(22, "Ben", 62)
p.speak()  # Hello
s = Student(20, "Alex", 60, 12345)
s.speak()  # Hello
           # My name is Alex
```

Overriding __init__()

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")

    def speak(self):
        super().speak()
        print(f"My name is {self.name}")
```

```
p = People(22, "Ben", 62)
p.speak()  # Hello
s = Student(20, "Alex", 60, 12345)
s.speak()  # Hello
           # My name is Alex
```

Overriding __init__()

```
class Animal(object):
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, ag):
        self.age = ag
```

```
class People(Animal):
    def __init__(self, age, name, height):
        super().__init__(age, name)
        self.height = height

    def speak(self):
        print("Hello")
```

```
class Student(People):
    def __init__(self, age, name, height, studentId):
        super().__init__(age, name, height)
        self.studentId = studentId

    def do_homework(self):
        print(f"student {self.name} is doing homework")

    def speak(self):
        super().speak()
        print(f"My name is {self.name}")
```

```
p = People(22, "Ben", 62)
p.speak()  # Hello
s = Student(20, "Alex", 60, 12345)
s.speak()  # Hello
           # My name is Alex
```

```
class Employee(object):
    def __init__(self):
        self.salary = 0

    def set_salary(self, sa):
        self.salary = sa

    def get_salary(self):
        return self.salary

    def raise_salary(self, rate):
        self.salary = self.salary * rate

class Manager(Employee):
    def raise_salary(self, rate):
        self.salary = self.salary * (rate + 0.1)

m = Manager()
m.set_salary(1000)
m.raise_salary(1.2)
print(m.get_salary())
```

What will be printed?

- A. 0
- B. 1000
- C. 1200
- D. 1300
- E. Error

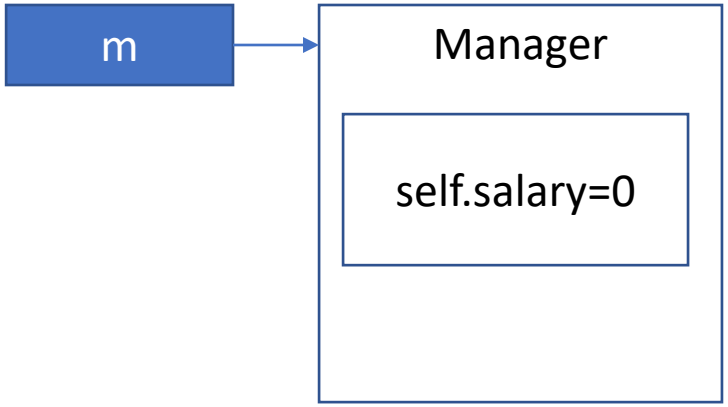
```
class Employee(object):
    def __init__(self):
        self.salary = 0

    def set_salary(self, sa):
        self.salary = sa

    def get_salary(self):
        return self.salary

    def raise_salary(self, rate):
        self.salary = self.salary * rate

class Manager(Employee):
    def raise_salary(self, rate):
        self.salary = self.salary * (rate + 0.1)

m = Manager()
m.set_salary(1000)
m.raise_salary(1.2)
print(m.get_salary())
```

What will be printed?

- A. 0
- B. 1000
- C. 1200
- D. 1300
- E. Error

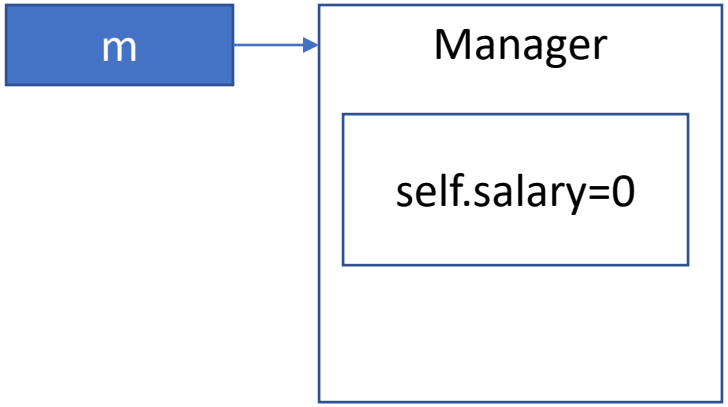
What will be printed?

```
class Employee(object):
    def __init__(self):
        self.salary = 0

    def set_salary(self, sa):
        self.salary = sa

    def get_salary(self):
        return self.salary

    def raise_salary(self, rate):
        self.salary = self.salary * rate

class Manager(Employee):
    def raise_salary(self, rate):
        self.salary = self.salary * (rate + 0.1)

m = Manager()
m.set_salary(1000)
m.raise_salary(1.2)
print(m.get_salary())
```



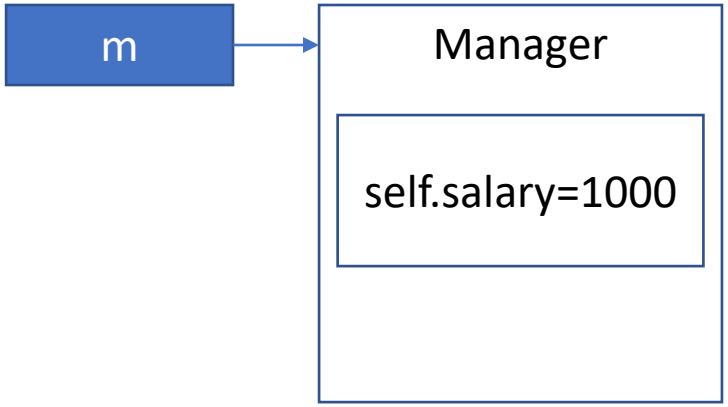
What will be printed?

```
class Employee(object):
    def __init__(self):
        self.salary = 0

    def set_salary(self, sa):
        self.salary = sa

    def get_salary(self):
        return self.salary

    def raise_salary(self, rate):
        self.salary = self.salary * rate

class Manager(Employee):
    def raise_salary(self, rate):
        self.salary = self.salary * (rate + 0.1)

m = Manager()
m.set_salary(1000)
m.raise_salary(1.2)
print(m.get_salary())
```



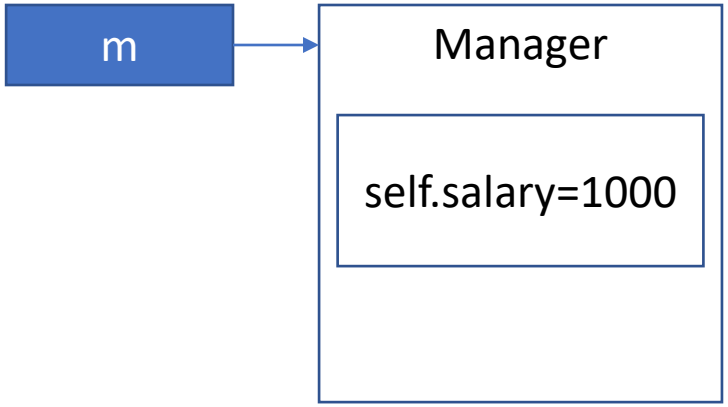
What will be printed?

```
class Employee(object):
    def __init__(self):
        self.salary = 0

    def set_salary(self, sa):
        self.salary = sa

    def get_salary(self):
        return self.salary

    def raise_salary(self, rate):
        self.salary = self.salary * rate

class Manager(Employee):
    def raise_salary(self, rate):
        self.salary = self.salary * (rate + 0.1)

m = Manager()
m.set_salary(1000)
m.raise_salary(1.2)
print(m.get_salary())
```



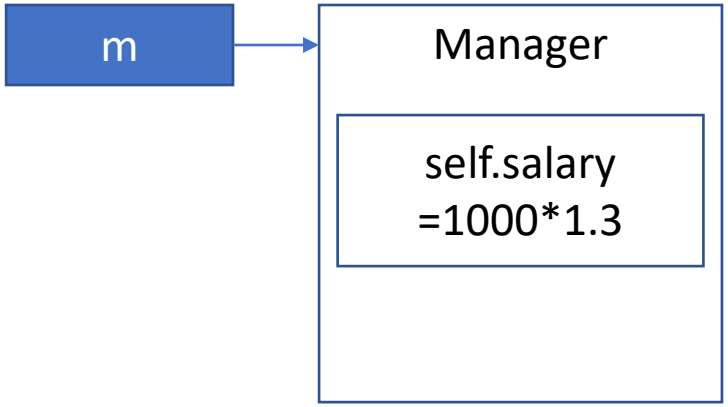
What will be printed?

```
class Employee(object):
    def __init__(self):
        self.salary = 0

    def set_salary(self, sa):
        self.salary = sa

    def get_salary(self):
        return self.salary

    def raise_salary(self, rate):
        self.salary = self.salary * rate

class Manager(Employee):
    def raise_salary(self, rate):
        self.salary = self.salary * (rate + 0.1)

m = Manager()
m.set_salary(1000)
m.raise_salary(1.2)
print(m.get_salary())
```



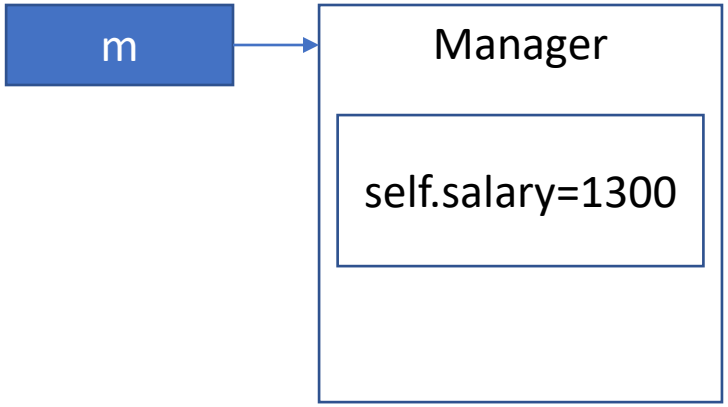
What will be printed?

```
class Employee(object):
    def __init__(self):
        self.salary = 0

    def set_salary(self, sa):
        self.salary = sa

    def get_salary(self):
        return self.salary

    def raise_salary(self, rate):
        self.salary = self.salary * rate

class Manager(Employee):
    def raise_salary(self, rate):
        self.salary = self.salary * (rate + 0.1)

m = Manager()
m.set_salary(1000)
m.raise_salary(1.2)
print(m.get_salary())
```



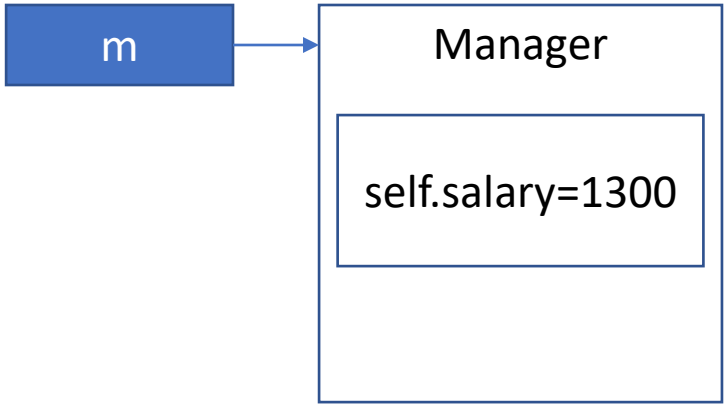
What will be printed?

```
class Employee(object):
    def __init__(self):
        self.salary = 0

    def set_salary(self, sa):
        self.salary = sa

    def get_salary(self):
        return self.salary

    def raise_salary(self, rate):
        self.salary = self.salary * rate

class Manager(Employee):
    def raise_salary(self, rate):
        self.salary = self.salary * (rate + 0.1)

m = Manager()
m.set_salary(1000)
m.raise_salary(1.2)
print(m.get_salary())
```



What will be printed?

```
class Employee(object):
    def __init__(self):
        self.salary = 0

    def set_salary(self, sa):
        self.salary = sa

    def get_salary(self):
        return self.salary

    def raise_salary(self, rate):
        self.salary = self.salary * rate

class Manager(Employee):
    def raise_salary(self, rate):
        self.salary = self.salary * (rate + 0.1)

m = Manager()
m.set_salary(1000)
m.raise_salary(1.2)
print(m.get_salary())
```



```python
class Employee(object):
    def __init__(self):
        self.salary = 0

    def set_salary(self, sa):
        self.salary = sa

    def get_salary(self):
        return self.salary

    def raise_salary(self, rate):
        self.salary = self.salary * rate

class Manager(Employee):
    def raise_salary(self, rate):
        super().raise_salary(rate)
        self.salary += 100

m = Manager()
m.set_salary(1000)
m.raise_salary(1.2)
print(m.get_salary())
```

What will be printed?

- A. 0
- B. 1000
- C. 1200
- D. 1300
- E. Error

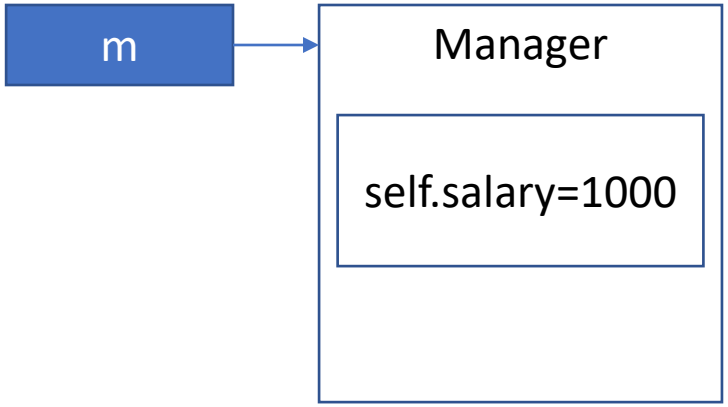
```python
class Employee(object):
    def __init__(self):
        self.salary = 0

    def set_salary(self, sa):
        self.salary = sa

    def get_salary(self):
        return self.salary

    def raise_salary(self, rate):
        self.salary = self.salary * rate

class Manager(Employee):
    def raise_salary(self, rate):
        super().raise_salary(rate)
        self.salary += 100

m = Manager()
m.set_salary(1000)
m.raise_salary(1.2)
print(m.get_salary())
```

What will be printed?

- A. 0
- B. 1000
- C. 1200
- D. 1300
- E. Error

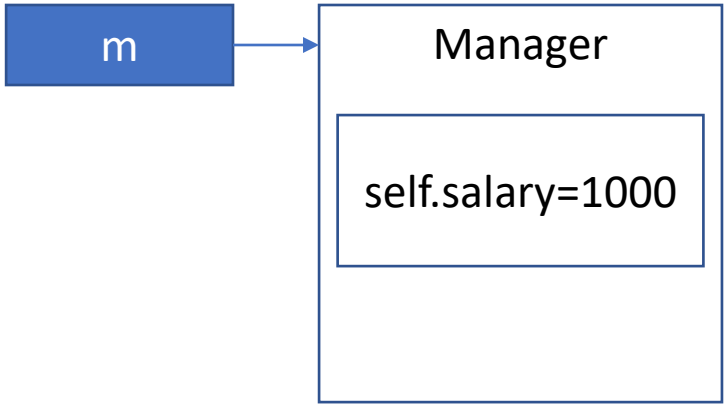
```
class Employee(object):
    def __init__(self):
        self.salary = 0

    def set_salary(self, sa):
        self.salary = sa

    def get_salary(self):
        return self.salary

    def raise_salary(self, rate):
        self.salary = self.salary * rate

class Manager(Employee):
    def raise_salary(self, rate):
        super().raise_salary(rate)
        self.salary += 100

m = Manager()
m.set_salary(1000)
m.raise_salary(1.2)
print(m.get_salary())
```

What will be printed?



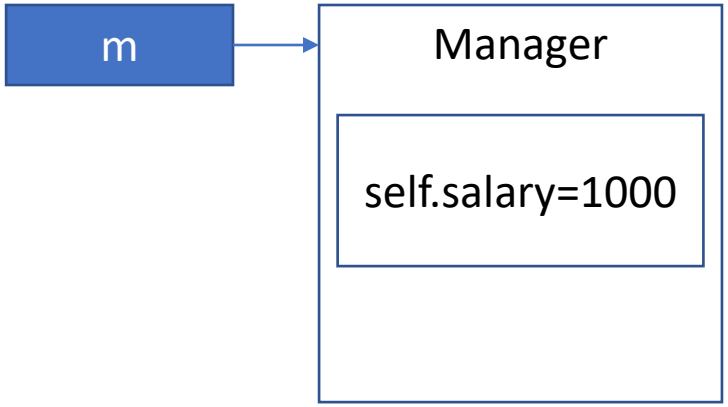
What will be printed?

```
class Employee(object):
    def __init__(self):
        self.salary = 0

    def set_salary(self, sa):
        self.salary = sa

    def get_salary(self):
        return self.salary

    def raise_salary(self, rate):
        self.salary = self.salary * rate

class Manager(Employee):
    def raise_salary(self, rate):
        super().raise_salary(rate)
        self.salary += 100

m = Manager()
m.set_salary(1000)
m.raise_salary(1.2)
print(m.get_salary())
```



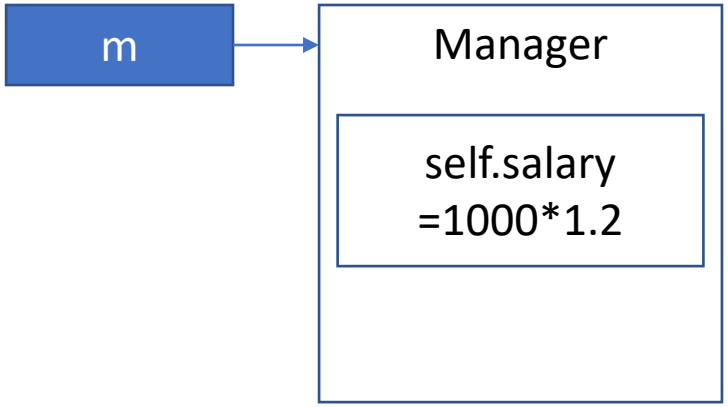
```
class Employee(object):
    def __init__(self):
        self.salary = 0

    def set_salary(self, sa):
        self.salary = sa

    def get_salary(self):
        return self.salary

    def raise_salary(self, rate):
        self.salary = self.salary * rate

class Manager(Employee):
    def raise_salary(self, rate):
        super().raise_salary(rate)
        self.salary += 100

m = Manager()
m.set_salary(1000)
m.raise_salary(1.2)
print(m.get_salary())
```

What will be printed?



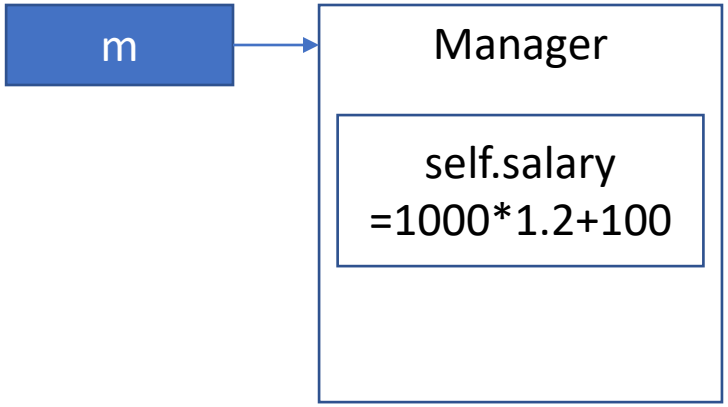
What will be printed?

```
class Employee(object):
    def __init__(self):
        self.salary = 0

    def set_salary(self, sa):
        self.salary = sa

    def get_salary(self):
        return self.salary

    def raise_salary(self, rate):
        self.salary = self.salary * rate

class Manager(Employee):
    def raise_salary(self, rate):
        super().raise_salary(rate)
        self.salary += 100

m = Manager()
m.set_salary(1000)
m.raise_salary(1.2)
print(m.get_salary())
```



```
class Employee(object):
    def __init__(self):
        self.salary = 0

    def set_salary(self, sa):
        self.salary = sa

    def get_salary(self):
        return self.salary

    def raise_salary(self, rate):
        self.salary = self.salary * rate

class Manager(Employee):
    def raise_salary(self, rate):
        super().raise_salary(rate)
        self.salary += 100

m = Manager()
m.set_salary(1000)
m.raise_salary(1.2)
print(m.get_salary())
```

What will be printed?



```
class Employee(object):
    def __init__(self):
        self.salary = 0

    def set_salary(self, sa):
        self.salary = sa

    def get_salary(self):
        return self.salary

    def raise_salary(self, rate):
        self.salary = self.salary * rate

class Manager(Employee):
    def raise_salary(self, rate):
        super().raise_salary(rate)
        self.salary += 100

m = Manager()
m.set_salary(1000)
m.raise_salary(1.2)
print(m.get_salary())
```

What will be printed?