Inheritance

Inheritance

- Inheritance is one of the important concepts in Object-oriented programming.

- Inheritance allows one class to *inherit all attributes and methods* of an another class and reuse existing codes.
  - Student class can inherit attributes and methods from Person class (e.g., name, age, incrementAge(), getInfo(), etc.)
  - ElectricCar class can inherit attributes and methods from Car class (e.g., model, year, mileage, weight, incrementMileage, start(), stop(), etc.)

Inheritance

- The original class is called the parent class (superclass/base class), and the new class is the child class (subclass/derived class).
- The child class inherits all attributes and methods from its parent class.
- The syntax to inherit another class

class ChildClassName(ParentClassName):

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def printInfo(self):
        print("Name: "+self.name+ " Age: "+str(self.age))


class Student(Person):
    pass

st = Student("Adam",123123)
st.printInfo()
```

Defining Child Class Attributes and Methods

- The child class can also define new attributes and methods of its own.

- When you use inheritance, you can make your child classes retain what you need and override anything you don't need from the parent class.

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def printInfo(self):
        print("Name: "+self.name+ " Age: "+str(self.age))

class Student(Person):
    def __init__(self, sid, name, age):
        self.studentID = sid
        super().__init__(name, age)

st = Student(10,"Adam",123123)
st.printInfo()
```

Defining Child Class Attributes and Methods

- The child class can also define new attributes and methods of its own.
- When you use inheritance, you can make your child classes retain what you need and override anything you don't need from the parent class.

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def printInfo(self):
        print("Name: "+self.name+ " Age: "+str(self.age))

class Student(Person):
    def __init__(self, sid, name, age):
        self.studID = sid
        super().__init__(name, age)
    def printInfo(self):
        print("ID: "+str(self.studID)+"Name: "+self.name+ " Age: "+str(self.age))

st = Student(10,"Adam",123123)
st.printInfo()
```

Instances as Attributes

- Our program may get large as we add more details to our class.
- When it gets large, we can break our large class into smaller classes that work together

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def printInfo(self):
        print("Name: "+self.name+ " Age: "+str(self.age))

class Student(Person):
    def __init__(self, sid, year, name, age):
        self.studID = sid
        self.idYear = year
        super().__init__(name, age)
    def printInfo(self):
        print("ID: "+str(self.studID)+"Name: "+self.name+ " Age: "+str(self.age))
    def getIDYear(self):
        return self.idYear

st = Student(10,"Adam",123123)
st.printInfo()
```

Instances as Attributes

```
class StudentID:
    def __init__(self, id, year):
        self.id = id
        self.year = year
     def getIDYear(self):
         return self.year
```

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def printInfo(self):
        print("Name: "+self.name+ " Age: "+str(self.age))

class Student(Person):
    def __init__(self, sid, year, name, age):
        self.studid = StudentID(sid, year)
        super().__init__(name, age)
    def printInfo(self):
        print("ID: "+str(self.studID)+"Name: "+self.name+ " Age: "+str(self.age))
    #def getIDYear(self):
    #   return self.idYear

st = Student(10,"Adam",123123)
st.printInfo()
print(st.studid.getIDYear())
```

Importing Classes

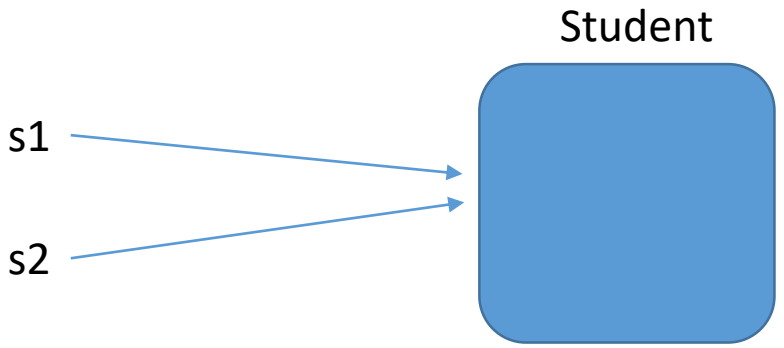
- We can write as many classes as we want in a single file. But it makes your program difficult to read.
- We can store our classes in modules and import them to our main program.
- We can store several classes in one module

```
from moduleName import ClassName
from moduleName import ClassName1, ClassName2
import moduleName
```

Copying Objects

- When we assign one object to another (aliasing), then both will point to the same object

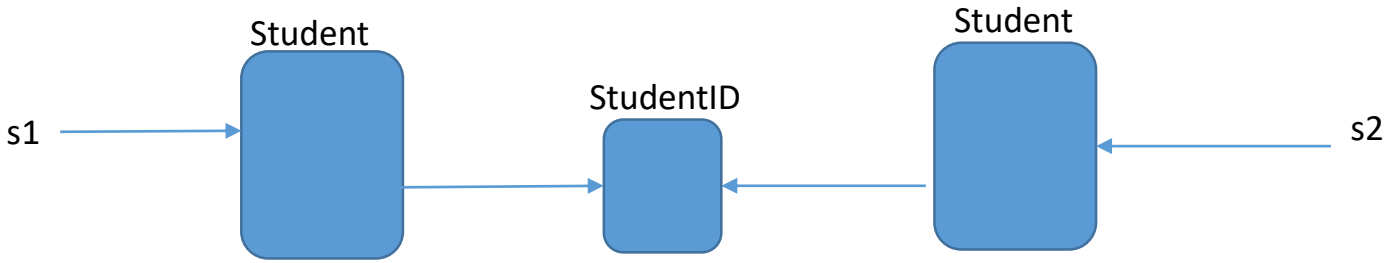
```
s1 = Student()
s2 = s1
print(s1 == s2) #True
```



- We can use the copy module to copy objects.

Copying Objects

```
import copy
s1 = Student()
s2 = copy.copy(s1)    # shallow copy – does not copy embedded objects; copies references
```




```
import copy
s1 = Student()
s2 = copy.deepcopy(s1)    # deep copy – copies embedded objects
```