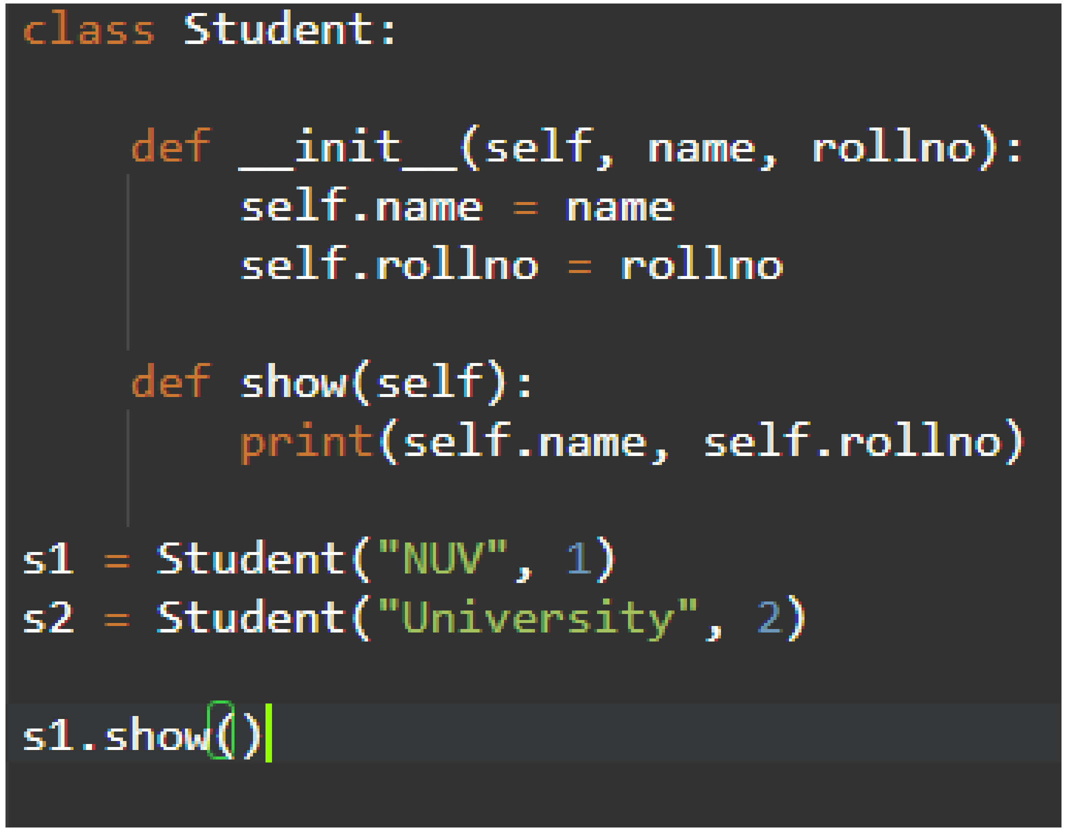
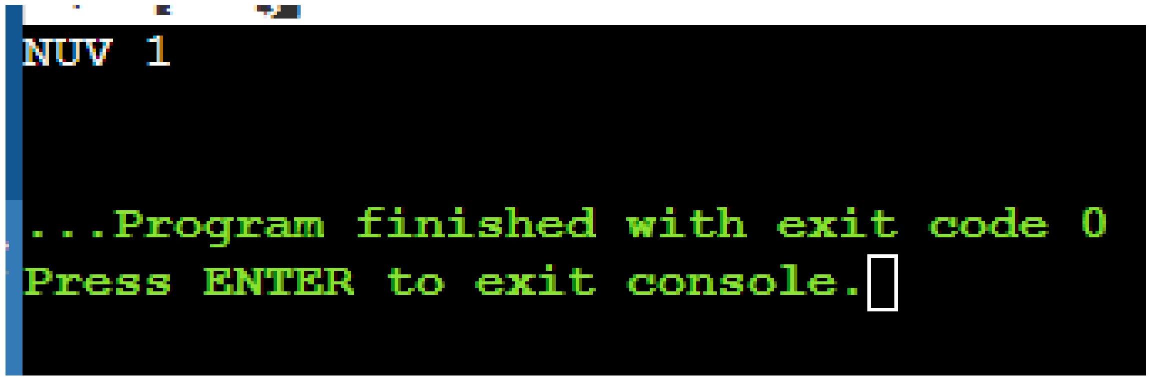
Class & Object – II

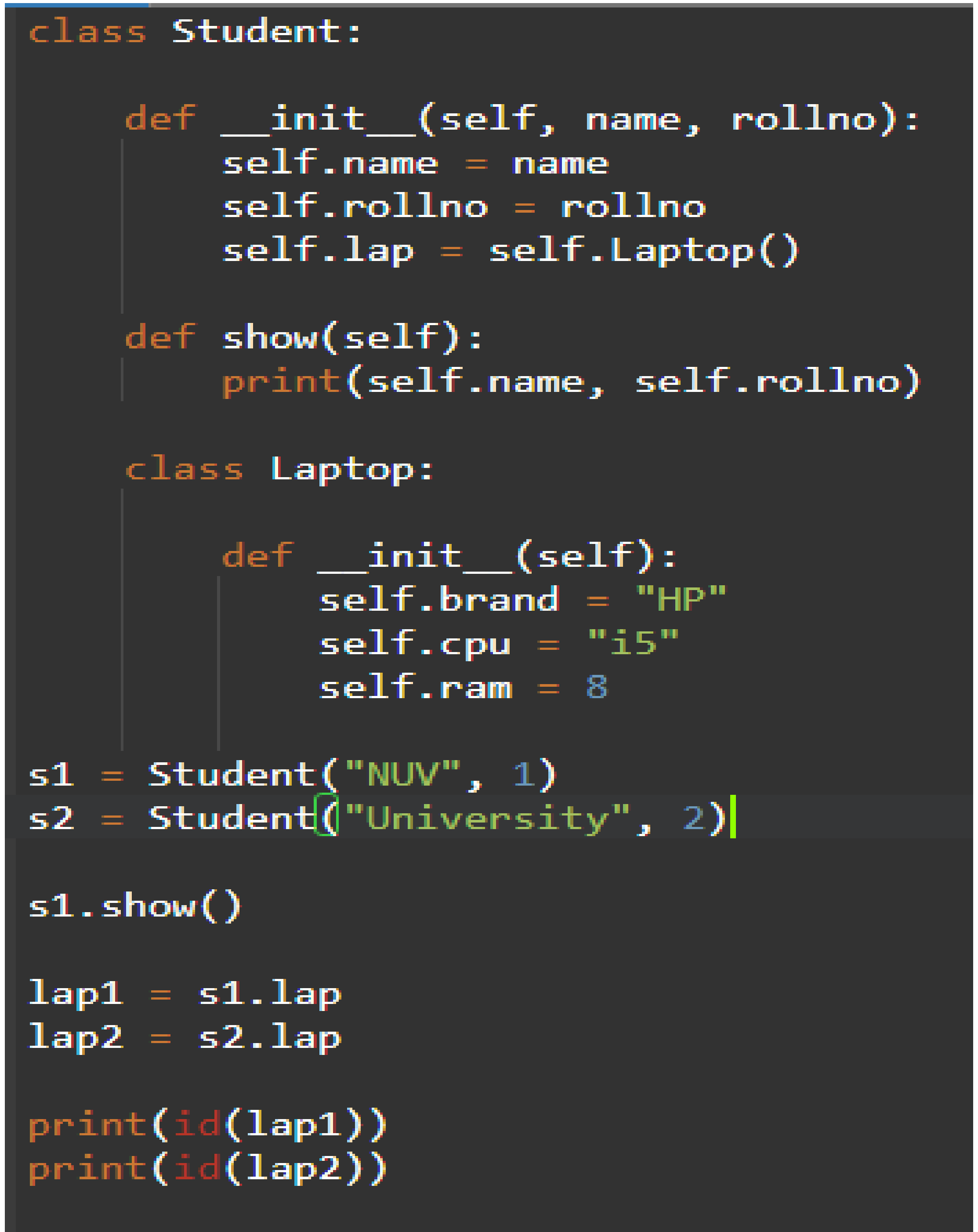
Inner Class



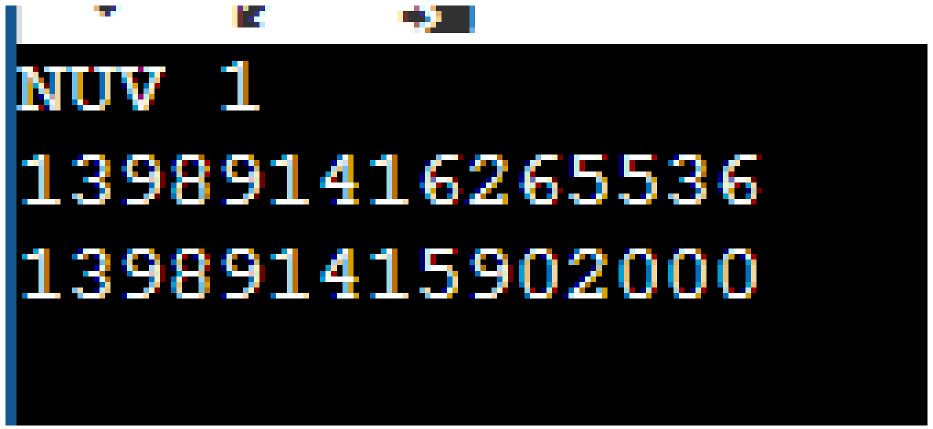
```python
class Student:

    def __init__(self, name, rollno):
        self.name = name
        self.rollno = rollno

    def show(self):
        print(self.name, self.rollno)

s1 = Student("NUV", 1)
s2 = Student("University", 2)

s1.show()
```



```
NUV 1


...Program finished with exit code 0
Press ENTER to exit console.
```

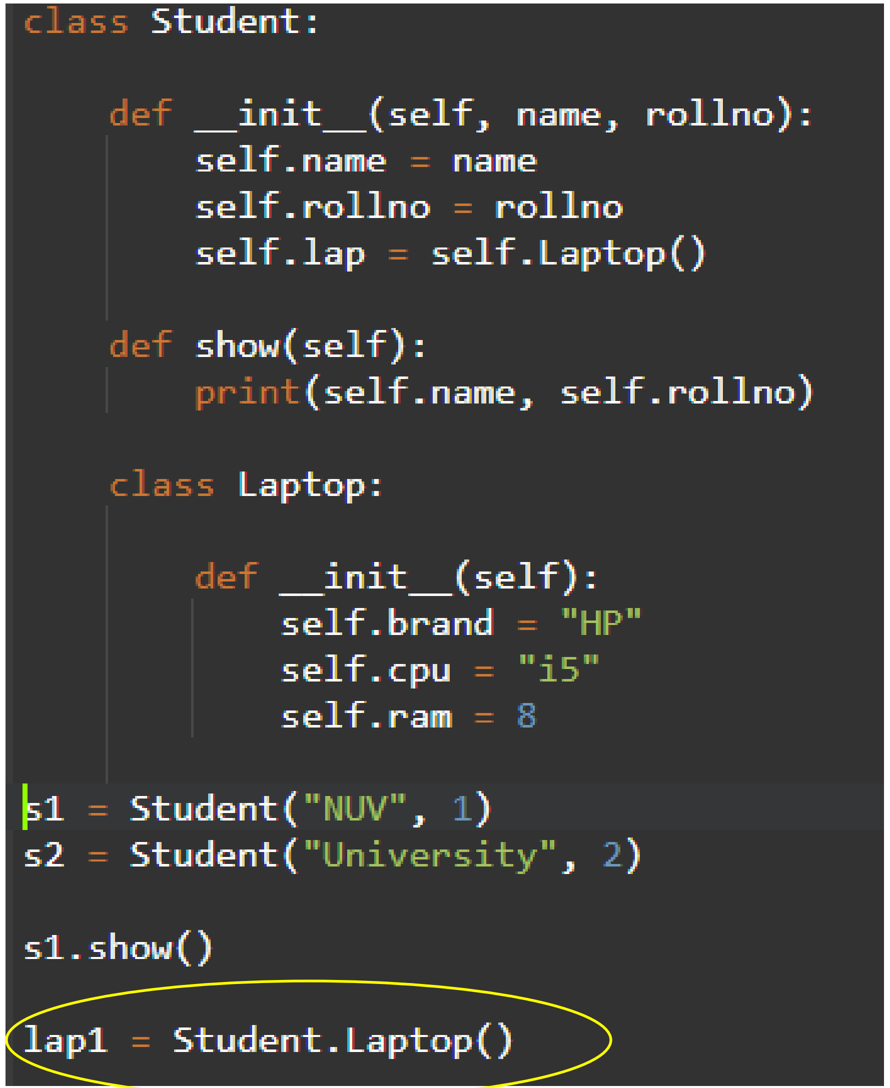


```python
class Student:

    def __init__(self, name, rollno):
        self.name = name
        self.rollno = rollno
        self.lap = self.Laptop()

    def show(self):
        print(self.name, self.rollno)

    class Laptop:

        def __init__(self):
            self.brand = "HP"
            self.cpu = "i5"
            self.ram = 8

s1 = Student("NUV", 1)
s2 = Student("University", 2)

s1.show()

lap1 = s1.lap
lap2 = s2.lap

print(id(lap1))
print(id(lap2))
```



```
NUV 1
139891416265536
139891415902000
```

Inner Class

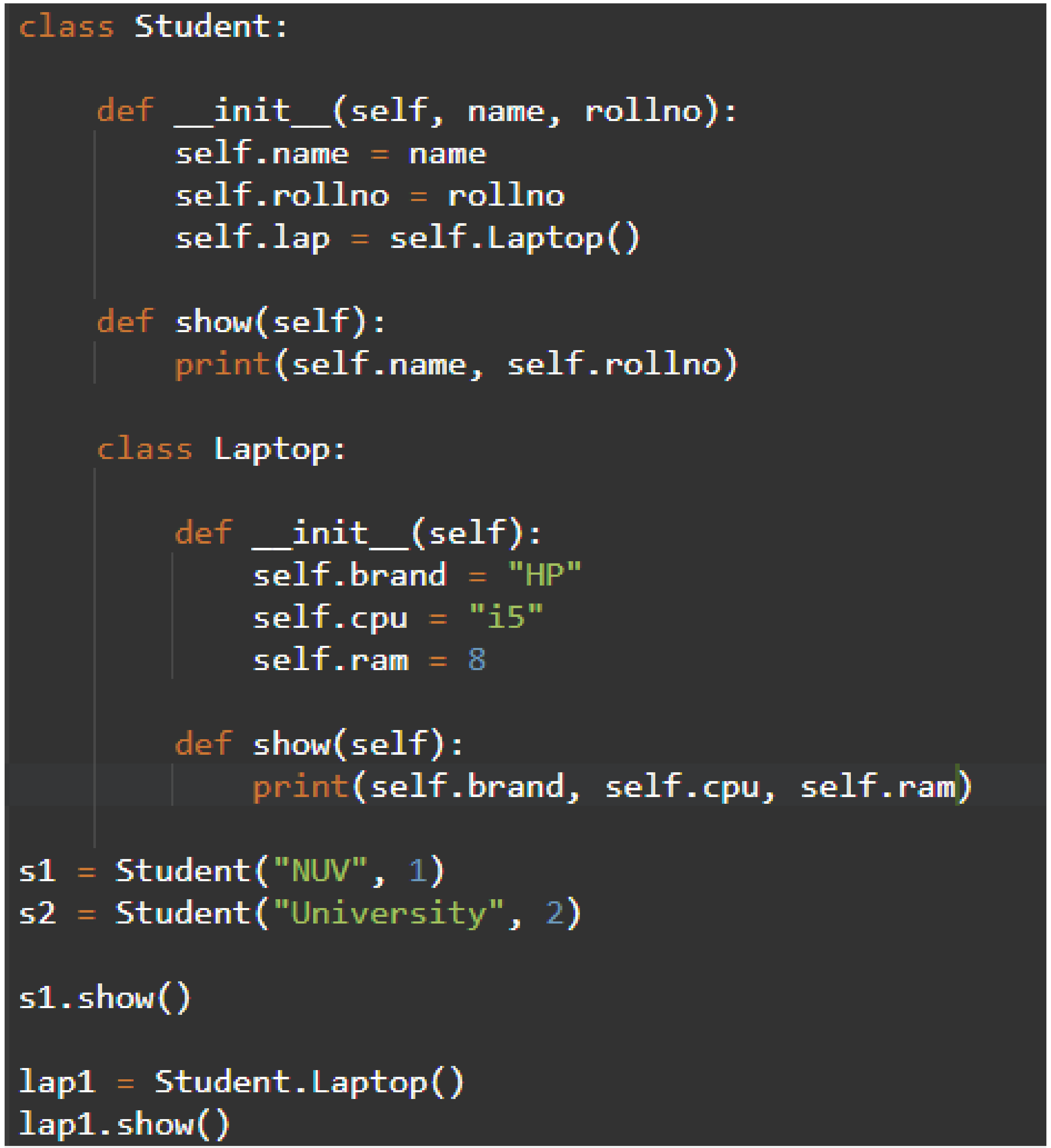


```python
class Student:

    def __init__(self, name, rollno):
        self.name = name
        self.rollno = rollno
        self.lap = self.Laptop()

    def show(self):
        print(self.name, self.rollno)

    class Laptop:

        def __init__(self):
            self.brand = "HP"
            self.cpu = "i5"
            self.ram = 8

s1 = Student("NUV", 1)
s2 = Student("University", 2)

s1.show()

lap1 = Student.Laptop()
```

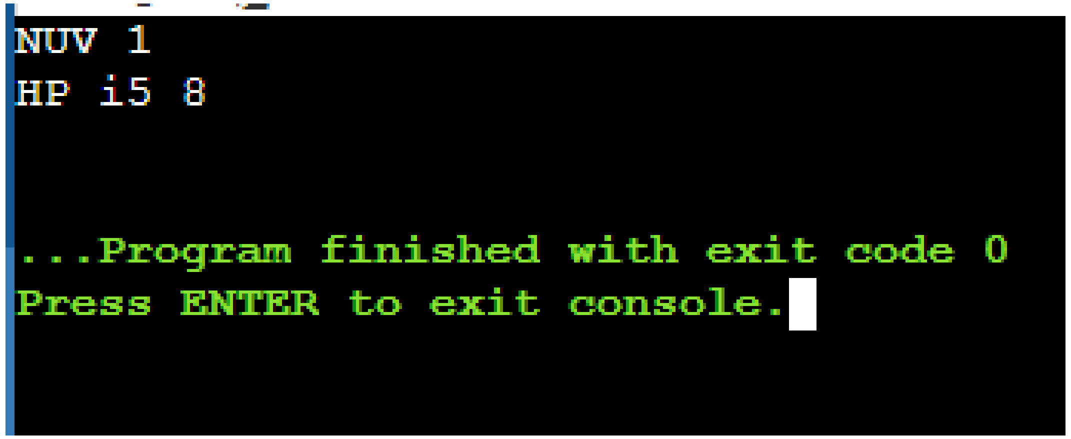
- You can create the object of inner class inside the outer class

OR

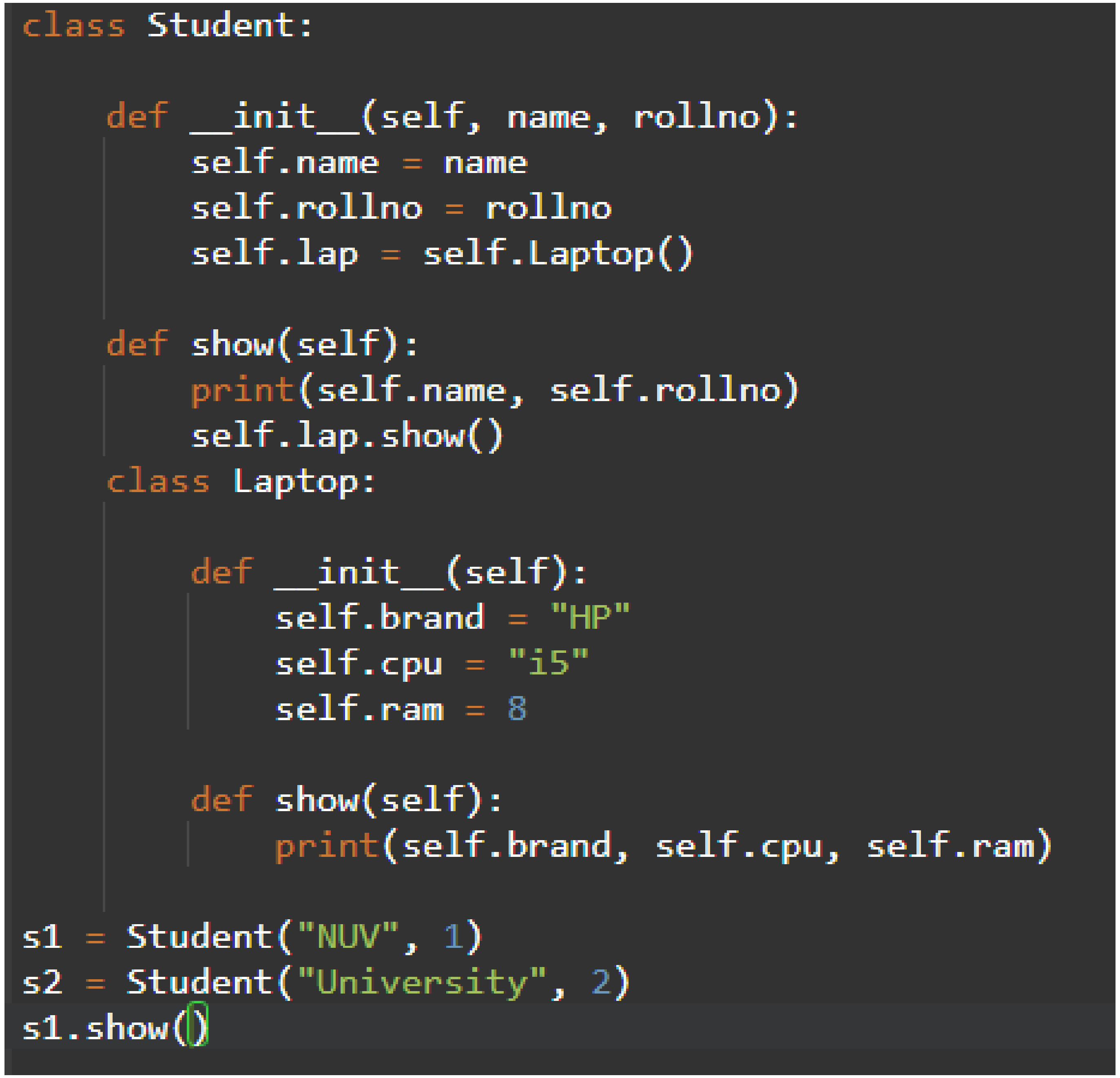
- You can create the object of inner class outside the outer class provided you use outer class name to call it



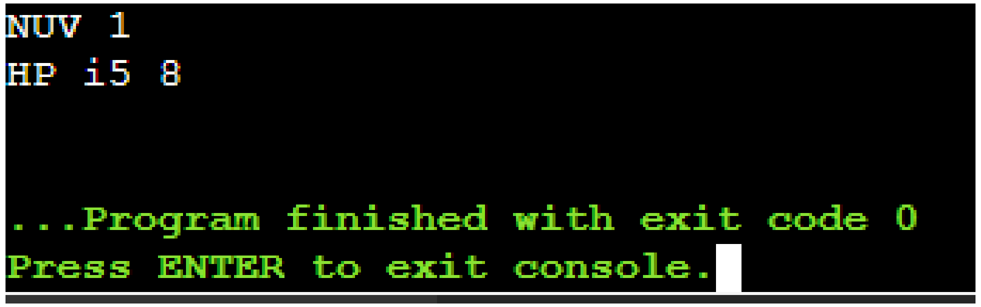
```python
class Student:

    def __init__(self, name, rollno):
        self.name = name
        self.rollno = rollno
        self.lap = self.Laptop()

    def show(self):
        print(self.name, self.rollno)

    class Laptop:

        def __init__(self):
            self.brand = "HP"
            self.cpu = "i5"
            self.ram = 8

        def show(self):
            print(self.brand, self.cpu, self.ram)

s1 = Student("NUV", 1)
s2 = Student("University", 2)

s1.show()

lap1 = Student.Laptop()
lap1.show()
```



```
NUV 1
HP i5 8


...Program finished with exit code 0
Press ENTER to exit console.
```



```
class Student:

    def __init__(self, name, rollno):
        self.name = name
        self.rollno = rollno
        self.lap = self.Laptop()

    def show(self):
        print(self.name, self.rollno)
        self.lap.show()
    class Laptop:

        def __init__(self):
            self.brand = "HP"
            self.cpu = "i5"
            self.ram = 8

        def show(self):
            print(self.brand, self.cpu, self.ram)

s1 = Student("NUV", 1)
s2 = Student("University", 2)
s1.show()
```

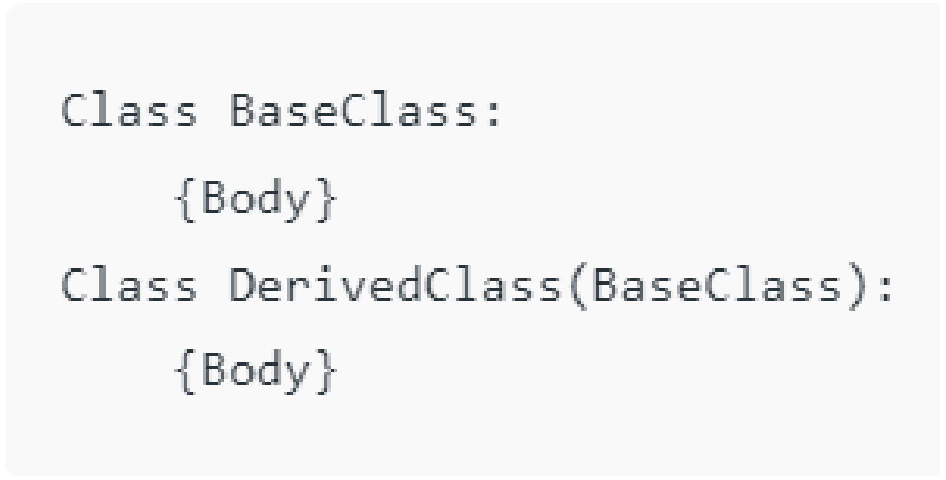


```
NUV 1
HP i5 8


...Program finished with exit code 0
Press ENTER to exit console.
```

Inheritance

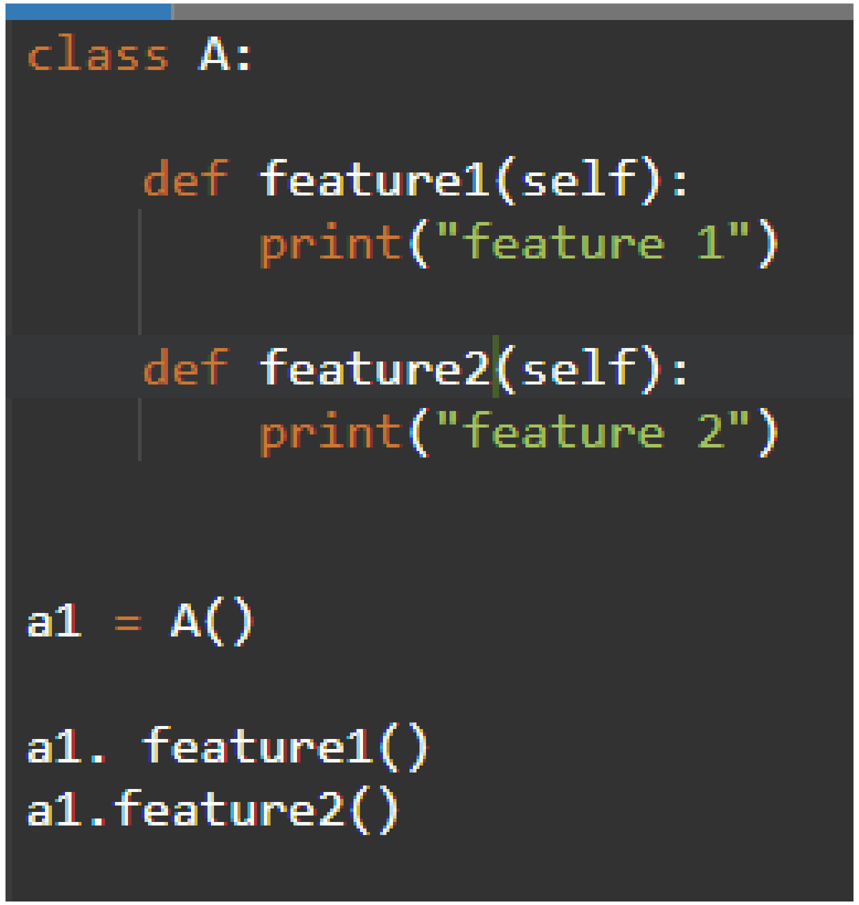
- Inheritance is an important aspect of the object-oriented paradigm.
- Inheritance provides code reusability to the program because we can use an existing class to create a new class instead of creating it from scratch.
- In inheritance, the child class acquires the properties and can access all the data members and functions defined in the parent class.
- A child class can also provide its specific implementation to the functions of the parent class.

- **Syntax :**

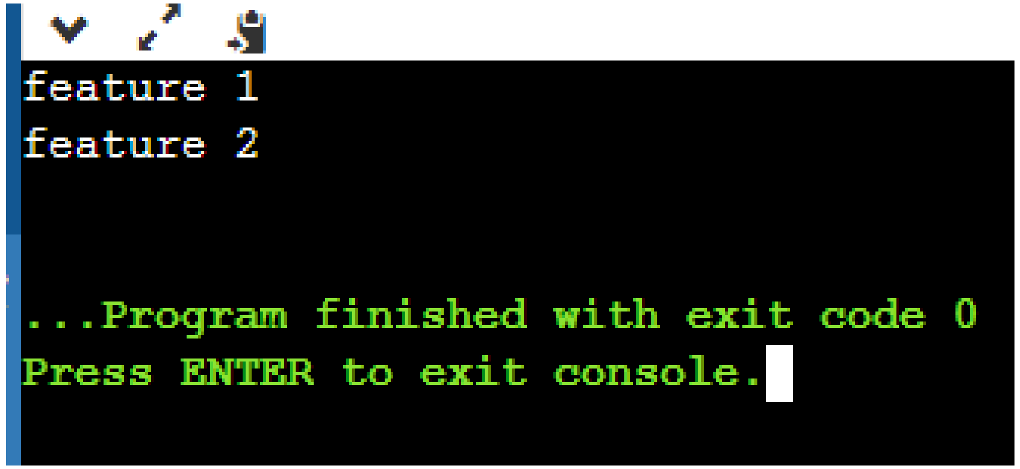


```
Class BaseClass:
    {Body}
Class DerivedClass(BaseClass):
    {Body}
```

Inheritance

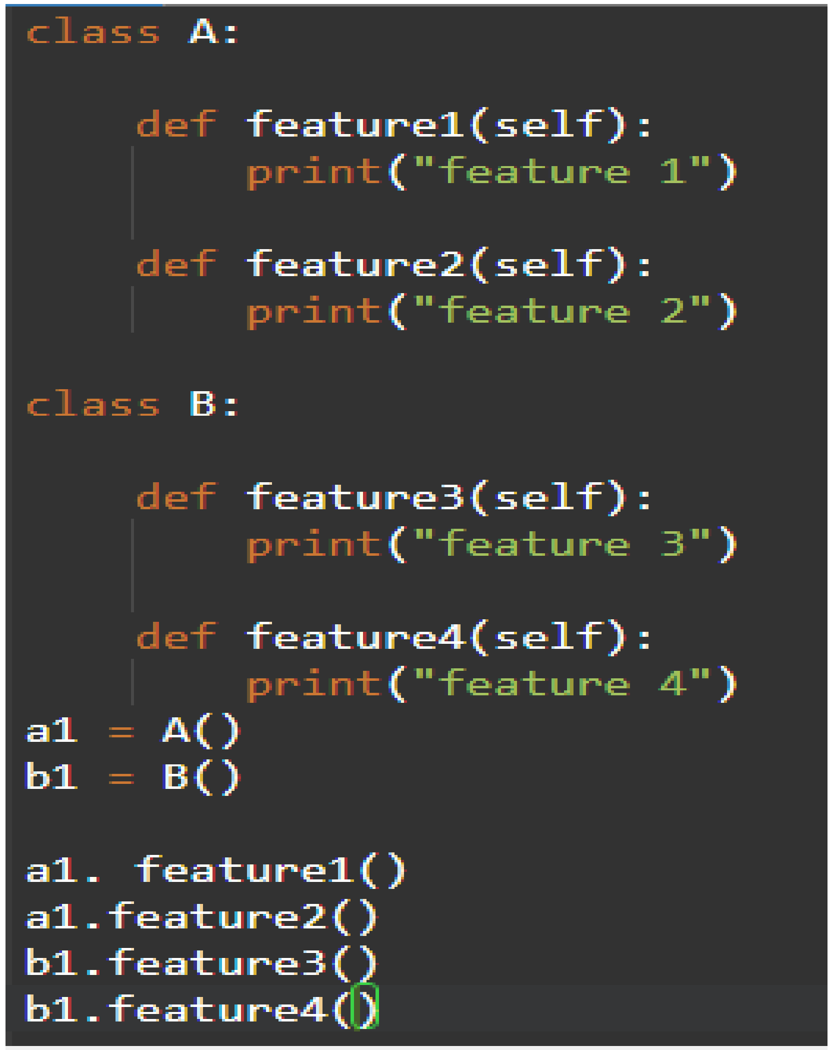


```
class A:

    def feature1(self):
        print("feature 1")

    def feature2(self):
        print("feature 2")


a1 = A()

a1. feature1()
a1.feature2()
```

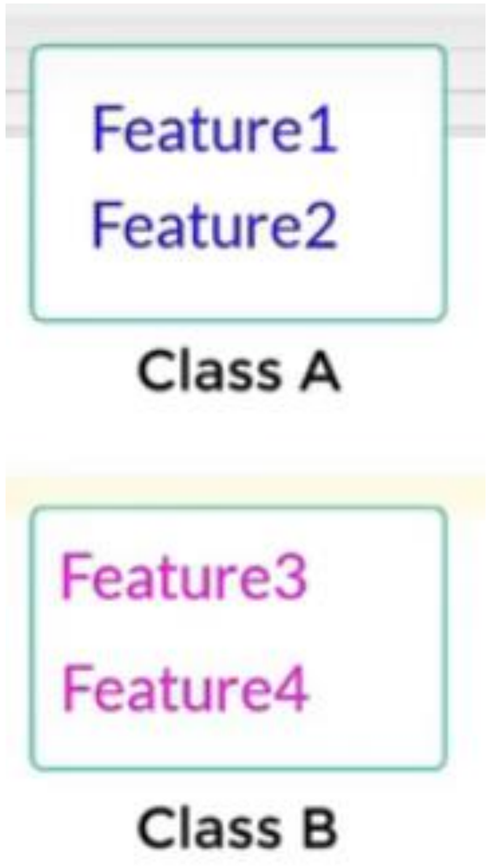


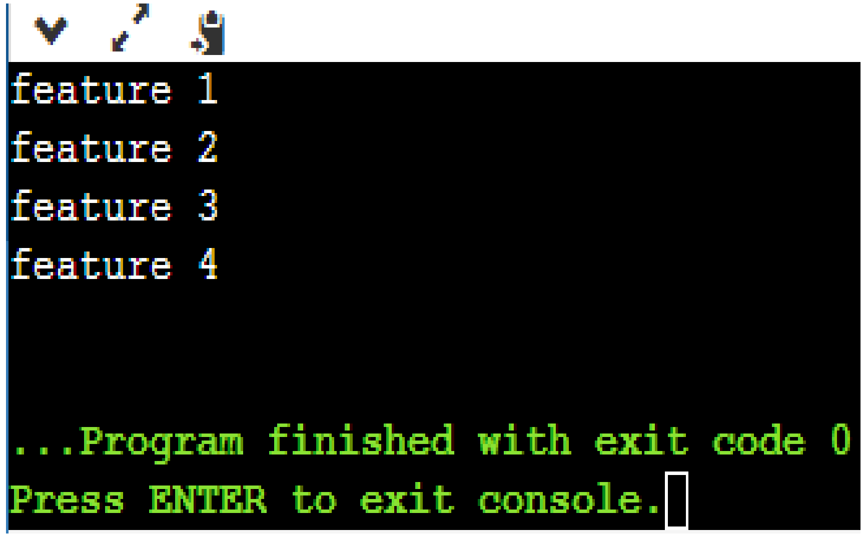
```
feature 1
feature 2


...Program finished with exit code 0
Press ENTER to exit console.
```

Inheritance



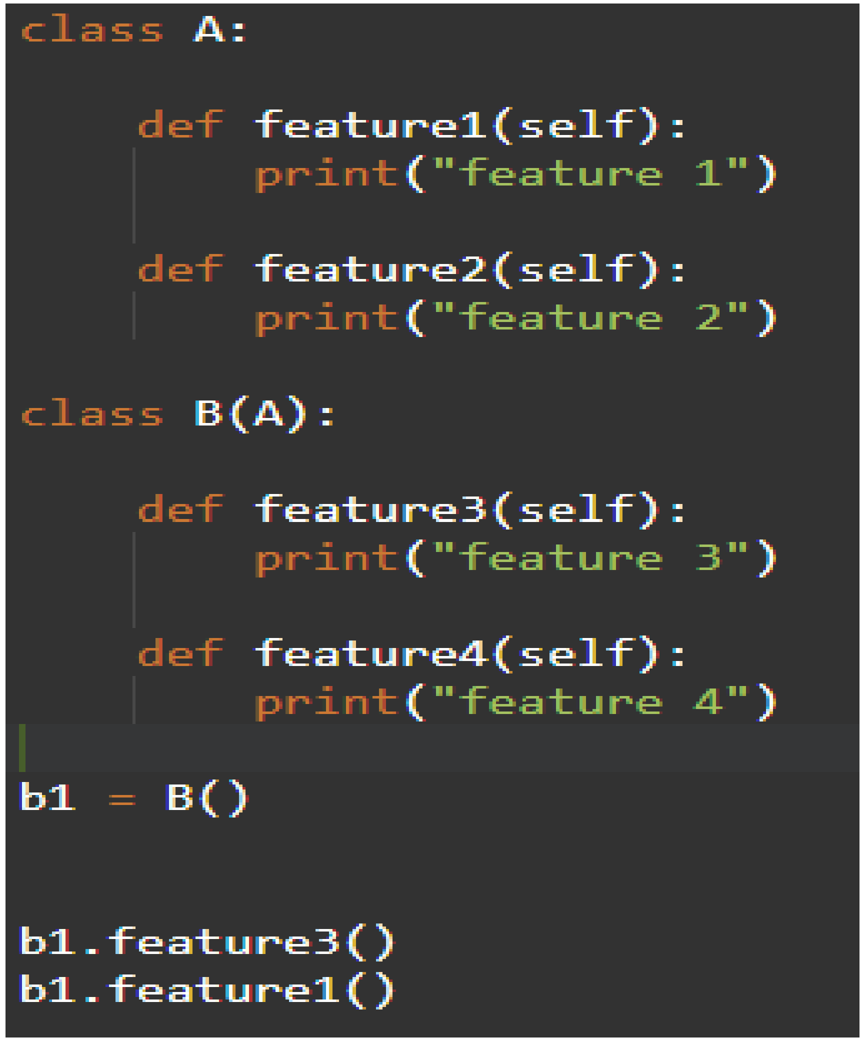
```
class A:

    def feature1(self):
        print("feature 1")

    def feature2(self):
        print("feature 2")

class B:

    def feature3(self):
        print("feature 3")

    def feature4(self):
        print("feature 4")
a1 = A()
b1 = B()

a1. feature1()
a1.feature2()
b1.feature3()
b1.feature4()
```



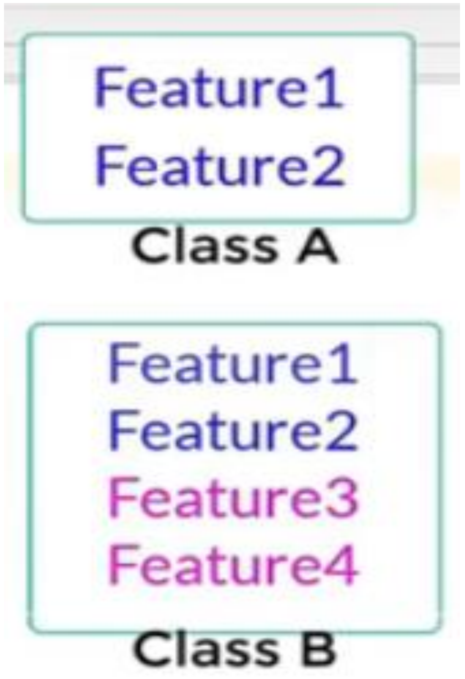


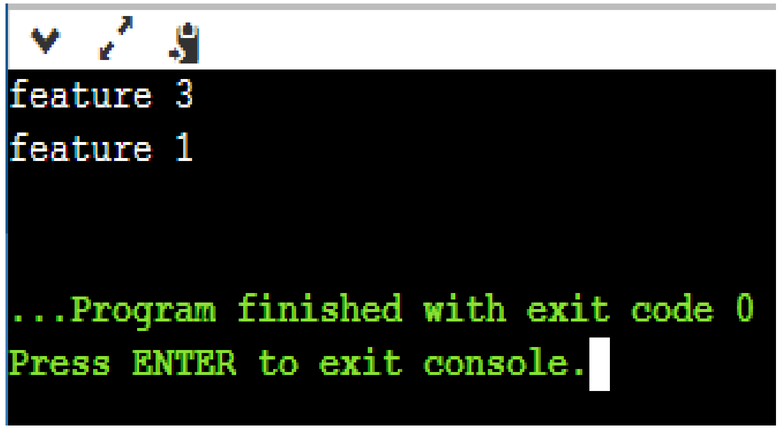
```
feature 1
feature 2
feature 3
feature 4


...Program finished with exit code 0
Press ENTER to exit console.
```

Inheritance



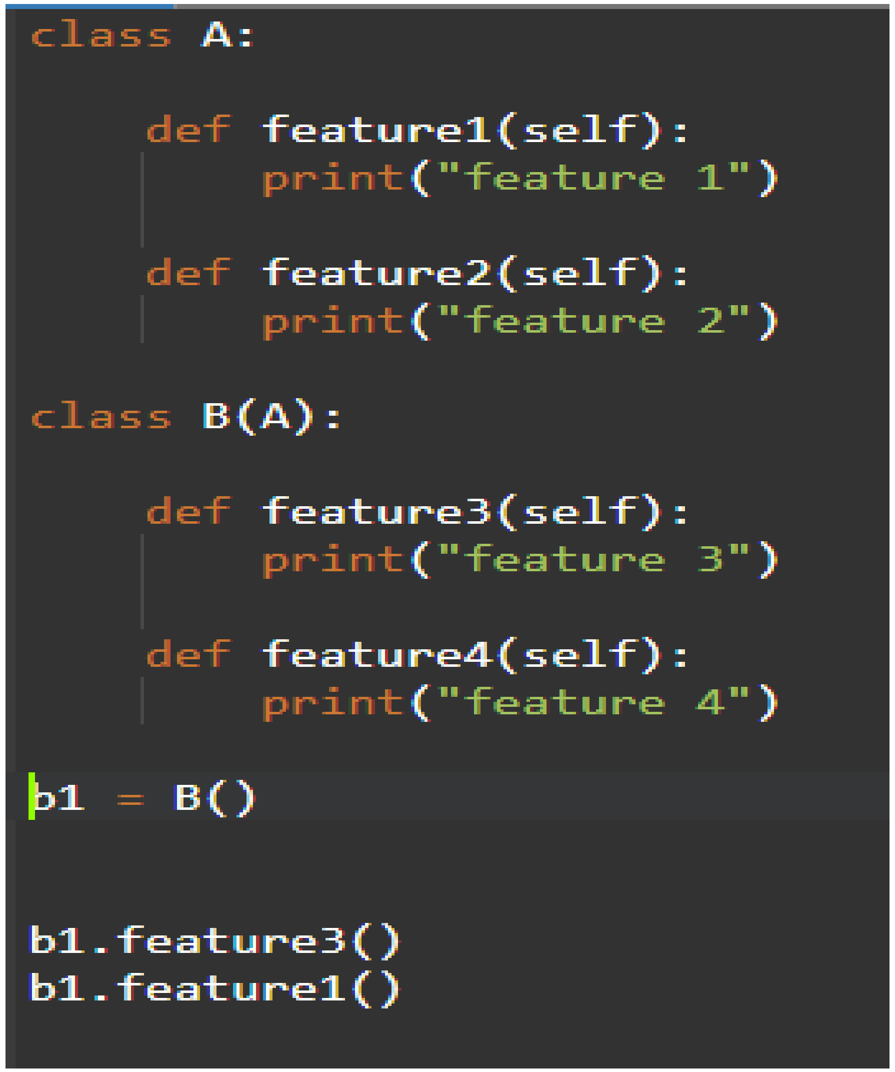
```
class A:

    def feature1(self):
        print("feature 1")

    def feature2(self):
        print("feature 2")

class B(A):

    def feature3(self):
        print("feature 3")

    def feature4(self):
        print("feature 4")

b1 = B()


b1.feature3()
b1.feature1()
```



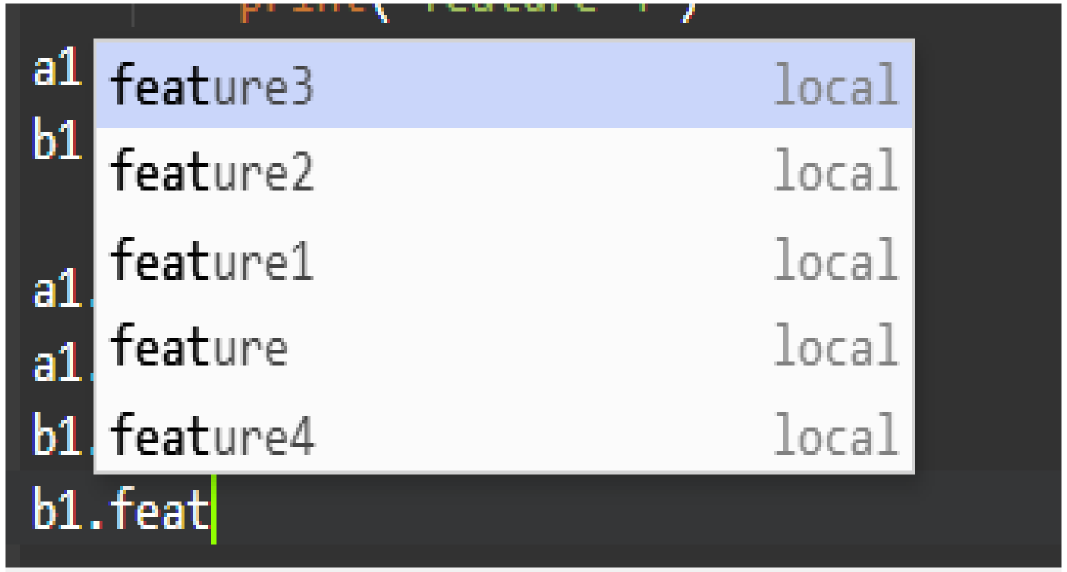


```
feature 3
feature 1


...Program finished with exit code 0
Press ENTER to exit console.
```

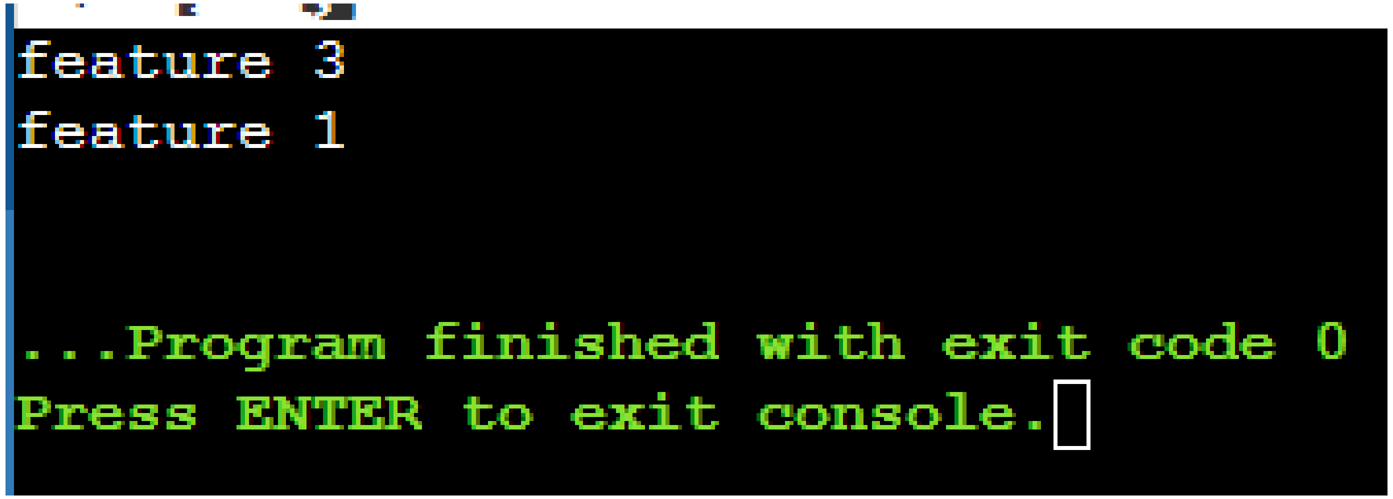
Inheritance



```
class A:

    def feature1(self):
        print("feature 1")

    def feature2(self):
        print("feature 2")

class B(A):

    def feature3(self):
        print("feature 3")

    def feature4(self):
        print("feature 4")

b1 = B()


b1.feature3()
b1.feature1()
```

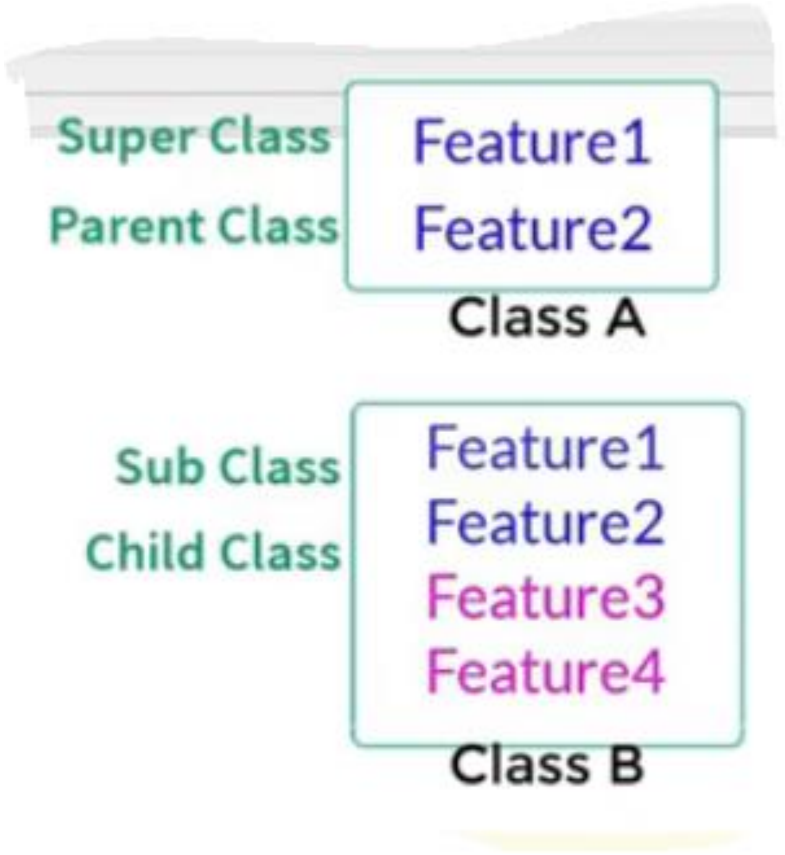


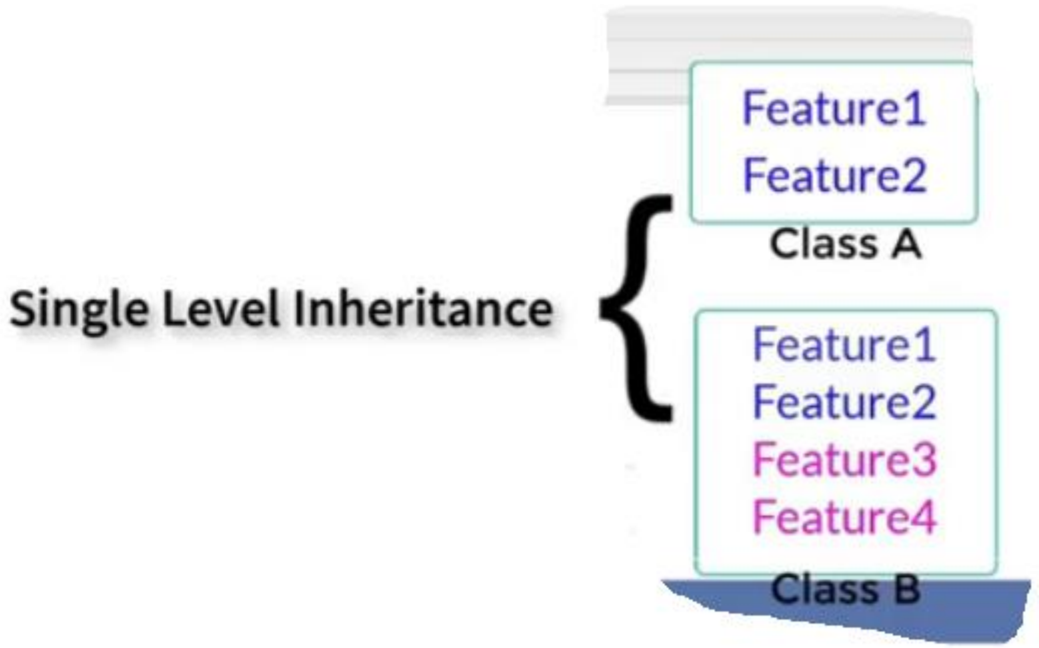
```
a1  feature3    local
b1  feature2    local
    feature1    local
a1  feature     local
a1  feature4    local
b1
b1.feat
```

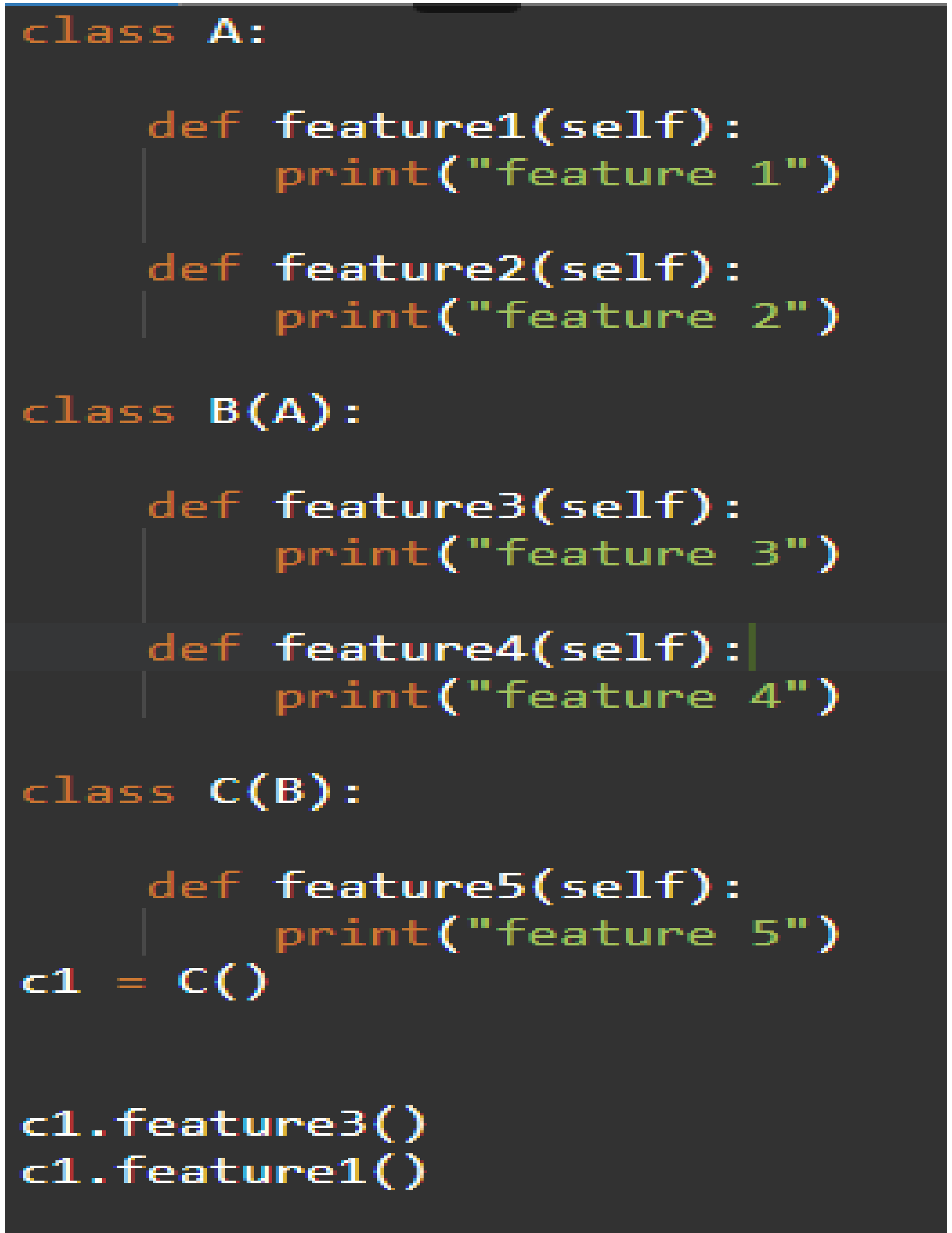


```
feature 3
feature 1


...Program finished with exit code 0
Press ENTER to exit console.
```

Inheritance

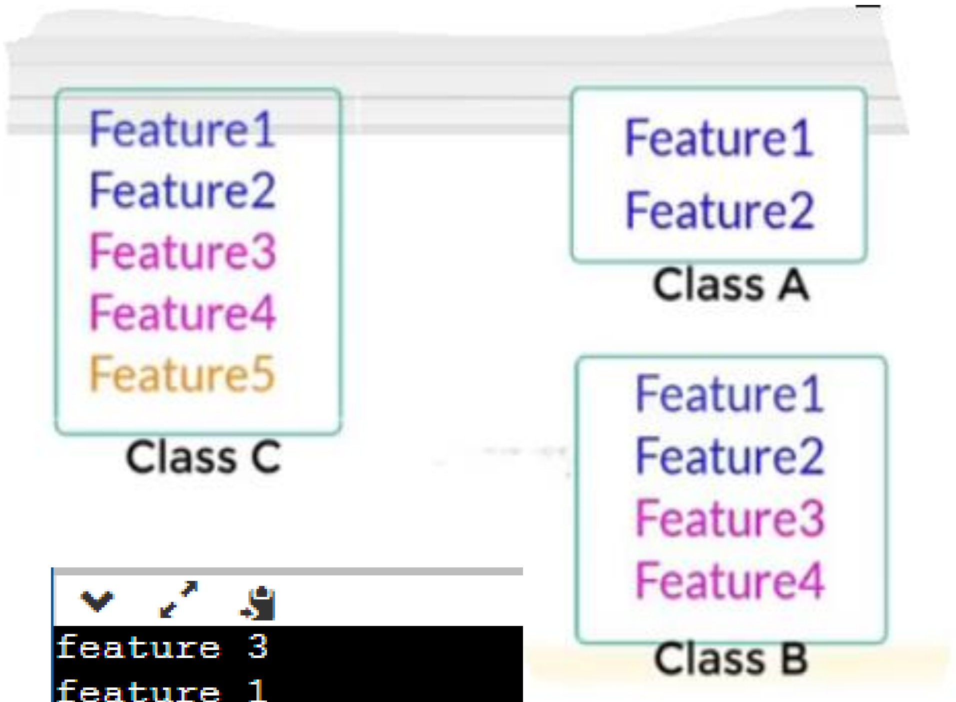


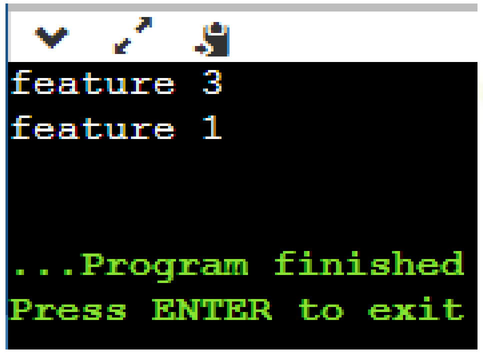




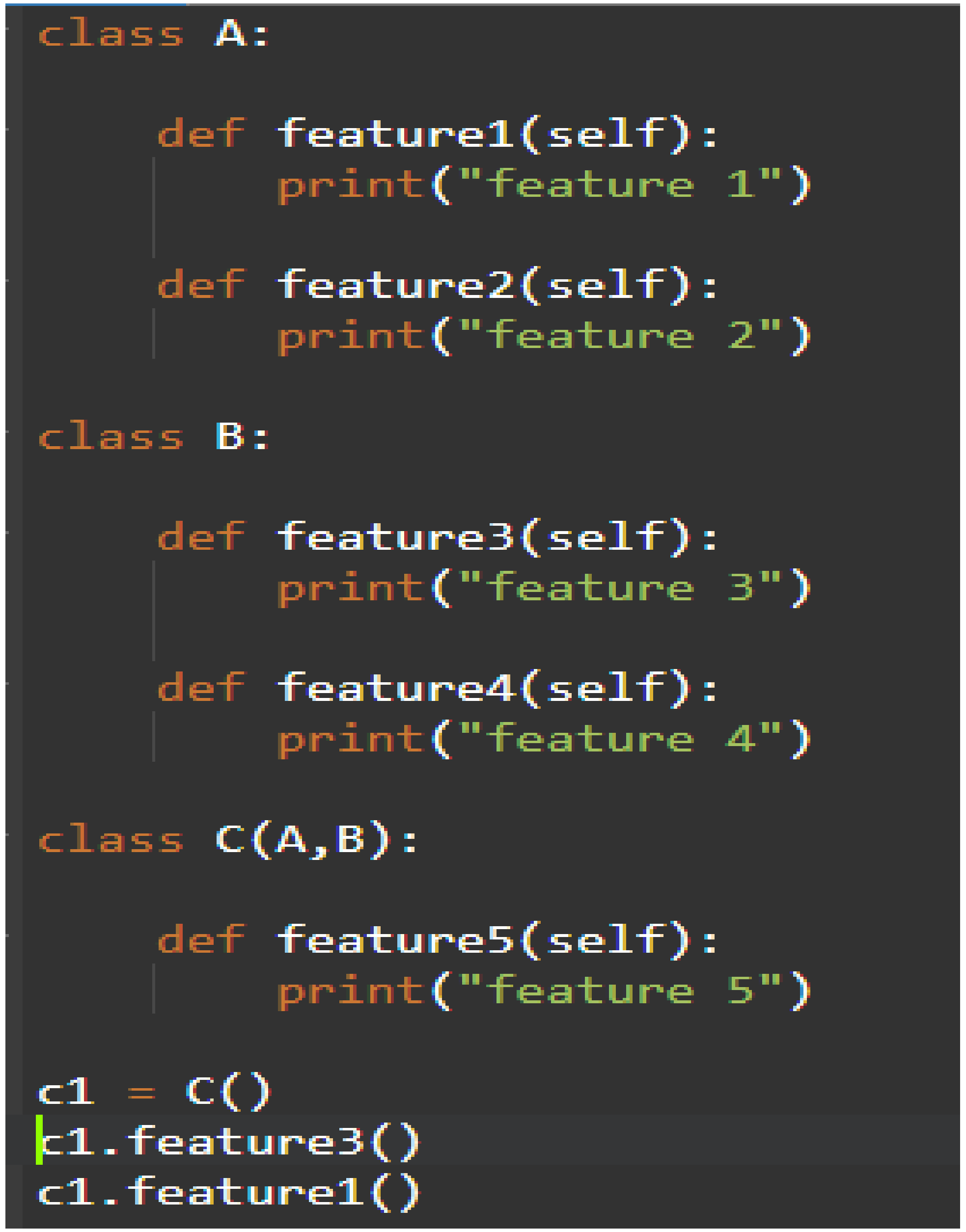
```
class A:

    def feature1(self):
        print("feature 1")

    def feature2(self):
        print("feature 2")

class B(A):

    def feature3(self):
        print("feature 3")

    def feature4(self):
        print("feature 4")

class C(B):

    def feature5(self):
        print("feature 5")
c1 = C()


c1.feature3()
c1.feature1()
```

- **Multi Level Inheritance**



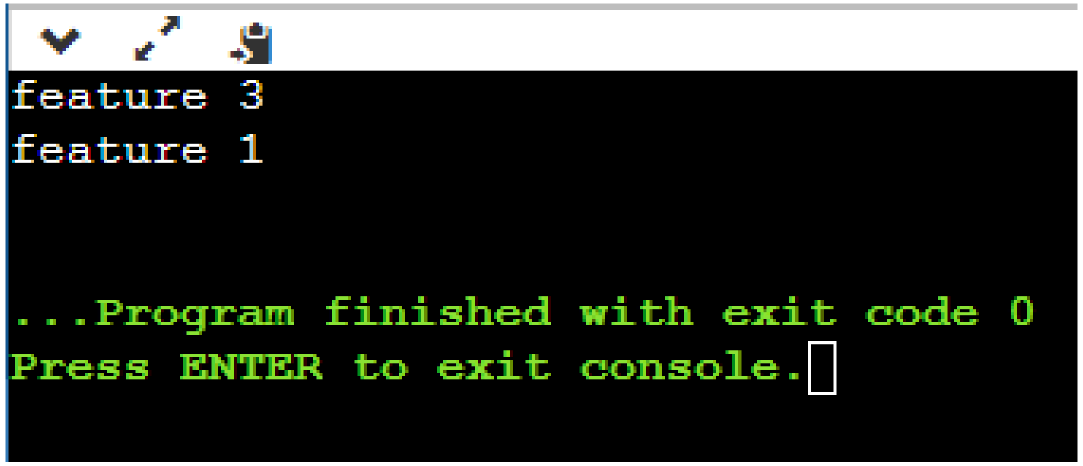


```
feature 3
feature 1


...Program finished
Press ENTER to exit
```



```
class A:

    def feature1(self):
        print("feature 1")

    def feature2(self):
        print("feature 2")

class B:

    def feature3(self):
        print("feature 3")

    def feature4(self):
        print("feature 4")

class C(A,B):

    def feature5(self):
        print("feature 5")

c1 = C()
c1.feature3()
c1.feature1()
```

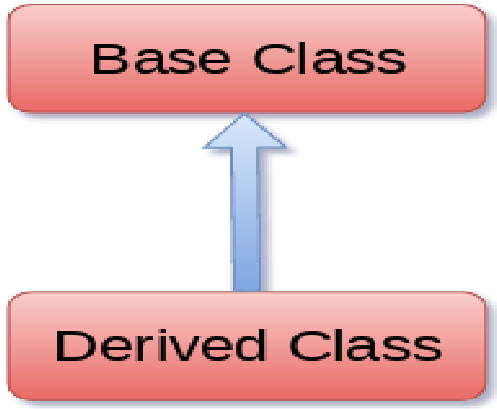
- **Multiple Inheritance**



```
feature 3
feature 1


...Program finished with exit code 0
Press ENTER to exit console.
```

Single Level Inheritance

- When a child class inherits from only one parent class, it is called single inheritance.
- A class can inherit multiple classes by mentioning all of them inside the bracket.



- **Syntax**

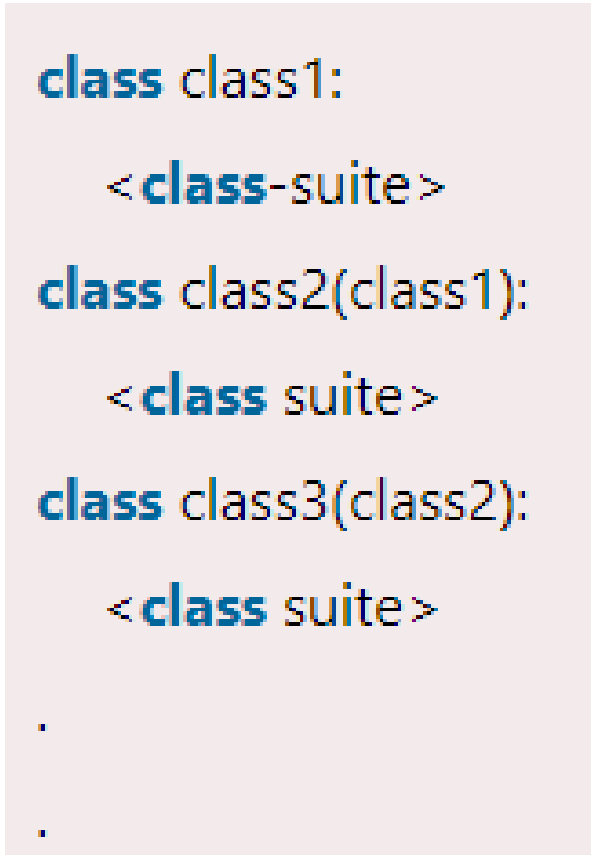
```
class derived-class(base class):
    <class-suite>
```

```
class derive-class(<base class 1>, <base class 2>, ..... <base class n>):
   <class - suite>
```

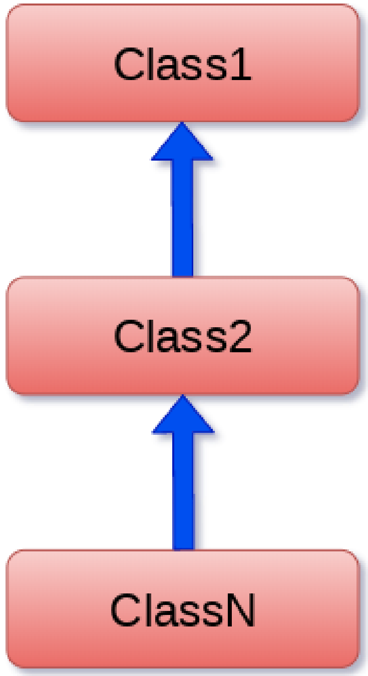
Multilevel Inheritance

- Multi-level inheritance is achieved when a derived class inherits another derived class.
- There is no limit on the number of levels up to which, the multi-level inheritance is achieved in python.

- **Syntax**

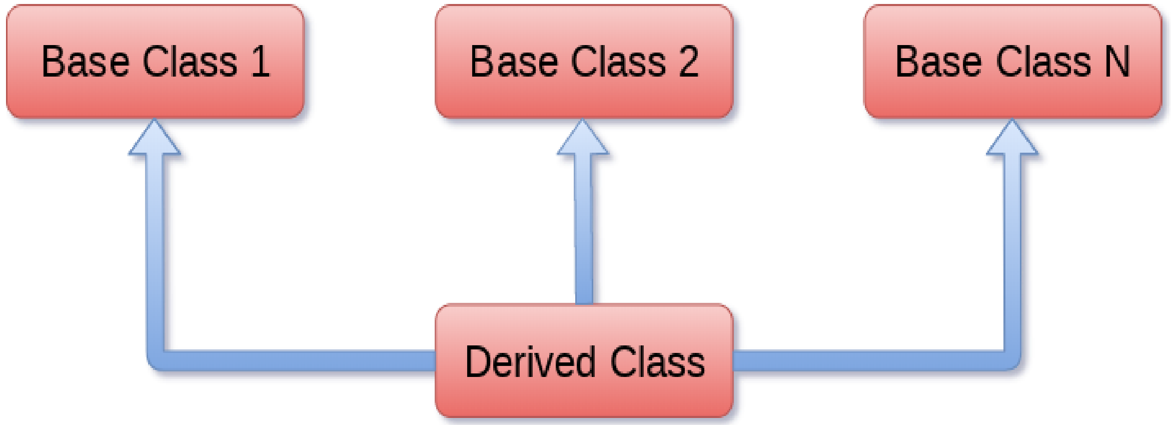


```
class class1:
    <class-suite>
class class2(class1):
    <class suite>
class class3(class2):
    <class suite>
.
.
```

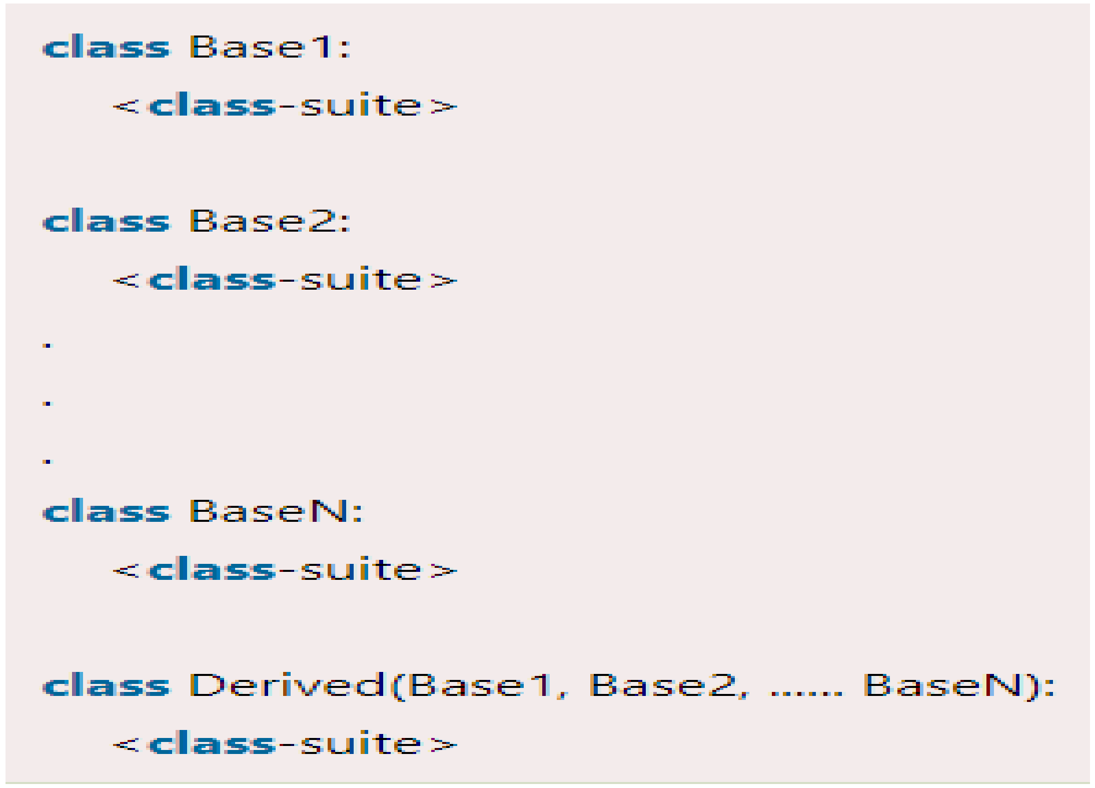


Multiple Inheritance

- When a child class inherits from multiple parent classes, it is called multiple inheritances.



- **Syntax**



```
class Base1:
    <class-suite>

class Base2:
    <class-suite>
.
.
.
class BaseN:
    <class-suite>

class Derived(Base1, Base2, ...... BaseN):
    <class-suite>
```

Inheritance

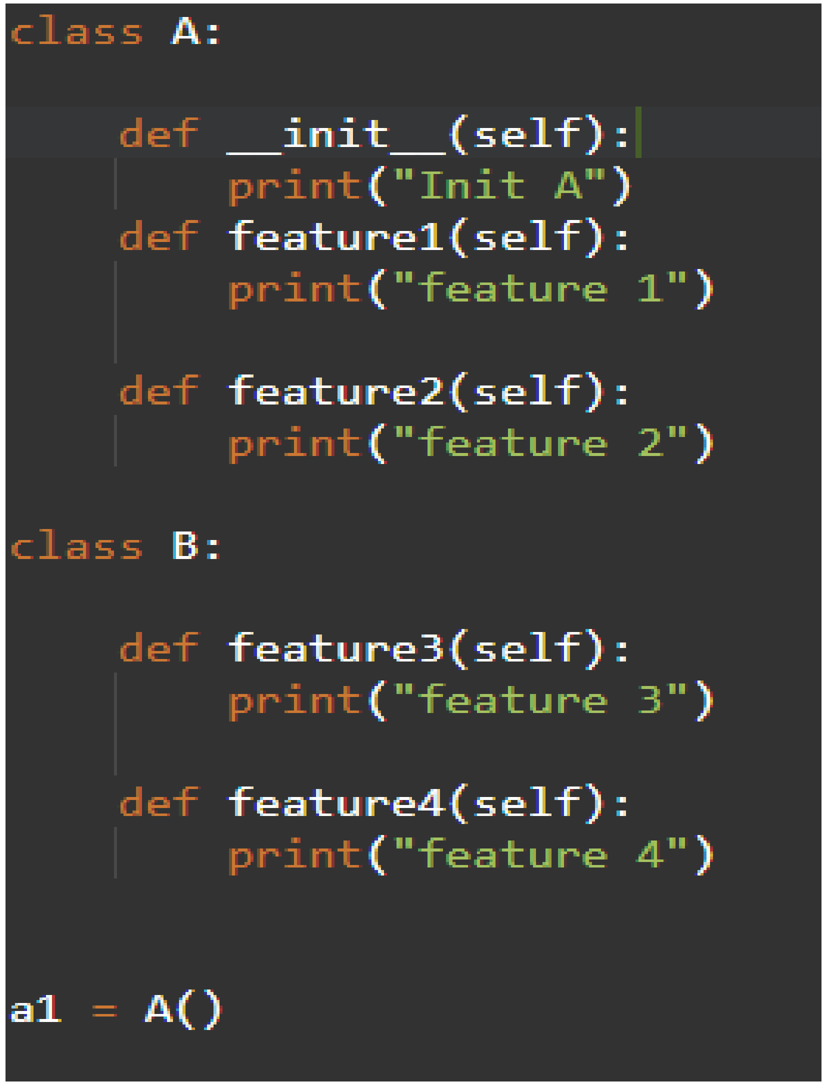
- **NOTE:* - Sub class can access all the features of Super class**

  BUT

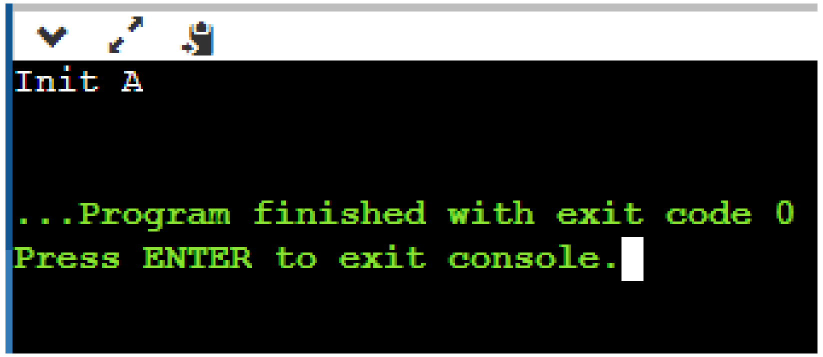
  Super class cannot access any features of Sub class

Constructor in Inheritance

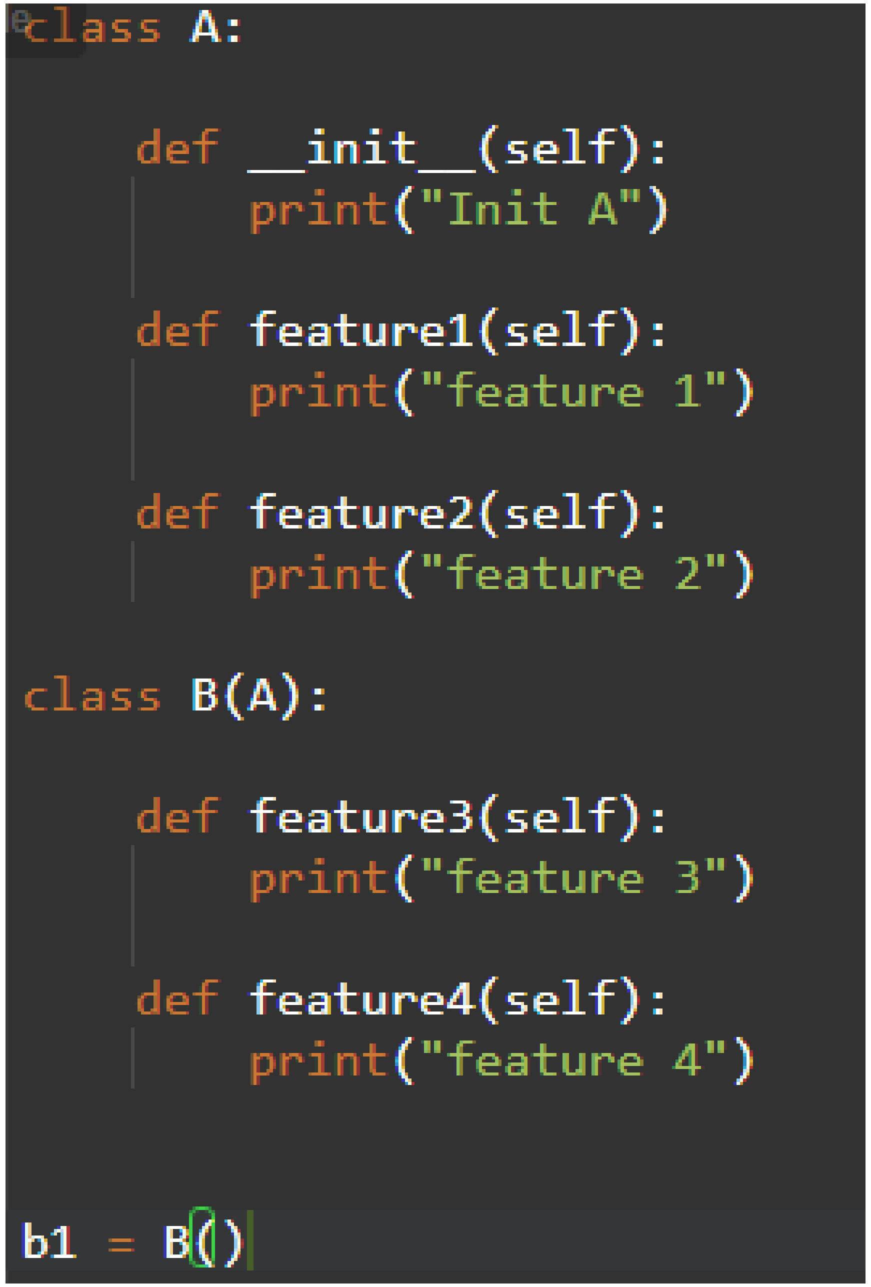
Method Resolution Order



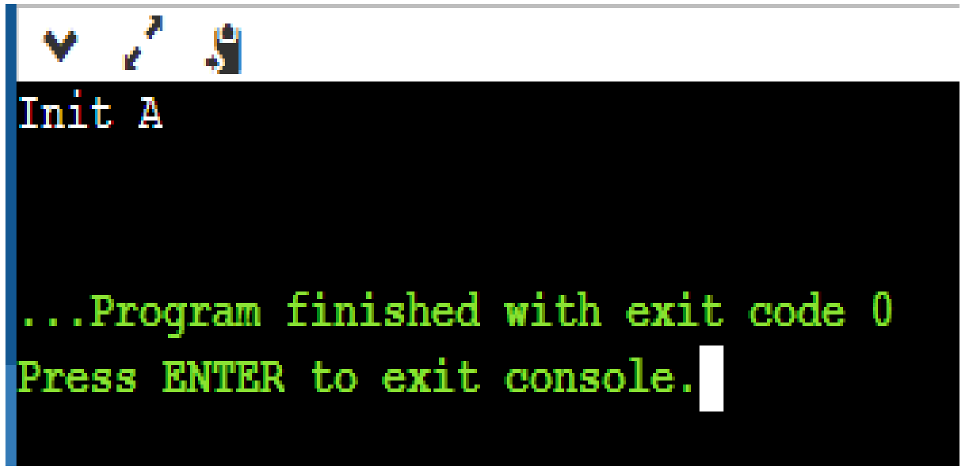
```python
class A:

    def __init__(self):
        print("Init A")
    def feature1(self):
        print("feature 1")

    def feature2(self):
        print("feature 2")

class B:

    def feature3(self):
        print("feature 3")

    def feature4(self):
        print("feature 4")


a1 = A()
```



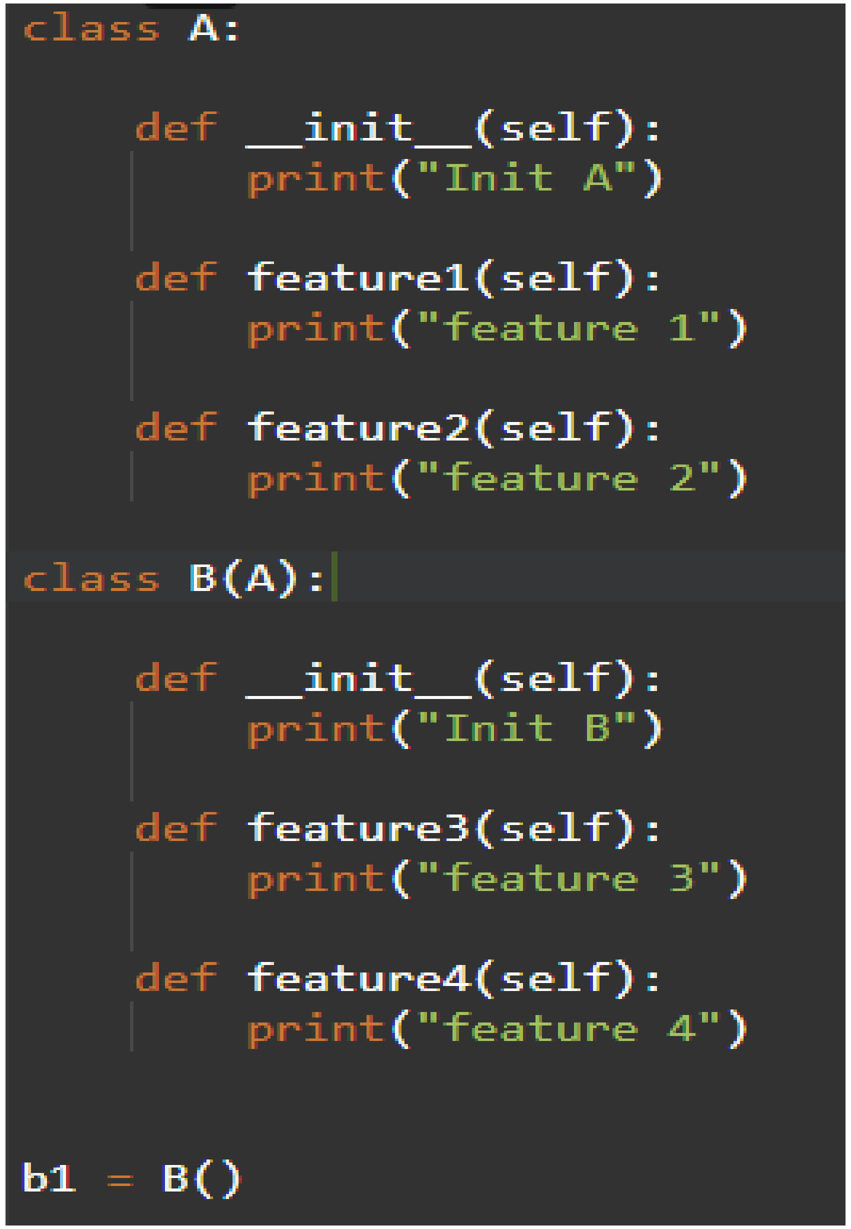
```
Init A


...Program finished with exit code 0
Press ENTER to exit console.
```



```
class A:

    def __init__(self):
        print("Init A")

    def feature1(self):
        print("feature 1")

    def feature2(self):
        print("feature 2")

class B(A):

    def feature3(self):
        print("feature 3")

    def feature4(self):
        print("feature 4")


b1 = B()
```

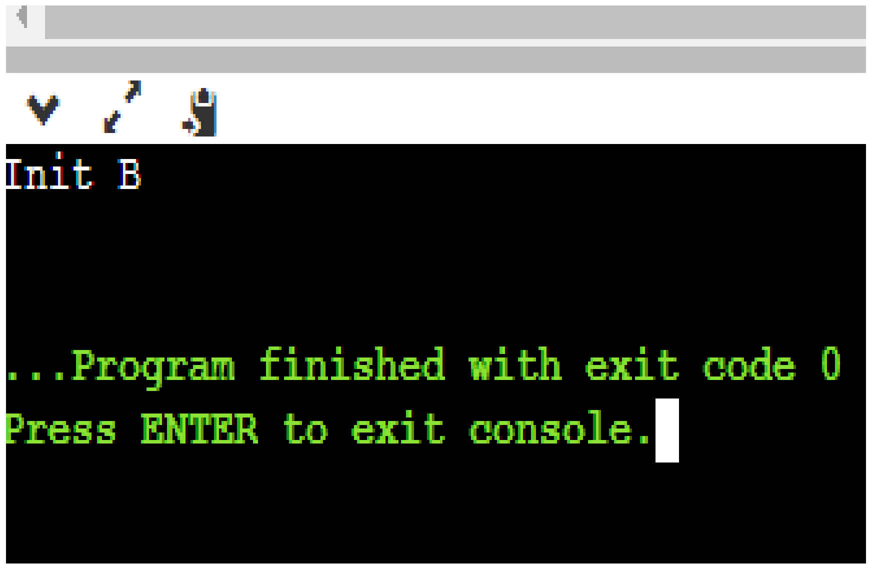


```
Init A



...Program finished with exit code 0
Press ENTER to exit console.
```

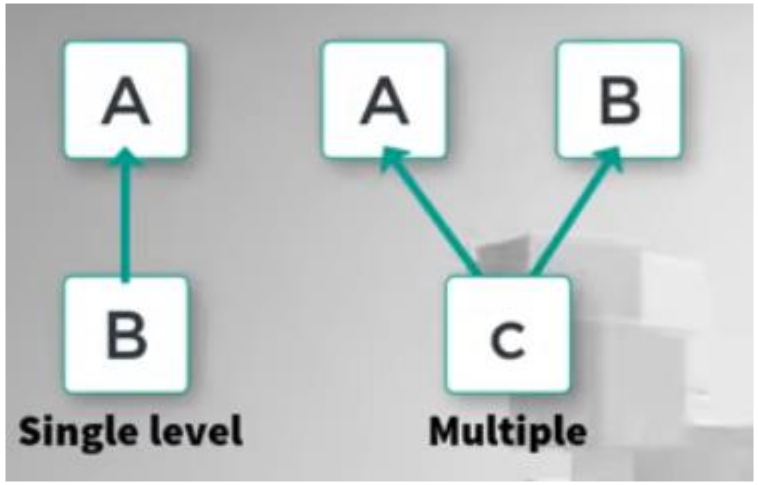


```python
class A:

    def __init__(self):
        print("Init A")

    def feature1(self):
        print("feature 1")

    def feature2(self):
        print("feature 2")

class B(A):

    def __init__(self):
        print("Init B")

    def feature3(self):
        print("feature 3")

    def feature4(self):
        print("feature 4")


b1 = B()
```

in A Init
?
in B Init
✓



```
Init B


...Program finished with exit code 0
Press ENTER to exit console.
```


A
B
Single level
A
B
C
Multiple