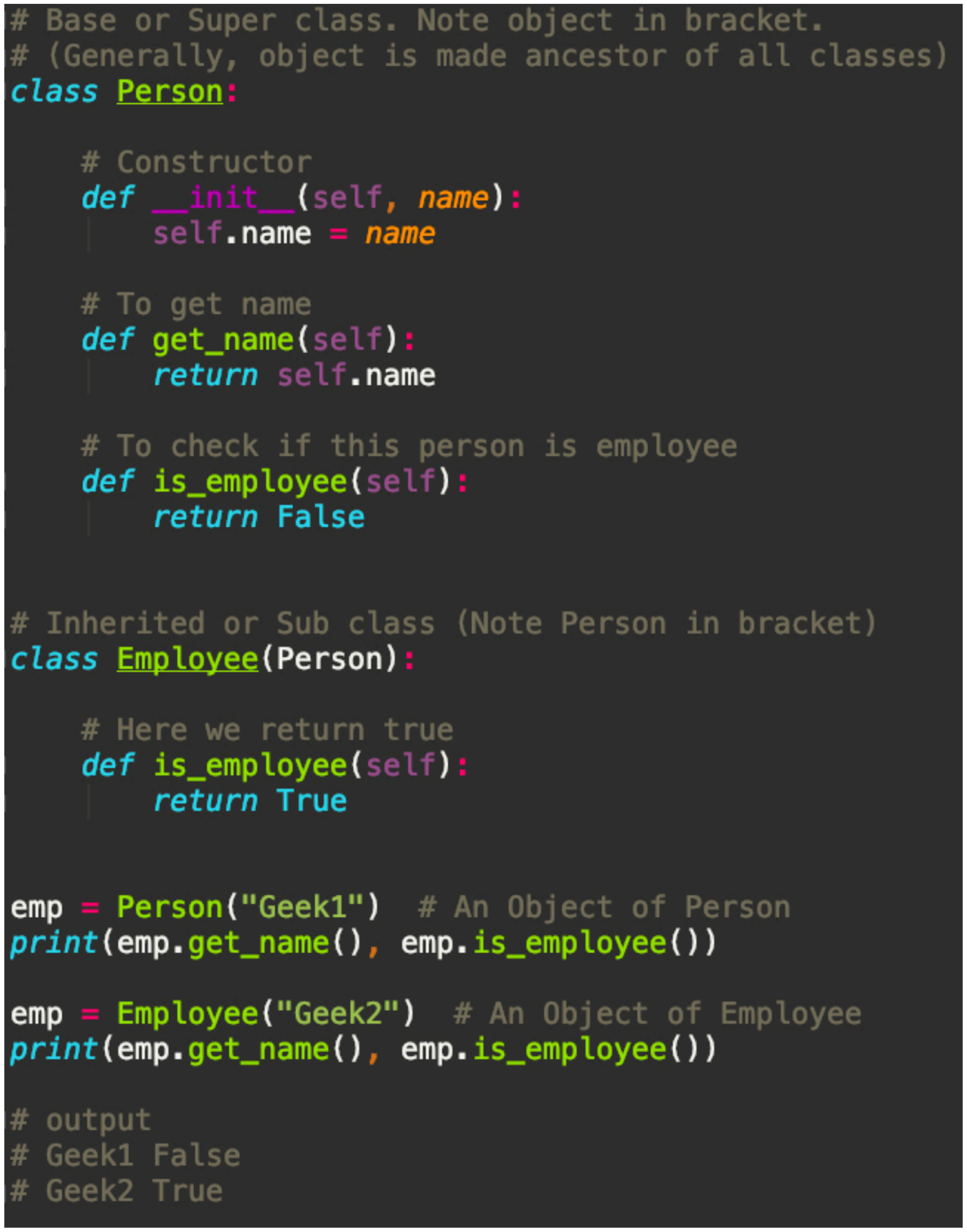
Python basics V: Advanced class topics

Class Inheritance

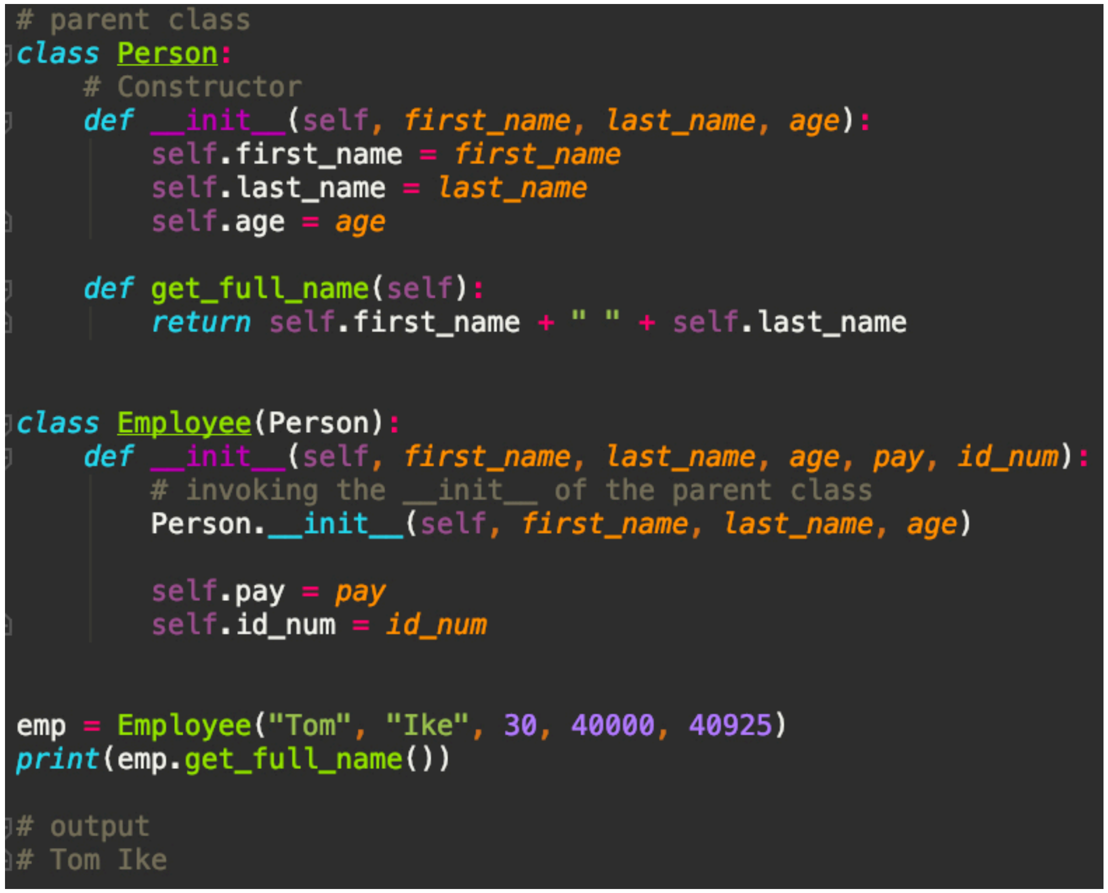
Inheritance is the capability of one class to derive or inherit the properties from some another class.



```python
# Base or Super class. Note object in bracket.
# (Generally, object is made ancestor of all classes)
class Person:

    # Constructor
    def __init__(self, name):
        self.name = name

    # To get name
    def get_name(self):
        return self.name

    # To check if this person is employee
    def is_employee(self):
        return False


# Inherited or Sub class (Note Person in bracket)
class Employee(Person):

    # Here we return true
    def is_employee(self):
        return True


emp = Person("Geek1")  # An Object of Person
print(emp.get_name(), emp.is_employee())

emp = Employee("Geek2")  # An Object of Employee
print(emp.get_name(), emp.is_employee())

# output
# Geek1 False
# Geek2 True
```

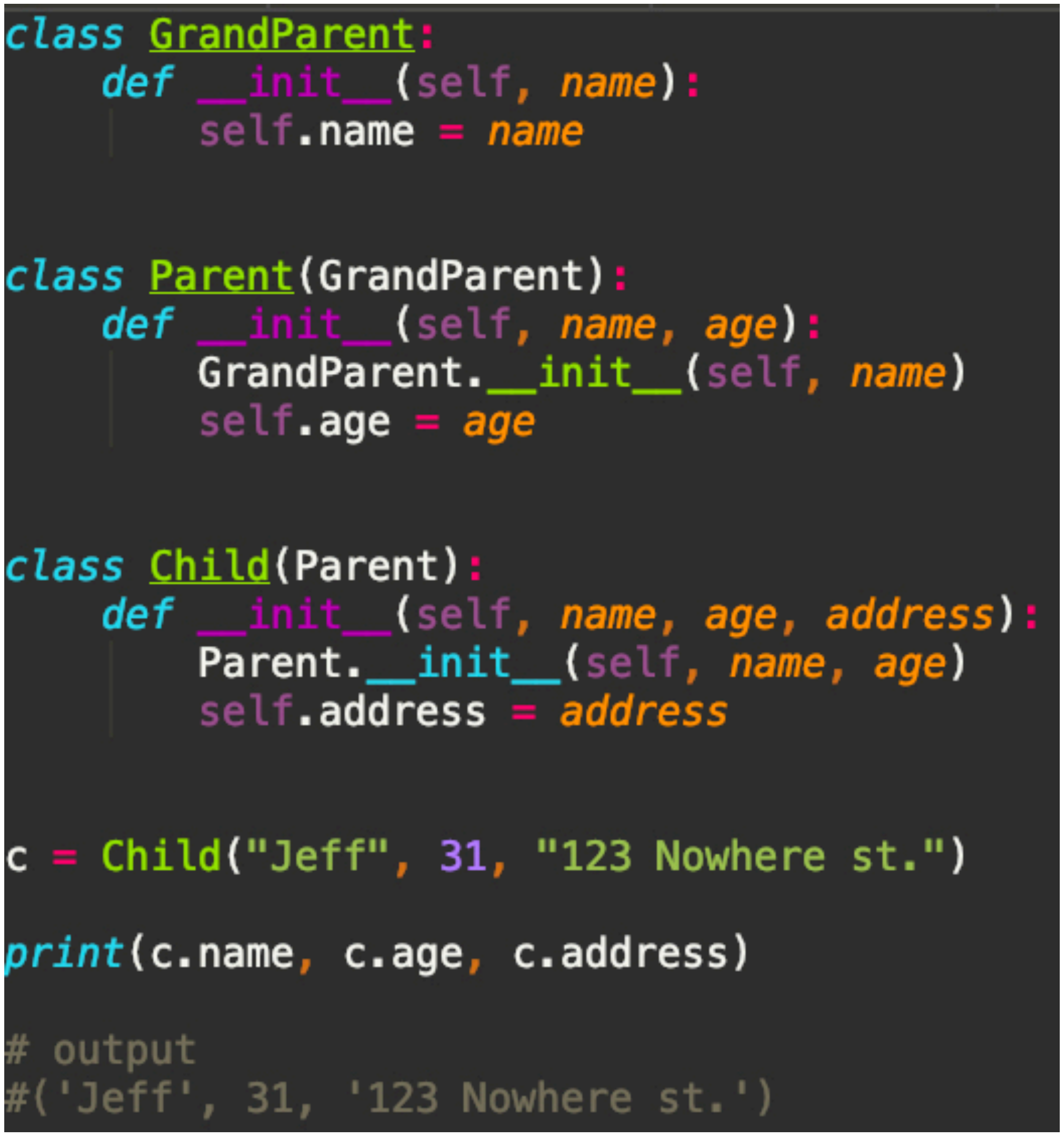
Class Inheritance

Calling the super or parent class constructor from the child class



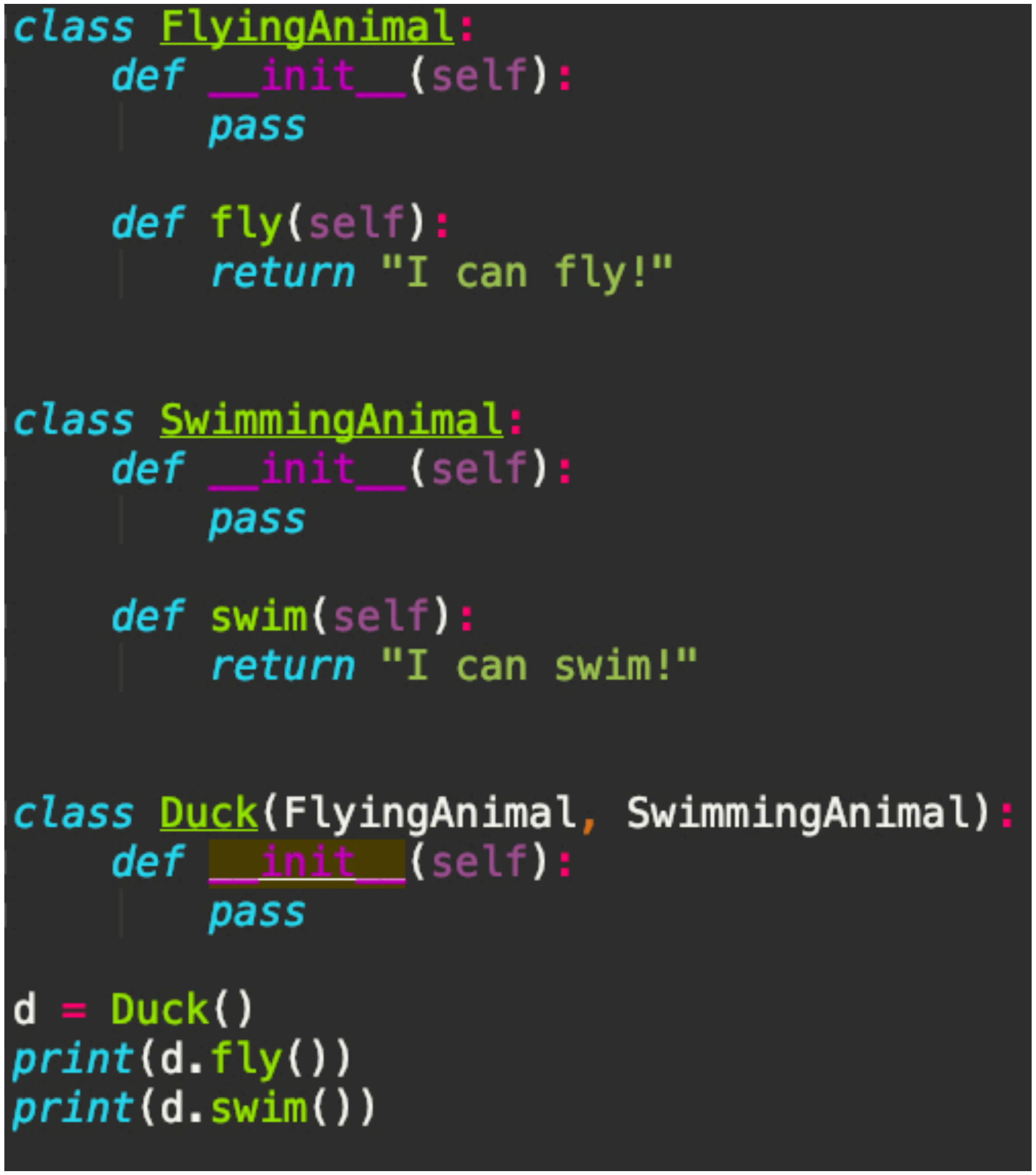
```python
# parent class
class Person:
    # Constructor
    def __init__(self, first_name, last_name, age):
        self.first_name = first_name
        self.last_name = last_name
        self.age = age

    def get_full_name(self):
        return self.first_name + " " + self.last_name


class Employee(Person):
    def __init__(self, first_name, last_name, age, pay, id_num):
        # invoking the __init__ of the parent class
        Person.__init__(self, first_name, last_name, age)

        self.pay = pay
        self.id_num = id_num


emp = Employee("Tom", "Ike", 30, 40000, 40925)
print(emp.get_full_name())

# output
# Tom Ike
```

Chaining Inheritance



```
class GrandParent:
    def __init__(self, name):
        self.name = name


class Parent(GrandParent):
    def __init__(self, name, age):
        GrandParent.__init__(self, name)
        self.age = age


class Child(Parent):
    def __init__(self, name, age, address):
        Parent.__init__(self, name, age)
        self.address = address


c = Child("Jeff", 31, "123 Nowhere st.")

print(c.name, c.age, c.address)

# output
#('Jeff', 31, '123 Nowhere st.')
```

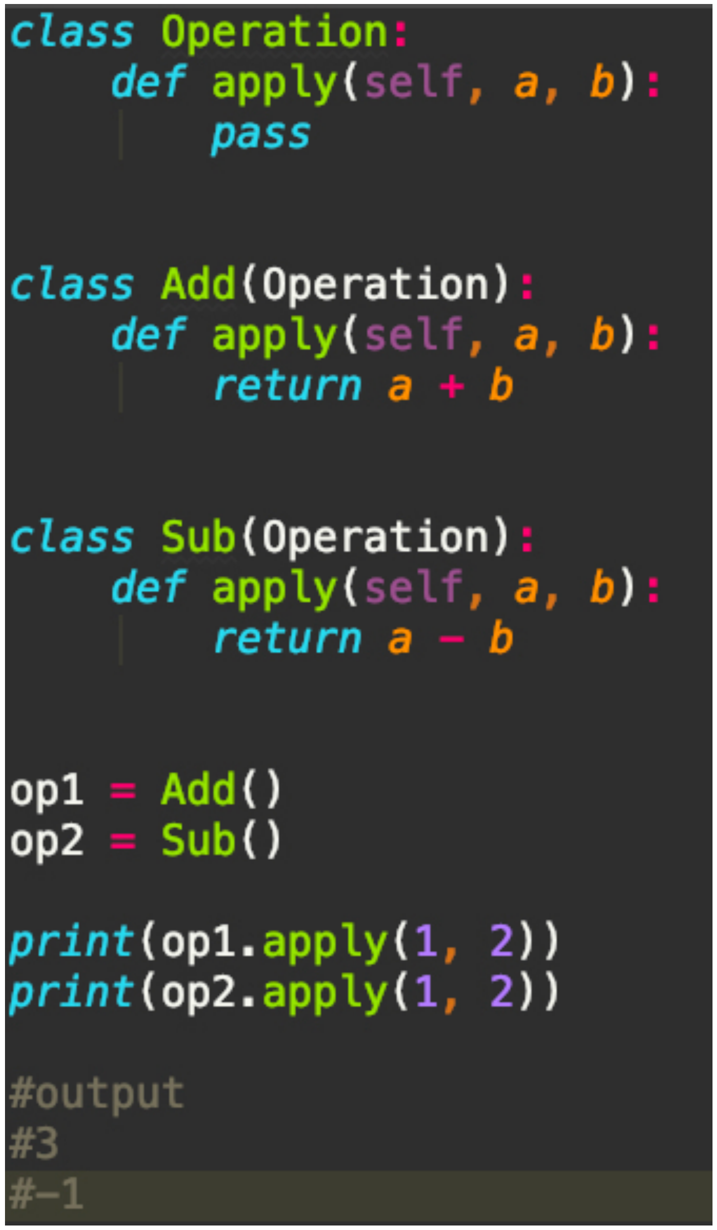
Multiple Inheritance



```
class FlyingAnimal:
    def __init__(self):
        pass

    def fly(self):
        return "I can fly!"


class SwimmingAnimal:
    def __init__(self):
        pass

    def swim(self):
        return "I can swim!"


class Duck(FlyingAnimal, SwimmingAnimal):
    def __init__(self):
        pass

d = Duck()
print(d.fly())
print(d.swim())
```

Polymorphism

Polymorphism means same function name (but different signatures) being uses for different types.



```
class Operation:
    def apply(self, a, b):
        pass


class Add(Operation):
    def apply(self, a, b):
        return a + b


class Sub(Operation):
    def apply(self, a, b):
        return a - b


op1 = Add()
op2 = Sub()

print(op1.apply(1, 2))
print(op2.apply(1, 2))

#output
#3
#-1
```