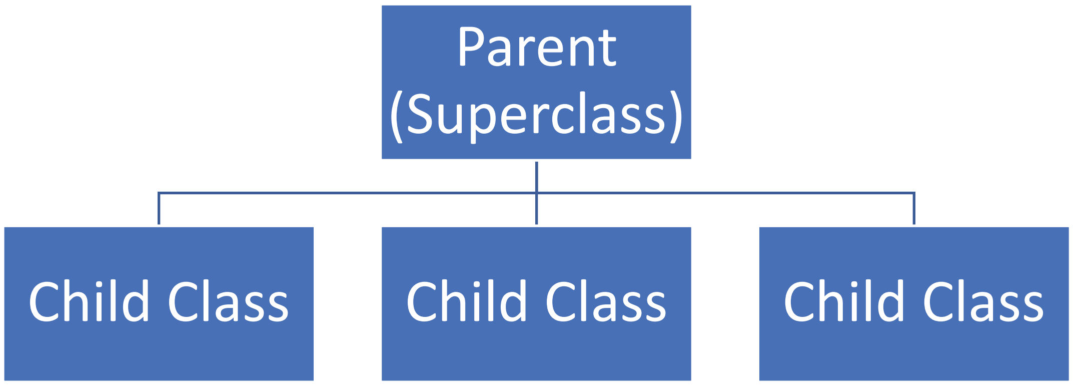
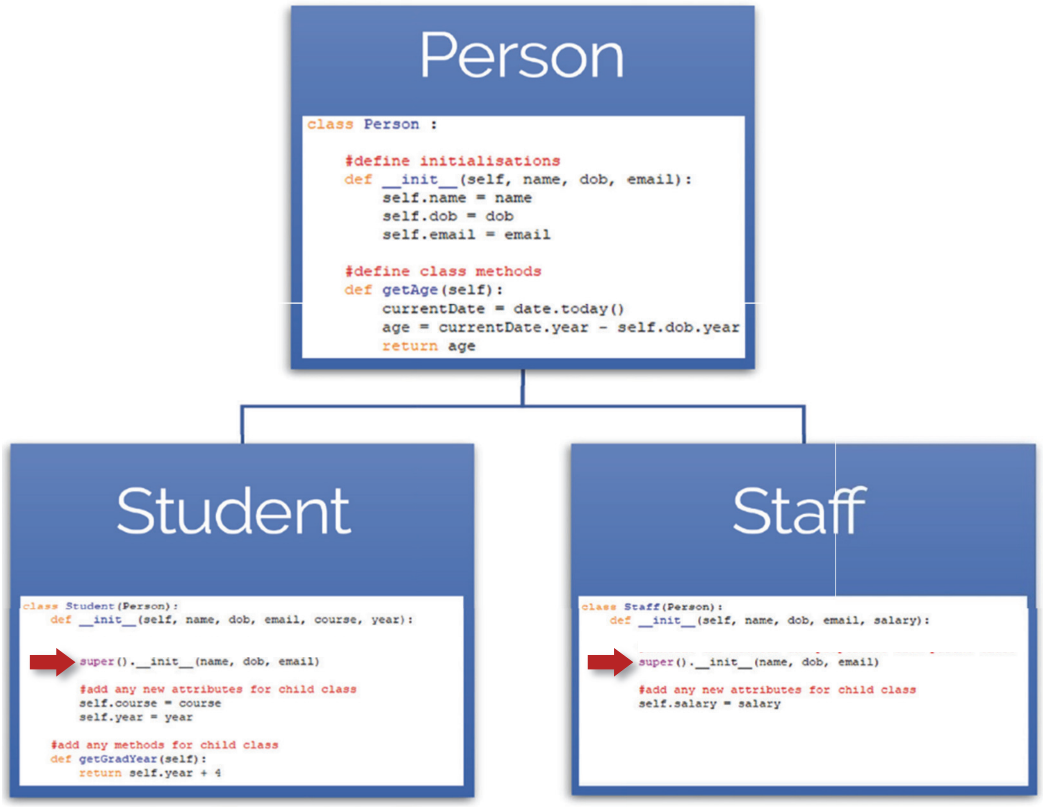
Inheritance

A class can inherit all the methods and attributes from another class.



If a class called 'person' had name, age, and dob. We could define two other child classes called 'student' and 'staff'. Both can inherit the methods and attributes from the 'person' class.



```python
class Person :

    #define initialisations
    def __init__(self, name, dob, email):
        self.name = name
        self.dob = dob
        self.email = email

    #define class methods
    def getAge(self):
        currentDate = date.today()
        age = currentDate.year - self.dob.year
        return age
```

```python
class Student(Person):
    def __init__(self, name, dob, email, course, year):


        super().__init__(name, dob, email)

        #add any new attributes for child class
        self.course = course
        self.year = year

    #add any methods for child class
    def getGradYear(self):
        return self.year + 4
```

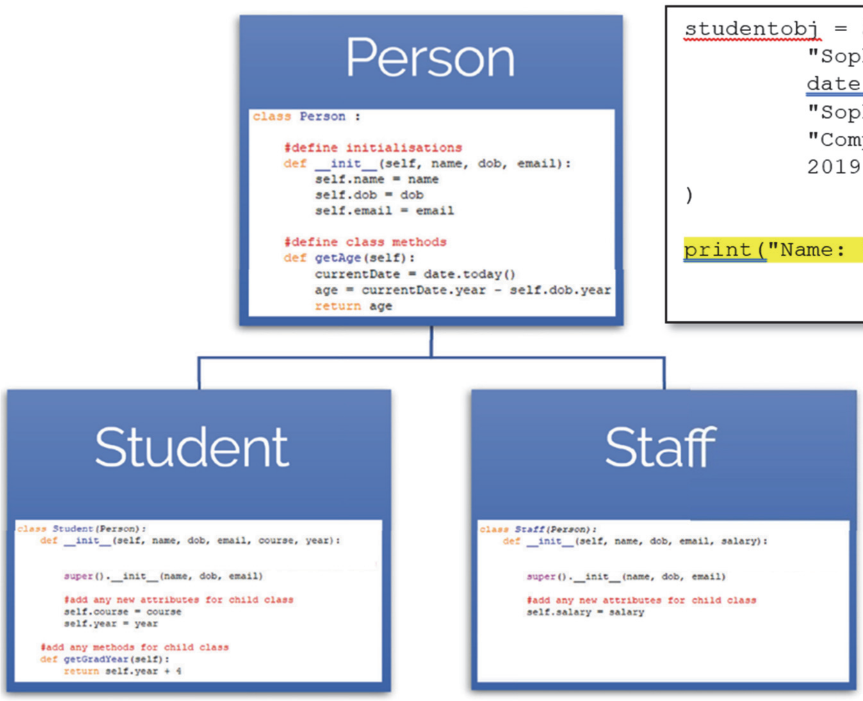
```python
class Staff(Person):
    def __init__(self, name, dob, email, salary):


        super().__init__(name, dob, email)

        #add any new attributes for child class
        self.salary = salary
```

When defining your child classes, you might need to access methods defined in the parent. To do this, use the `super()` function. The `super()` function allows us to refer to the parent class explicitly.

For example, if you need to access the attributes in the parent, you'll need to call the parent's `__init__()` constructor using `super()` in the child class.

```python
super().__init__(name, dob, email)
```

In the example below, in the print statement, we are calling the “name” attribute which is defined in the parent class Person, from the object (studentobj) created using the Student class. To access this we need to use `super()` to call the `__init__()` constructor method defined in the parent class (Person).



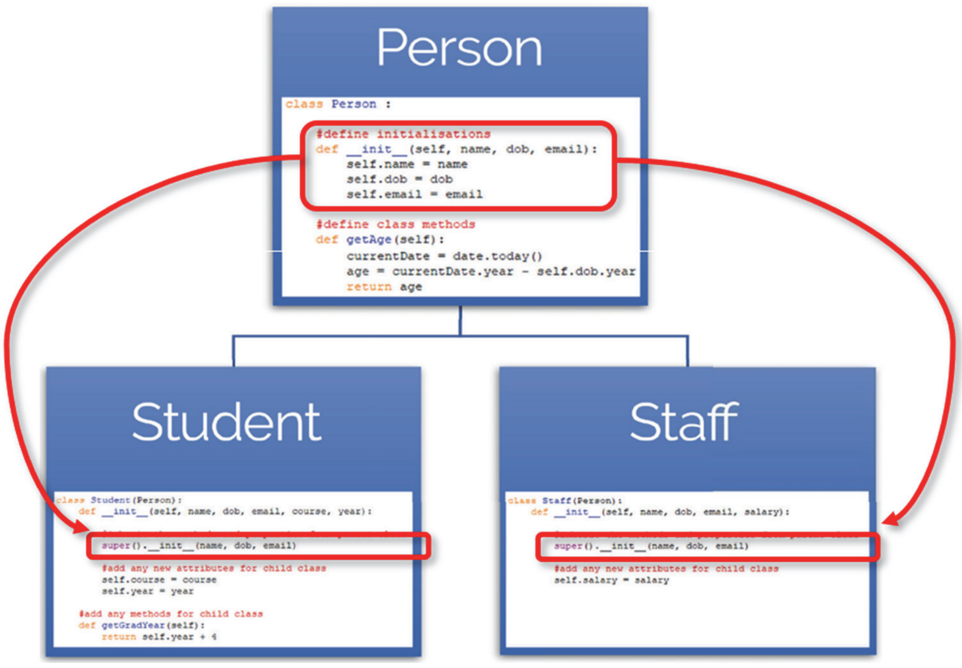
```
class Person :

    #define initialisations
    def __init__(self, name, dob, email):
        self.name = name
        self.dob = dob
        self.email = email

    #define class methods
    def getAge(self):
        currentDate = date.today()
        age = currentDate.year - self.dob.year
        return age
```

```
class Student(Person):
    def __init__(self, name, dob, email, course, year):


        super().__init__(name, dob, email)

        #add any new attributes for child class
        self.course = course
        self.year = year

    #add any methods for child class
    def getGradYear(self):
        return self.year + 4
```

```
class Staff(Person):
    def __init__(self, name, dob, email, salary):


        super().__init__(name, dob, email)

        #add any new attributes for child class
        self.salary = salary
```

```
studentobj = Student (
          "Sophie",
          date(1999, 4, 2),
          "Sophie@mymail.com",
          "Computing",
          2019
)

print("Name:  ", studentobj.name)
```

Here in the two child classes we’ve called the `super()` function in the `__init__()` method of the child class.



The `super()` function calls the the `__init__()` method of parent class, which gives the child class access to all the attributes of the parent class. If we don’t need to access the methods defined in the parent class from an object created from the child class, then we don’t need the `super()` function.