Multiple Inheritance

In Python, a class can inherit **attributes** and **methods** from **more than one class**. Critics of multiple inheritance say that it is confusing and complex, like the diamond problem, which is out of the scope of this lesson.

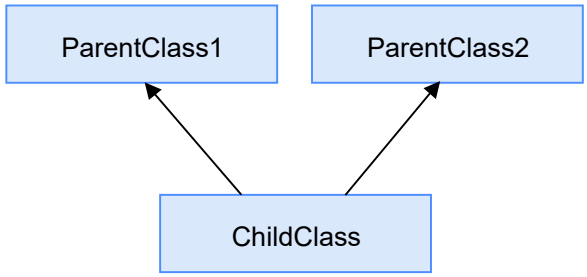
The idea that multiple inheritance is somehow “bad” is based on the fact that some languages either do not support multiple inheritance (like Java) or do not have a very good implementation for it. Python, however, has a very sophisticated and well-structured approach to multiple inheritance.

Consider the following code snippet where `ChildClass` inherits from `ParentClass1`, `ParentClass2`, `ParentClass3`, and so on.

```
class ChildClass (ParentClass1, ParentClass2, ParentClass3, ...):
  pass
```

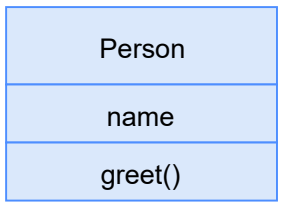
This behavior is illustrated in the following figure:

1 of 2

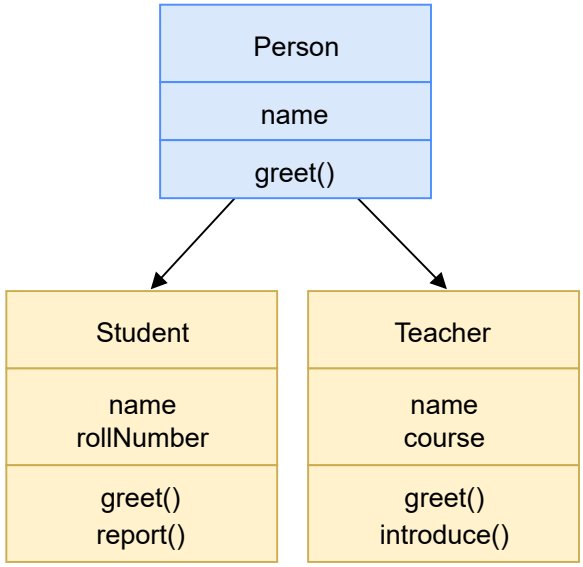


ChildClass inherits from both ParentClass1 and ParentClass2

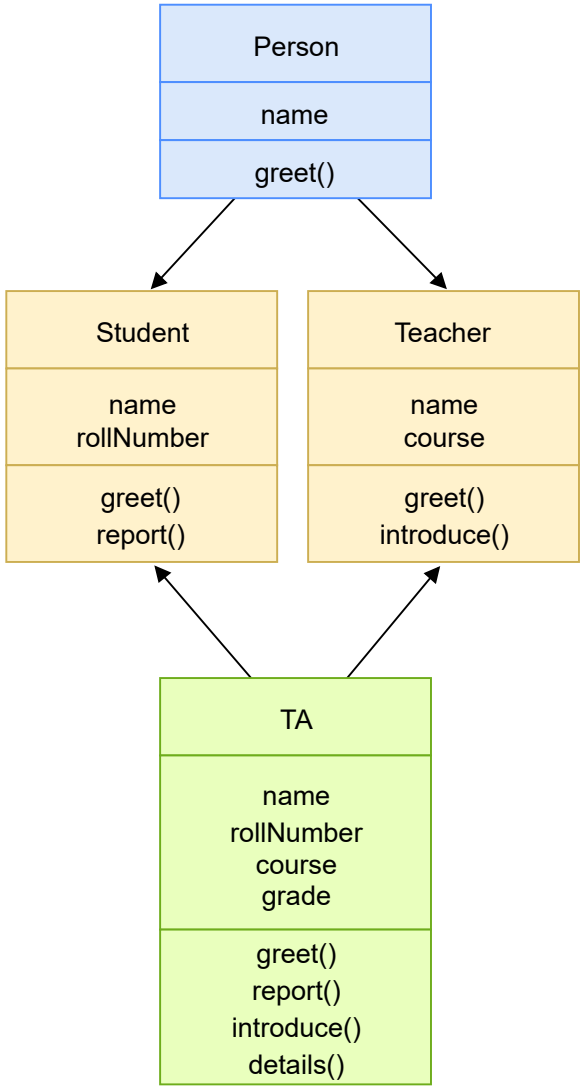
Now, let's discuss a real-world example where a `Person` can either be a `Student` or a `Teacher`. Additionally, a `Person` can be a `TA`. A `TA` inherits the attributes and methods of both the `Student` and `Teacher` class because, essentially, a `TA` is a `Student` who is performing some of the duties that a `Teacher` performs. This hierarchy is shown in the following figure:



TA class inheriting from both Student and Teacher



TA class inheriting from both Student and Teacher

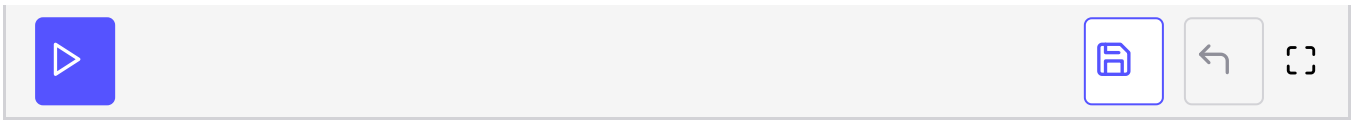


TA class inheriting from both Student and Teacher

Consider the following code:

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

  def greet(self):
    print("Hi, I am " + self.name + ".")


class Student (Person): # Student inherits from Person class
  def __init__(self, name, rollNumber):
    self.name = name # Attribute inherited from the Person class
    self.rollNumber = rollNumber # Student's attribute
    Person.__init__(self, name) # Person's constructor
```

```
  def report(self): # Student's method
    print("My roll number is " + self.rollNumber + ".")

class Teacher (Person): # Teacher inherits from Person class
  def __init__(self, name, course):
    self.name = name # Attribute inherited from the Person class
    self.course = course # Teacher's attribute
    Person.__init__(self, name) # Person's constructor

  def introduce(self): # Teacher's method
    print("I teach " + self.course + ".")

class TA (Student, Teacher): # TA inherits from Student and Teacher class
  def __init__(self, name, rollNumber, course, grade):
    self.name = name # Attribute inherited from the Person class
    self.rollNumber = rollNumber # Attribute inherited from the Student class
    self.course = course # Attribute inherited from the Teacher class
    self.grade = grade # TA's attribute

  def details(self): # TA's method
    if self.grade=="A*" or self.grade=="A" or self.grade=="A-": # if person is elligible for
      Person.greet(self) # can access Person's greet method
      Student.report(self) # can access Student's report method
      Teacher.introduce(self) # can access Teacher's introduce method
      print ("I got an " + self.grade + " in " + self.course + ".")
    else: # person is not elligible for TAship
      print(self.name + ", you can not apply for TAship.")

ta = TA('Ali', '13K-1234', 'Data Structures' ,'A') # TA object
ta.details()

#uncomment any of the following lines of code and see how they work
# ta.greet()
# ta.report()
# ta.introduce()

print("\n")

ta2 = TA('Ahmed', '14K-5678', 'Algorithms' ,'B')
ta2.details()
```



Now that you get the basic idea, let's discuss a better way to access methods and attributes of the superclass in the next lesson.