DataScientest • com

Object Oriented Programming

Inheritance

Inheritance

> *Inheritance* is used to create a *subclass* from an existing class. We say that this new class *inherits* from the first one because it will automatically have the same attributes and methods.
>
> Furthermore, it is possible to add attributes or methods that will be specific to this subclass.
>
> In the first part of this module, we introduced the `Vehicle` class defined as follows:
>
> ```
> class Vehicle:
>     def __init__(self, a, b = []):
>         self.seats = a
>         self.passengers = b
>     def print_passengers(self):
>         for i in range (len (self.passengers)):
>             print (self.passengers [i])
>     def add(self, name):
>             self.passengers.append (name)
> ```
>
> We can define a `Motorcycle` class which inherits from the `Vehicle` class as follows:
>
> ```
>     class Motorcycle(Vehicle):
>         def__init__(self, b, c):
>             self.seats = 2
>             self.passengers = b
>             self.brand = c
>
> Motorcycle = Motorcycle(['Pierre', 'Dimitri'], 'Yamaha')
> ```
>
> By rewriting the `__init__` method, any `Motorcycle` object will automatically have 2 seats and a new `brand` attribute.
>
> Thanks to inheritance, we can call the `print_passengers` method defined in the `Vehicle` class from an instance of the `Motorcycle` class.

- **(a)** Run the following cell to convince yourself.

In [4]:
```
class Vehicle: # Definition of the Vehicle class
    def __init__(self, a, b = []):
        self.seats = a       # number of seats in the vehicle
        self.passengers = b  # list containing the names of the passengers

    def print_passengers(self): # Prints the names of the passengers in the vehicle
        for i in range(len(self.passengers)):
            print(self.passengers[i])

    def add(self,name): # Adds a new passenger to the list of passengers.
            self.passengers.append(name)

class Motorcycle(Vehicle):
    def __init__(self, b, c):
        self.seats = 2     # The number of seats is automatically set to 2 and is not modified by the arguments
        self.passengers = b
        self.brand = c

moto1 = Motorcycle(['Pierre','Dimitri'], 'Yamaha')
moto1.add('Yohann')
moto1.print_passengers()
```

```
Pierre
Dimitri
Yohann
```

In [5]:
```
moto1.print_passengers()
```

```
Pierre
Dimitri
Yohann
```

In [6]:
```
moto1.add("Abdullah")
```

In [7]:
```
moto1.print_passengers()
```

```
Pierre
Dimitri
Yohann
Abdullah
```

- **(b)** Define in the `Motorcycle` class an `add` method which will add a name passed as an argument to the list of passengers while checking that there are still seats available. If there are no seats left on the `Motorcycle` , it should display *The vehicle is full.* If there are any remaining, the method should add the name to the list and

display the number of seats remaining

In [ ]:
```
class Motorcycle(Vehicle):
    def __init__(self, b, c):
        self.seats = 2
        self.passengers = b
        self.brand = c

    def add(self, name):
        raise NotImplementedError  #### Insert your code here
```

In [8]:
```
class Motorcycle(Vehicle):
    def __init__(self,  b, c, seats = 2):
        self.seats = seats
        self.passengers = b
        self.brand = c

    def add(self, name):
        if len(self.passengers) < self.seats :
            self.passengers.append(name)
            print("Number of seats remaining :", len(self.passengers)-self.seats)

        else:
            print("The vechicle is full.")

moto1 = Motorcycle(['Pierre','Dimitri'], 'Yamaha')
moto2 = Motorcycle(['Pierre'], 'Yamaha')

# add passenger to moto1
moto1.add("Abdullah")
print(moto1.seats)
# add passenger to moto2
moto2.add("Abdullah")
# try to add again passenger to moto2
moto2.add("Nurullah")
```

```
The vechicle is full.
2
Number of seats remaining : 0
The vechicle is full.
```

In [9]:
```
class Motorcycle(Vehicle):
    def __init__(self, a, b, c):
        self.seats = a
        self.passengers = b
        self.brand = c

    def add(self, name):
        if len(self.passengers) >= 2:
            print("The vechicle is full.")
        else:
            self.passengers.append(name)
            print("Number of seats remaining :", len(self.passengers)-self.seats)
```

In [10]:
```
moto1.__class__.__name__
```

Out[10]: 'Motorcycle'

Hide solution

In [14]:
```
class Motorcycle(Vehicle):
    def __init__(self, b, c):
        self.seats = 2
        self.passengers = b
        self.brand = c

    def add(self, name):
        if(len(self.passengers) <  self.seats):
            self.passengers.append(name)
            print('There are', self.seats - len(self.passengers), 'seats left.')
        else:
            print("The vehicle is full.")
```

We run the following instructions:

```
car2 = Vehicle(3, ['Antoine', 'Thomas', 'Raphaël'])
moto2 = Motorcycle(['Guillaume', 'Charles'], 'Honda')
car2.add('Benjamin')
moto2.add('Dimitri')
```

In addition, we recall that the classes `Vehicle` and `Motorcycle` are defined as follows:

```
class Vehicle:
        def __init__(self, a, b = []):
            self.seats = a
            self.passengers = b

        def print_passengers(self):
            for i in range(len(self.passengers)):
                print(self.passengers [i])

        def add(self, name):
            self.passengers.append(name)
```

```
class Motorcycle(Vehicle):
    def __init__(self, b, c):
        self.seats = 2
        self.passengers = b
        self.brand = c

    def add(self, name):
        if(len(self.passengers) <  self.seats):
            self.passengers.append(name)
            print('There are', self.seats - len(self.passengers), 'seats left.')
        else:
            print("The vehicle is full.")
```

VBox(children=(ToggleButtons(button_style='success', options=('Answer A', 'Answer B', 'Answer C'), tooltips=('…

- What is the output of the `print(car2.seats)` instruction?
  - A: `Antoine Thomas Raphael Benjamin`
  - B: 4
  - C: `The vehicle is full.`
  - D: 3

VBox(children=(ToggleButtons(button_style='success', options=('Answer A', 'Answer B', 'Answer C', 'Answer D'),…

- Why is the instruction `car3 = Vehicle(4)` well written but the instruction `moto3 = Motorcycle(6)` returns an error?
  - A: A `Motorcycle` object cannot have 6 seats.
  - B: The constructor of the `Vehicle` class takes only one argument.
  - C: An argument is missing when initializing the `moto3` instance.

VBox(children=(ToggleButtons(button_style='success', options=('Answer A', 'Answer B', 'Answer C'), tooltips=('…

- **(c)** Create a `Convoy` class which will have 2 attributes: The first attribute, named `vehicle_list` is a list of `Vehicle` objects and the second attribute `length` is the total number of vehicles in the `Convoy` . A convoy will be automatically initialized with a `Vehicle` that has 4 seats and no passengers.
- **(d)** Define in `Convoy` class an `add_vehicle` method which will add an object of type `Vehicle` at the end of the list of vehicles of the convoy. Do not forget to update the length of the convoy.

In [15]:

```
### Insert your code here

class Convoy():
    def __init__(self, length = 0, vehicle_list = [] ):
        self.length = length
        self.vehicle_list = vehicle_list


    def add_vehicle(self, new_vehicle):
        self.vehicle_list.append(new_vehicle)
        self.length += 1

car1 = Vehicle(4, [])
car2 = Vehicle(3, ["Abdullah", "Hatice"])

convoy1 = Convoy()
convoy1.add_vehicle("car2")
print(convoy1.vehicle_list)
```

```
['car2']
```

Hide solution

In [4]:

```
class Convoy:
    def __init__(self):
        self.vehicle_list = []
        self.vehicle_list.append(Vehicle(4)) # vehicule_list is initialized with a list containing 1 vehicle
        self.length = 1                      # the length attribute is initialized to 1

    def add_vehicle(self, vehicle):
        self.vehicle_list.append(vehicle)    # a Vehicle is added at the end of the list
        self.length = self.length + 1        # update the length of the convoy
```

- **(e)** Initialize a `convoy1` object of the `Convoy` class.
- **(f)** Add the passenger `"Albert"` to the first vehicle of `convoy1` .
- **(g)** Add a motorcycle from the brand `"Honda"` to `convoy1` which will be driven by `"Raphael"` .

In [17]:

```
### Insert your code here

convoy1 = Convoy()

convoy1.vehicle_list[0].add("Albert")

moto1 = Motorcycle(["Raphael"], "Honda")

convoy1.add_vehicle(moto1)
```

In [18]:
```
convoy1.vehicle_list
```

Out[18]:
```
[<__main__.Vehicle at 0x7f30f433f6a0>, <__main__.Motorcycle at 0x7f30f433f8e0>]
```

Hide solution

In [5]:
```
convoy1 = Convoy()                                      # Instanciation of the convoy

convoy1.vehicle_list[0].add('Albert')                   # "Albert" is added to the first vehicle in the convoy

convoy1.add_vehicle(Motorcycle(['Raphael'] , 'Honda')) # We have to remember that the first argument of the Motorcycle
                                                        # constructor is a list and not a string.
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-5-070e9d4959c6> in <module>
----> 1 convoy1 = Convoy()                                      # Instanciation of the convoy
      2
      3 convoy1.vehicle_list[0].add('Albert')                   # "Albert" is added to the first vehicle in the convoy
      4
      5 convoy1.add_vehicle(Motorcycle(['Raphael'] , 'Honda')) # We have to remember that the first argument of the Motorcycle

<ipython-input-4-e8d75ab76f00> in __init__(self)
      2     def __init__(self):
      3         self.vehicle_list = []
----> 4         self.vehicle_list.append(Vehicle(4)) # vehicule_list is initialized with a list containing 1 vehicle
      5         self.length = 1                      # the length attribute is initialized to 1
      6

NameError: name 'Vehicle' is not defined
```

In [20]:
```
convoy1.vehicle_list
```

Out[20]:
```
[<__main__.Vehicle at 0x7f30f433f610>, <__main__.Motorcycle at 0x7f30f433f7f0>]
```

- **(h)** Write a small script that will display all the passengers in `convoy1` .

In [21]:
```
### Insert your code here

class Convoy():
    def __init__(self):
        self.vehicle_list = []
        car1 = Vehicle(0, [])
        self.vehicle_list.append(car1)
        self.length = 1


    def add_vehicle(self, new_vehicle):
        self.vehicle_list.append(new_vehicle)
        self.length += 1

    def display(self):
        num = 0
        for i in self.vehicle_list:
            print(i.passengers)
            for j in i.passengers:
                num +=1
        print("Number of vehicle :", len(self.vehicle_list))
        print("Total number of passengers : ", num)



convoy1 = Convoy()

convoy1.vehicle_list[0].add('Albert')

convoy1.add_vehicle(Motorcycle(['Raphael', 'Ali'] , 'Honda'))
```

In [22]:
```
convoy1.display()
```

```
['Albert']
['Raphael', 'Ali']
Number of vehicle : 2
Total number of passengers :  3
```

Hide solution

In [168]:
```
for vehicle in convoy1.vehicle_list: # We go through the list of vehicles in the convoy
    vehicle.print_passengers()       # We use the print_passengers method of the Vehicle class
```

```
Albert
Raphael
```

In [ ]:
```

```

Unvalidate