The class, Pokemon, is provided below and describes a Pokemon and its leveling and evolving characteristics. An instance of the class is one pokemon that you create.

Grass\_Pokemon is a subclass that inherits from Pokemon but changes some aspects, for instance, the boost values are different.

For the subclass Grass\_Pokemon, add another method called action that returns the string "[name of pokemon] knows a lot of different moves!". Create an instance of this class with the name as "Belle". Assign this instance to the variable p1.

class Pokemon(object):

attack = 12

defense = 10

health = 15

p\_type = "Normal"

def \_\_init\_\_(self, name, level = 5):

self.name = name

self.level = level

def train(self):

self.update()

self.attack\_up()

self.defense\_up()

self.health\_up()

self.level = self.level + 1

if self.level%self.evolve == 0:

return self.level, "Evolved!"

else:

return self.level

def attack\_up(self):

self.attack = self.attack + self.attack\_boost

return self.attack

def defense\_up(self):

self.defense = self.defense + self.defense\_boost

return self.defense

def health\_up(self):

self.health = self.health + self.health\_boost

return self.health

def update(self):

self.health\_boost = 5

self.attack\_boost = 3

self.defense\_boost = 2

self.evolve = 10

def \_\_str\_\_(self):

self.update()

return "Pokemon name: {}, Type: {}, Level: {}".format(self.name, self.p\_type, self.level)

class Grass\_Pokemon(Pokemon):

attack = 15

defense = 14

health = 12

def \_\_init\_\_(self , name):

self.name = name

def update(self):

self.health\_boost = 6

self.attack\_boost = 2

self.defense\_boost = 3

self.evolve = 12

def moves(self):

self.p\_moves = ["razor leaf", "synthesis", "petal dance"]

def action(self):

return "{} knows a lot of different moves!".format(self.name)

p1 =Grass\_Pokemon("Belle")

Modify the Grass\_Pokemon subclass so that the attack strength for Grass\_Pokemon instances does not change until they reach level 10. At level 10 and up, their attack strength should increase by the attack\_boost amount when they are trained.

To test, create an instance of the class with the name as "Bulby". Assign the instance to the variable p2. Create another instance of the Grass\_Pokemon class with the name set to "Pika" and assign that instance to the variable p3. Then, use Grass\_Pokemon methods to train the p3 Grass\_Pokemon instance until it reaches at least level 10.

class Pokemon(object):

attack = 12

defense = 10

health = 15

p\_type = "Normal"

def \_\_init\_\_(self, name, level = 5):

self.name = name

self.level = level

def train(self):

self.update()

self.attack\_up()

self.defense\_up()

self.health\_up()

self.level = self.level + 1

if self.level%self.evolve == 0:

return self.level, "Evolved!"

else:

return self.level

def attack\_up(self):

self.attack = self.attack + self.attack\_boost

return self.attack

def defense\_up(self):

self.defense = self.defense + self.defense\_boost

return self.defense

def health\_up(self):

self.health = self.health + self.health\_boost

return self.health

def update(self):

self.health\_boost = 5

self.attack\_boost = 3

self.defense\_boost = 2

self.evolve = 10

def \_\_str\_\_(self):

return "Pokemon name: {}, Type: {}, Level: {}".format(self.name, self.p\_type, self.level)

class Grass\_Pokemon(Pokemon):

attack = 15

defense = 14

health = 12

p\_type = "Grass"

def update(self):

self.health\_boost = 6

self.attack\_boost = 2

self.defense\_boost = 3

self.evolve = 12

def train(self):

self.update()

if self.level >= 10:

self.attack\_up()

self.defense\_up()

self.health\_up()

self.level = self.level + 1

if self.level%self.evolve == 0:

return self.level, "Evolved!"

else:

return self.level

def moves(self):

self.p\_moves = ["razor leaf", "synthesis", "petal dance"]

p2 = Grass\_Pokemon("Bulby", level = 5)

p3 = Grass\_Pokemon("Pika", level = 5)

p3.train()

Along with the Pokemon parent class, we have also provided several subclasses. Write another method in the parent class that will be inherited by the subclasses. Call it opponent. It should return which type of pokemon the current type is weak and strong against, as a tuple.

* **Grass** is weak against *Fire* and strong against *Water*
* **Ghost** is weak against *Dark* and strong against *Psychic*
* **Fire** is weak against *Water* and strong against *Grass*
* **Flying** is weak against *Electric* and strong against *Fighting*

For example, if the p\_type of the subclass is 'Grass', .opponent() should return the tuple ('Fire', 'Water')

class Pokemon():

attack = 12

defense = 10

health = 15

p\_type = "Normal"

def \_\_init\_\_(self, name,level = 5):

self.name = name

self.level = level

self.weak = "Normal"

self.strong = "Normal"

def train(self):

self.update()

self.attack\_up()

self.defense\_up()

self.health\_up()

self.level = self.level + 1

if self.level%self.evolve == 0:

return self.level, "Evolved!"

else:

return self.level

def attack\_up(self):

self.attack = self.attack + self.attack\_boost

return self.attack

def defense\_up(self):

self.defense = self.defense + self.defense\_boost

return self.defense

def health\_up(self):

self.health = self.health + self.health\_boost

return self.health

def update(self):

self.health\_boost = 5

self.attack\_boost = 3

self.defense\_boost = 2

self.evolve = 10

def \_\_str\_\_(self):

self.update()

return "Pokemon name: {}, Type: {}, Level: {}".format(self.name, self.p\_type, self.level)

def opponent(self):

return self.weak, self.strong

class Grass\_Pokemon(Pokemon):

attack = 15

defense = 14

health = 12

p\_type = "Grass"

def \_\_init\_\_(self, name,level = 5):

self.name = name

self.level = level

self.weak = "Fire"

self.strong = "Water"

def update(self):

self.health\_boost = 6

self.attack\_boost = 2

self.defense\_boost = 3

self.evolve = 12

class Ghost\_Pokemon(Pokemon):

p\_type = "Ghost"

def \_\_init\_\_(self, name,level = 5):

self.name = name

self.level = level

self.weak = "Dark"

self.strong = "Psychic"

def update(self):

self.health\_boost = 3

self.attack\_boost = 4

self.defense\_boost = 3

class Fire\_Pokemon(Pokemon):

p\_type = "Fire"

def \_\_init\_\_(self, name,level = 5):

self.name = name

self.level = level

self.weak = "Water"

self.strong = "Grass"

class Flying\_Pokemon(Pokemon):

p\_type = "Flying"

def \_\_init\_\_(self, name,level = 5):

self.name = name

self.level = level

self.weak = "Electric"

self.strong = "Fighting"