**Name:** Bonumaddi Naga Pravallika

**No:** 120CS0121

**Mini Project**: Pokemon game

**Description**:

we'll build a tiny implementation of a Pokemon-like game that allows us to create little creatures, feed them to gain health, and let them battle against each other

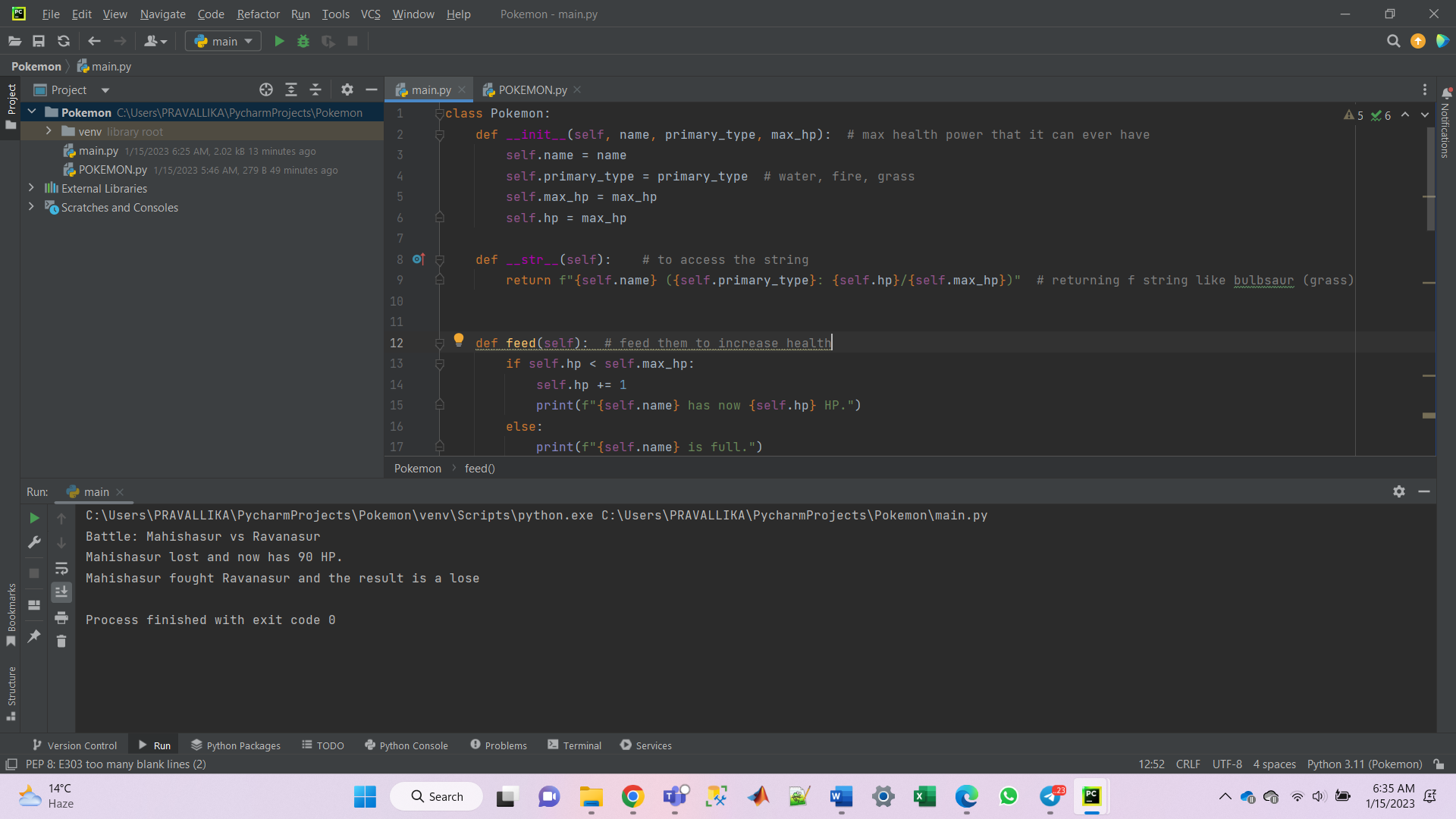
**Source code:**

class Pokemon:  
 def \_\_init\_\_(self, name, primary\_type, max\_hp): # max health power that it can ever have  
 self.name = name  
 self.primary\_type = primary\_type # water, fire, grass  
 self.max\_hp = max\_hp  
 self.hp = max\_hp  
  
 def \_\_str\_\_(self): # to access the string  
 return f"{self.name} ({self.primary\_type}: {self.hp}/{self.max\_hp})" # returning f string like bulbsaur (grass)  
  
  
 def feed(self): # feed them to increase health  
 if self.hp < self.max\_hp:  
 self.hp += 1  
 print(f"{self.name} has now {self.hp} HP.")  
 else:  
 print(f"{self.name} is full.")  
  
 def battle(self, other):  
 print("Battle:", self.name, "vs", other.name)  
 result = self.typewheel(self.primary\_type, other.primary\_type)  
 if result == "lose":  
 self.hp -= 10  
 print(f"{self.name} lost and now has {self.hp} HP.")  
 print(f"{self.name} fought {other.name} and the result is a {result}")  
 @staticmethod # if not it will show an error like, it needs self and all  
 def typewheel(type1, type2): # like what type of things they are  
 # making a dictionary  
 result\_map = {0: "lose", 1: "win", -1: "tie"}  
 # The mapping between types and result conditions  
 game\_map = {"water": 0, "fire": 1, "grass": 2}  
 # Win-lose matrix  
 rps\_table = [  
 [-1, 1, 0], # water  
 [0, -1, 1], # fire  
 [1, 0, -1] # grass  
 ]

# water # 1. water vs water => tie = -1 # 2. water vs fire => win = 1  
 result = rps\_table[game\_map[type1]][game\_map[type2]] # using a list lookup to declare a winner  
 return result\_map[result]

if \_\_name\_\_ == '\_\_main\_\_':  
 mahi = Pokemon(name="Mahishasur", primary\_type="grass", max\_hp=100)  
 kamsa = Pokemon(name="Kamsasur", primary\_type="fire", max\_hp=150)  
 ravan = Pokemon(name="Ravanasur", primary\_type="fire", max\_hp=150)  
 mahi.battle(ravan)

**OUTPUT:**



**concepts covered:**

Python Object Oriented Programming

Game logic - Instance methods - @staticmethod - Instance attributes, self - Dunder methods: *\_init\_\_, \_\_str\_* - if *\_name\_* == '\_\_main\_\_'

**Motto**: By the end of day three we'll understand the basics of object-oriented programming in Python and how we can apply them to build a small game that you can play in our Python REPL(Read Evaluate print loop).