

```

1  #Classes Challenge 38: Pykemon Simulation App
2  import random
3
4  #Parent class
5  class Pykemon():
6      """A model of a generic Pykemon character."""
7
8      def __init__(self, name, element, health, speed):
9          """Initialize attributes."""
10         self.name = name.title()
11         self.element = element
12         #Current health is current, max health will be referenced for healing
13         self.current_health = health
14         self.max_health = health
15         self.speed = speed
16         self.is_alive = True
17
18
19     def light_attack(self, enemy):
20         """A light attack guaranteed to do minimal damage."""
21         #Generate damage
22         damage = random.randint(15, 25)
23
24         #All pykemon will have a list moves = [light, heavy, restore, special]
25         #All light attacks will appear at index 0 in the list moves
26         #This attribute will be initialized in the child class
27         print("Pykemon " + self.name + " used " + self.moves[0] + ".")
28         print("It dealt " + str(damage) + " damage.")
29
30         #Deal damage to the enemy
31         enemy.current_health -= damage
32
33
34     def heavy_attack(self, enemy):
35         """A heavy attack that could deal MASSIVE damage, or no damage at all."""
36         #Generate damage
37         damage = random.randint(0, 50)
38
39         #All pykemon will have a list moves = [light, heavy, restore, special]
40         #All heavy attacks will appear at index 1 in the list moves
41         #This attribute will be initialized in the child class
42         print("Pykemon " + self.name + " used " + self.moves[1] + ".")
43
44         #Dealt no damage
45         if damage < 10:
46             print("The attack missed!!!")
47         else:
48             print("It dealt " + str(damage) + " damage.")
49             #Deal the damage to the enemy
50             enemy.current_health -= damage
51
52
53     def restore(self):
54         """A healing move that will restore our current health."""
55         #Generate restore value
56         heal = random.randint(15, 25)
57
58         #All pykemon will have a list moves = [light, heavy, restore, special]
59         #All restore moves will appear at index 2 in the list moves
60         #This attribute will be initialized in the child class
61         print("Pykemon " + self.name + " used " + self.moves[2] + ".")
62         print("It healed " + str(heal) + " health points.")
63
64         #Heal the pykemon

```

```

65         self.current_health += heal
66
67         #Check to see if we have exceeded the max health of the pykemon
68         if self.current_health > self.max_health:
69             self.current_health = self.max_health
70
71
72     def faint(self):
73         """If you run out of health, you faint..."""
74         if self.current_health <= 0:
75             self.is_alive = False
76             print("Pykemon " + self.name + " has fainted!")
77             input("Press Enter to continue.")
78
79
80     def show_stats(self):
81         """Display the current pykemon stats."""
82         print("\nName: " + self.name)
83         print("Element Type: " + self.element)
84         print("Health: " + str(self.current_health) + " / " +
85 str(self.max_health))
86         print("Speed: " + str(self.speed))
87
88 #Child Classes
89 class Fire(Pykemon):
90     """A Fire based pykemon that is a child of the Pykemon parent class."""
91
92     def __init__(self, name, element, health, speed):
93         """Initialize attributes from the parent Pykemon class."""
94         super().__init__(name, element, health, speed)
95         #Move list unique to all Fire type Pykemon
96         self.moves = ['Scratch', 'Ember', 'Light', 'Fire Blast']
97
98     def special_attack(self, enemy):
99         """FIRE BLAST: an elemental fire move. Massive damage to grass type,
100 normal damage to fire type, minimal damage to water type."""
101         print("Pykemon " + self.name + " used " + self.moves[3].upper() + "!")
102
103         #Generate damage based on enemy type
104         if enemy.element == 'GRASS':
105             print("It's SUPER effective!")
106             damage = random.randint(35, 50)
107         elif enemy.element == 'WATER':
108             print("It's not very effective...")
109             damage = random.randint(5, 10)
110         else:
111             damage = random.randint(10, 30)
112
113         #Deal damage
114         print("It dealt " + str(damage) + " damage.")
115         enemy.current_health -= damage
116
117
118     def move_info(self):
119         """Display pykemon move info"""
120         print("\n" + self.name + " Moves: ")
121
122         #Light attack
123         print("-- " + self.moves[0] + " --")
124         print("\tAn efficient attack...")
125         print("\tGuaranteed to do damage within a range of 15 to 25 damage
126 points.")
127         #Heavy attack

```

```

127         print("-- " + self.moves[1] + " --")
128         print("\tAn risky attack...")
129         print("\tCould deal damage up to 50 damage points or as little as 0
damage points.")
130         #Restore move
131         print("-- " + self.moves[2] + " --")
132         print("\tA restorative move...")
133         print("\tGuaranteed to heal your Pykemon 15 to 25 damage points.")
134         #Special attack
135         print("-- " + self.moves[3] + " --")
136         print("\tA powerful FIRE based attack...")
137         print("\tGuaranteed to deal MASSIVE damage to GRASS type Pykemon.")
138
139     class Water(Pykemon):
140         """A Water based pykemon that is a child of the Pykemon parent class."""
141
142         def __init__(self, name, element, health, speed):
143             """Initialize attributes from the parent Pykemon class."""
144             super().__init__(name, element, health, speed)
145             #Move list unique to all Water type Pykemon
146             self.moves = ['Bite', 'Splash', 'Dive', 'Water Cannon']
147
148         def special_attack(self, enemy):
149             """WATER CANNON: an elemental water move. Massive damage to fire type,
150             normal damage to water type, minimal damage to grass type."""
151             print("Pykemon " + self.name + " used " + self.moves[3].upper() + "!")
152
153             #Generate damage based on enemy type
154             if enemy.element == 'FIRE':
155                 print("It's SUPER effective!")
156                 damage = random.randint(35, 50)
157             elif enemy.element == 'GRASS':
158                 print("It's not very effective...")
159                 damage = random.randint(5, 10)
160             else:
161                 damage = random.randint(10, 30)
162
163             #Deal damage
164             print("It dealt " + str(damage) + " damage.")
165             enemy.current_health -= damage
166
167         def move_info(self):
168             """Display pykemon move info"""
169             print("\n" + self.name + " Moves: ")
170
171             #Light attack
172             print("-- " + self.moves[0] + " --")
173             print("\tAn efficient attack...")
174             print("\tGuaranteed to do damage within a range of 15 to 25 damage
points.")
175
176             #Heavy attack
177             print("-- " + self.moves[1] + " --")
178             print("\tAn risky attack...")
179             print("\tCould deal damage up to 50 damage points or as little as 0
damage points.")
180
181             #Restore move
182             print("-- " + self.moves[2] + " --")
183             print("\tA restorative move...")
184             print("\tGuaranteed to heal your Pykemon 15 to 25 damage points.")
185             #Special attack
186             print("-- " + self.moves[3] + " --")
187             print("\tA powerful WATER based attack...")

```

```

188         print("\tGuaranteed to deal MASSIVE damage to FIRE type Pykemon.")
189
190     class Grass(Pykemon):
191         """A Grass based pykemon that is a child of the Pykemon parent class."""
192
193         def __init__(self, name, element, health, speed):
194             """Initialize attributes from the parent Pykemon class."""
195             super().__init__(name, element, health, speed)
196             #Move list unique to all Grass type Pykemon
197             self.moves = ['Vine Whip', 'Wrap', 'Grow', 'Leaf Blade']
198
199
200         def special_attack(self, enemy):
201             """LEAF BLADE: an elemental grass move. Massive damage to water type,
202             normal damage to grass type, minimal damage to fire type."""
203             print("Pykemon " + self.name + " used " + self.moves[3].upper() + "!")
204
205             #Generate damage based on enemy type
206             if enemy.element == 'WATER':
207                 print("It's SUPER effective!")
208                 damage = random.randint(35, 50)
209             elif enemy.element == 'FIRE':
210                 print("It's not very effective...")
211                 damage = random.randint(5, 10)
212             else:
213                 damage = random.randint(10, 30)
214
215             #Deal damage
216             print("It dealt " + str(damage) + " damage.")
217             enemy.current_health -= damage
218
219
220         def move_info(self):
221             """Display pykemon move info"""
222             print("\n" + self.name + " Moves: ")
223
224             #Light attack
225             print("-- " + self.moves[0] + " --")
226             print("\tAn efficient attack...")
227             print("\tGuaranteed to do damage within a range of 15 to 25 damage
points.")
228
229             #Heavy attack
230             print("-- " + self.moves[1] + " --")
231             print("\tAn risky attack...")
232             print("\tCould deal damage up to 50 damage points or as little as 0
damage points.")
233
234             #Restore move
235             print("-- " + self.moves[2] + " --")
236             print("\tA restorative move...")
237             print("\tGuaranteed to heal your Pykemon 15 to 25 damage points.")
238
239             #Special attack
240             print("-- " + self.moves[3] + " --")
241             print("\tA powerful GRASS based attack...")
242             print("\tGuaranteed to deal MASSIVE damage to WATER type Pykemon.")
243
244         #Game class
245         class Game():
246             """A game object to control the creation and flow of pykemon and simulate
battle!"""
247
248             def __init__(self):
249                 """Initialize attributes"""
250                 #Upon creating a pykemon, element and name will be chosen randomly

```

```

249         self.pykemon_elements = ['FIRE', 'WATER', 'GRASS']
250         self.pykemon_names = ['Chewdie', 'Spatol', 'Burnmander', 'Pykachu',
'Pyonx', 'Abbacab',
251                                 'Sweetil', 'Jampot', 'Hownstooth', 'Swagilybo',
'Muttie', 'Zantbat',
252                                 'Wiggly Poof', 'Rubblesaur']
253         #Upon creating a pykemon, no battles are won
254         self.battles_won = 0
255
256
257     def create_pykemon(self):
258         """Randomly generate a Pykemon!"""
259         #Randomly generate health and speed attributes
260         health = random.randint(70, 100)
261         speed = random.randint(1, 10)
262
263         #Randomly choose an element and name
264         element = self.pykemon_elements[random.randint(0,
len(self.pykemon_elements)-1)]
265         name = self.pykemon_names[random.randint(0, len(self.pykemon_names)-1)]
266
267         #Create the right elemental pykemon
268         if element == 'FIRE':
269             pykemon = Fire(name, element, health, speed)
270         elif element == 'WATER':
271             pykemon = Water(name, element, health, speed)
272         else:
273             pykemon = Grass(name, element, health, speed)
274
275         return pykemon
276
277
278
279     def choose_pykemon(self):
280         """A method to simulate choosing a starting Pykemon similar to Pokemon"""
281         #A list to hold 3 unique starter pykemon
282         starters = []
283
284         #Pick 3 different elemental type pykemon for the starter list
285         while len(starters) < 3:
286             #Make a starter pykemon
287             pykemon = self.create_pykemon()
288
289             #Bool to determine if it is unique and should be added to the
starters list
290             valid_pykemon = True
291             for starter in starters:
292                 #Check if the name or element is already used by another starter
293                 if starter.name == pykemon.name or starter.element ==
pykemon.element:
294                     valid_pykemon = False
295                 #The created pykemon is unique, add it to the list starter
296                 if valid_pykemon:
297                     starters.append(pykemon)
298
299             #Starters list is complete, show off the starter pykemon
300             for starter in starters:
301                 starter.show_stats()
302                 starter.move_info()
303
304             #Present information to user
305             print("\nProfessor Eramo presents you with three Pykemon: ")
306             print("(1) - " + starters[0].name)
307             print("(2) - " + starters[1].name)

```

```

308     print("(3) - " + starters[2].name)
309     choice = int(input("Which Pykemon would you like to choose: "))
310     pykemon = starters[choice-1]
311
312     return pykemon
313
314     def get_attack(self, pykemon):
315         """Get a users attack choice"""
316         #Show the moves list using pykemon specific move names
317         print("\nWhat would you like to do...")
318         print("(1) - " + pykemon.moves[0])
319         print("(2) - " + pykemon.moves[1])
320         print("(3) - " + pykemon.moves[2])
321         print("(4) - " + pykemon.moves[3])
322         choice = int(input("Please enter your move choice: "))
323
324         #Formatting
325         print()
326
327     print("-----")
328
329     return choice
330
331     def player_attack(self, move, player, computer):
332         """Attack the computer AI"""
333         #Call the appropriate attack method based on the given move
334         if move == 1:
335             player.light_attack(computer)
336         elif move == 2:
337             player.heavy_attack(computer)
338         elif move == 3:
339             player.restore()
340         elif move == 4:
341             player.special_attack(computer)
342
343         #Check to see if the computer has fainted
344         computer.faint()
345
346     def computer_attack(self, player, computer):
347         """Let the computer AI attack the player"""
348         #Randomly pick a move for the computer to execute
349         move = random.randint(1, 4)
350
351         #Call the appropriate attack method based on the given move
352         if move == 1:
353             computer.light_attack(player)
354         elif move == 2:
355             computer.heavy_attack(player)
356         elif move == 3:
357             computer.restore()
358         elif move == 4:
359             computer.special_attack(player)
360
361         #Check to see if the player has fainted
362         player.faint()
363
364
365     def battle(self, player, computer):
366         """Simulate a battle round. Faster Pykemon go first."""
367         #Get the players move for the round
368         move = self.get_attack(player)
369
370         #If the player's pykemon is faster than the computer, they go first

```

```

371         if player.speed >= computer.speed:
372             #Player attacks
373             self.player_attack(move, player, computer)
374             if computer.is_alive:
375                 #Computer is still alive, let them attack
376                 self.computer_attack(player, computer)
377             #The player's pykemon is slower than the computer, the computer goes
first
378         else:
379             self.computer_attack(player, computer)
380             if player.is_alive:
381                 #Player is still alive, let them attack
382                 self.player_attack(move, player, computer)
383
384
385 #The main code
386 #Narrative introduction
387 print("Welcome to Pykemon!")
388 print("Can you become the worlds greatest Pykemon Trainer???" )
389 print("\nDon't worry! Prof Eramo is here to help you on your quest.")
390 print("He would like to gift you your first Pykemon!")
391 print("Here are three potential Pykemon partners.")
392 input("Press Enter to choose your Pykemon!")
393
394 #The main game loop
395 playing_main = True
396 while playing_main:
397     #Create a game instance
398     game = Game()
399
400     #Choose your starter pykemon
401     player = game.choose_pykemon()
402     print("\nCongratulations Trainer, you have chosen " + player.name + "!")
403     input("\nYour journey with " + player.name + " begins now...Press Enter!")
404
405     #While your pykemon is alive, continue to do battle
406     while player.is_alive:
407         #Create a computer pykemon to battle
408         computer = game.create_pykemon()
409         print("\nOH NO! A wild " + computer.name + " has approached!")
410         computer.show_stats()
411
412         #While this enemy pykemon is alive and the player pykemon is alive,
engage in battle
413         while computer.is_alive and player.is_alive:
414             game.battle(player, computer)
415
416             #Both parties survived a round, show their current stats
417             if computer.is_alive and player.is_alive:
418                 player.show_stats()
419                 computer.show_stats()
420                 #Formatting
421
422             print("-----")
423
424             #If the player survived the battle, increment battles_won
425             if player.is_alive:
426                 game.battles_won += 1
427
428             #The player has finally fainted
429             print("\nPoor " + player.name + " has fainted...")
430             print("But not before defeating " + str(game.battles_won) + " Pykemon!")
431
432             #Ask the user if they want to play again

```

```
432     choice = input("Would you like to play again (y/n): ").lower()
433     if choice != 'y':
434         playing_main = False
435         print("Thank you for playing Pykemon!")
```