**1. Smoothie Class:**

class Smoothie:

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

self.ingredients = ingredients

def get\_cost(self):

prices = {

"Strawberries": 1.50,

"Banana": 0.50,

"Mango": 2.50,

"Blueberries": 1.00,

"Raspberries": 1.00,

"Apple": 1.75,

"Pineapple": 3.50

}

cost = sum(prices[ingredient] for ingredient in self.ingredients)

return f"${cost:.2f}"

def get\_price(self):

cost = sum({

"Strawberries": 1.50,

"Banana": 0.50,

"Mango": 2.50,

"Blueberries": 1.00,

"Raspberries": 1.00,

"Apple": 1.75,

"Pineapple": 3.50

}[ingredient] for ingredient in self.ingredients)

price = cost + cost \* 1.5

return f"${price:.2f}"

def get\_name(self):

name = [ingredient.replace("-berries", "-berry") for ingredient in self.ingredients]

name.sort()

if len(name) > 1:

return f"{' & '.join(name)} Fusion"

else:

return f"{' '.join(name)} Smoothie"

**2. Testpaper and Student Classes:**

class Testpaper:

def \_\_init\_\_(self, subject, markscheme, pass\_mark):

self.subject = subject

self.markscheme = markscheme

self.pass\_mark = pass\_mark

class Student:

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

self.tests\_taken = "No tests taken"

def take\_test(self, testpaper, answers):

correct\_answers = sum(1 for a, b in zip(answers, testpaper.markscheme) if a == b)

percentage = (correct\_answers / len(testpaper.markscheme)) \* 100

status = "Passed" if percentage >= float(testpaper.pass\_mark[:-1]) else "Failed"

self.tests\_taken = {testpaper.subject: f"{status}! ({percentage:.0f}%)"}

**3. Train Delay Management:**

class Train:

def \_\_init\_\_(self, destinations, expected\_time):

self.destinations = destinations

self.expected\_time = expected\_time

def manage\_delays(train, destination, delay):

if destination in train.destinations:

hours, minutes = map(int, train.expected\_time.split(":"))

minutes += delay

if minutes >= 60:

minutes -= 60

hours += 1

train.expected\_time = f"{hours:02d}:{minutes:02d}"

**4. Minecart Track Management:**

class Minecart:

def \_\_init\_\_(self, velocity=0):

self.velocity = velocity

def interact(self, track):

if track == "-->":

if self.velocity < 8:

self.velocity += 2.67

elif track == "<-->":

pass

elif track == "<--":

self.velocity = max(0, self.velocity - 2.67)

elif track == "---":

self.velocity = max(0, self.velocity - 1)

def mine\_run(tracks):

cart = Minecart()

for index, track in enumerate(tracks):

cart.interact(track)

if cart.velocity == 0:

return index

return True

**5. Rectangle Intersection:**

class Rectangle:

def \_\_init\_\_(self, x, y, width, height):

self.x = x

self.y = y

self.width = width

self.height = height

def intersecting(rect1, rect2):

return not (rect1.x + rect1.width < rect2.x or

rect2.x + rect2.width < rect1.x or

rect1.y + rect1.height < rect2.y or

rect2.y + rect2.height < rect1.y)