**1.** The user enters 50 numbers. For each number, the user has a choice (a or b) of doing one of the following: a) Multiply the number by 6. b) Halve the number Should the user not enter a choice of  'a' or 'b', an appropriate message must be printed and the answer must equal 0. Print the number, choice and answer.

**Proposed Solution**

numbers = []

for i in range(50):

    num = int(input("Enter a number: "))

    choice = input("Enter a choice (a or b): ")

    if choice == 'a':

        answer = num \* 6

    elif choice == 'b':

        answer = num / 2

    else:

        print("Invalid choice. Answer set to 0.")

        answer = 0

    print("Number:", num, "Choice:", choice, "Answer:", answer)

**2.** Input five numbers into an array. Once all the numbers have been input, do the following:

-If the index of the element is less than the element value, replace the element with the index value squared.

 -If the index of the element is equal to the element value, add 10 to the element.

 -If the index of the element is greater than the element value, replace the element with  its square root value.

After making all the changes,

print the values in the array

 put comments in each line of code

**Proposed Solution 2**

import math  # import the math module for the square root function

students = []  # create an empty list to store the student records

num\_students = 0  # initialize the number of students to 0

female\_distinction\_count = 0  # initialize the count of female students who passed with distinction to 0

total\_percentage = 0  # initialize the total percentage to 0

# loop until the user enters a student number of 000 or until we process at least 20 students

while num\_students < 20:

    student\_num = input("Enter student number (000 to stop): ")

    if student\_num == "000":  # check if the user wants to stop

        break

    name = input("Enter student name: ")

    gender = input("Enter student gender (M/F): ")

    grade1 = float(input("Enter grade for assignment 1: "))

    grade2 = float(input("Enter grade for assignment 2: "))

    grade3 = float(input("Enter grade for assignment 3: "))

    # calculate the percentage for the student

    percentage = (grade1 + grade2 + grade3) / 3

    # check the conditions and modify the percentage accordingly

    if percentage >= 90:

        percentage += 5

    elif percentage >= 80:

        percentage += 2

    # create a student record and add it to the list of students

    record = [student\_num, name, gender, percentage]

    students.append(record)

    num\_students += 1

    # check if the student is a female who passed with distinction

    if gender == "F" and percentage >= 80:

        female\_distinction\_count += 1

    # add the percentage to the total percentage for all students

    total\_percentage += percentage

# calculate the average percentage for all students

avg\_percentage = total\_percentage / num\_students

# print the number of female students who passed with distinction and the average percentage for all students

print("Number of female students who passed with distinction:", female\_distinction\_count)

print("Average percentage for all students:", avg\_percentage)

# loop through the list of students and make the necessary changes to each record

for i in range(len(students)):

    percentage = students[i][3]  # get the percentage for the student

    if i < percentage:  # if the index is less than the percentage, replace the percentage with the index squared

        percentage = i \*\* 2

    elif i == percentage:  # if the index is equal to the percentage, add 10 to the percentage

        percentage += 10

    else:  # if the index is greater than the percentage, replace the percentage with its square root

        percentage = math.sqrt(percentage)

    students[i][3] = percentage  # update the percentage for the student in the record

# print the final student records

for record in students:

    print(record)

**3.** Question

7. Grade students at a college on the basis of marks gained against possible marks.

The user must enter the student number, marks gained, possible marks and gender (m/f).

Calculate the percentage scored for each student.

 Print a report showing each student’s number, percentage mark and a comment, corresponding to the student’s percentage, as follows:

 • 0–59% Fail

• 60–79% Pass

 • 80–100% Distinction Processing is to continue until a student number of 000 is entered.

A minimum of 20 students must be processed.

At the end of processing, print the number of female students who passed with distinction and print the average percentage for all the students.

 PUT COMMENTS IN EACH LINE OF CODE

**PROPOSED SOLUTION**

# Initialize counters for female students who passed with distinction and total percentage

female\_distinction\_count = 0

total\_percentage = 0

# Initialize counter for number of students processed

students\_processed = 0

# Process students until student number of 000 is entered or 20 students are processed

while True:

    # Prompt user to enter student number, marks gained, possible marks, and gender

    student\_number = input("Enter student number (000 to exit): ")

    if student\_number == '000' or students\_processed >= 20:

        break

    marks\_gained = int(input("Enter marks gained: "))

    possible\_marks = int(input("Enter possible marks: "))

    gender = input("Enter gender (m/f): ")

    # Calculate percentage score

    percentage = marks\_gained / possible\_marks \* 100

    # Determine comment based on percentage score

    if percentage < 60:

        comment = "Fail"

    elif percentage < 80:

        comment = "Pass"

    else:

        comment = "Distinction"

        # Increment female\_distinction\_count if student is female and scored distinction

        if gender == 'f':

            female\_distinction\_count += 1

    # Print student number, percentage score, and comment

    print("Student number:", student\_number)

    print("Percentage:", percentage)

    print("Comment:", comment)

    # Increment total\_percentage and students\_processed counters

    total\_percentage += percentage

    students\_processed += 1

# Calculate average percentage for all students

average\_percentage = total\_percentage / students\_processed

print("Average percentage:", average\_percentage)

print("Number of female students who passed with distinction:", female\_distinction\_count)

**4.** A company wishes to print a report for each of its products.

The user enters the product number, name, unit price, stock level and reorder level for each product.

Include a check for the unit price, to ensure that it is greater than zero, before continuing.

 The stock level for each item must be compared to the reorder level.

 If the stock level is less than the reorder level, the difference between the two values is calculated and printed together with a message to the user.

 The name, unit price and stock level for each product must be printed.

 Processing stops when the user enters a value of 0 for the product number.

 Calculate and print the average price of all products once all the records have been processed. put comments in each line of code.

**PROPOSED SOLUTION**

# Initialize variables for total price and number of products

total\_price = 0

num\_products = 0

# Process products until product number of 0 is entered

while True:

    # Prompt user to enter product details

    product\_number = int(input("Enter product number (0 to exit): "))

    if product\_number == 0:

        break

    product\_name = input("Enter product name: ")

    # Check unit price is greater than zero

    while True:

        unit\_price = float(input("Enter unit price: "))

        if unit\_price > 0:

            break

        print("Error: unit price must be greater than zero")

    stock\_level = int(input("Enter stock level: "))

    reorder\_level = int(input("Enter reorder level: "))

    # Check if stock level is less than reorder level

    if stock\_level < reorder\_level:

        difference = reorder\_level - stock\_level

        print("Warning: Stock level is below reorder level by", difference)

    # Print product details

    print("Product name:", product\_name)

    print("Unit price:", unit\_price)

    print("Stock level:", stock\_level)

    # Increment total price and number of products counters

    total\_price += unit\_price

    num\_products += 1

# Calculate average price of all products

average\_price = total\_price / num\_products

print("Average price:", average\_price)

**5.** A local shopping mall does a survey of the shopping habits of their shoppers.

Input values are the shopper's number, name, age and the time spent shopping (in minutes).

 Convert the minutes it took shoppers to complete their shopping into hours and minutes.

 Hint: to accumulate the minutes use “minutes = time MOD 60” and for hours use “hours = (time – minutes) / 60”.

 Print the total number of shoppers that are older than 18 as well as those that are 18 or younger.

 Also print the total number of shoppers who are older than 18 and spent more than three hours shopping.

If this total is more than the total number of shoppers older than 18 spending less than three hours, a message must be printed: “Most people spend over 3 hours at our mall.”

 If any of the shoppers spent six hours or more shopping, print a message:

“You were shopping for too long.” together with their time (in hours and minutes).

 Determine and print the age of the oldest shopper in the following format: “Oldest shopper’s age: ”

 Processing continues until the number 999 has been entered.

 use comments in each line of code

**Proposed Solution**

# Initialize variables

total\_shoppers = 0

over\_18 = 0

over\_18\_3hrs = 0

oldest\_age = 0

# Continue processing until shopper number 999 is entered

while True:

    # Get input values

    shopper\_number = int(input("Shopper number: "))

    if shopper\_number == 999:

        break

    shopper\_name = input("Shopper name: ")

    shopper\_age = int(input("Shopper age: "))

    shopping\_time = int(input("Time spent shopping (in minutes): "))

    # Convert shopping time to hours and minutes

    minutes = shopping\_time % 60

    hours = (shopping\_time - minutes) / 60

    # Print shopping time in hours and minutes

    print("Shopping time: {} hours and {} minutes".format(int(hours), minutes))

    # Check if shopper is over 18

    if shopper\_age > 18:

        over\_18 += 1

        # Check if shopper spent more than 3 hours shopping

        if hours >= 3:

            over\_18\_3hrs += 1

    # Check if shopper spent 6 or more hours shopping

    if hours >= 6:

        print("You were shopping for too long. Shopping time: {} hours and {} minutes".format(int(hours), minutes))

    # Check if shopper is oldest

    if shopper\_age > oldest\_age:

        oldest\_age = shopper\_age

    # Increment total shoppers

    total\_shoppers += 1

# Print results

print("Total shoppers over 18: {}".format(over\_18))

print("Total shoppers 18 or younger: {}".format(total\_shoppers - over\_18))

print("Total shoppers over 18 spending more than 3 hours: {}".format(over\_18\_3hrs))

if over\_18\_3hrs > over\_18 - over\_18\_3hrs:

    print("Most people spend over 3 hours at our mall.")

print("Oldest shopper's age: {}".format(oldest\_age))

**6.**  There are six boutiques in an exclusive clothing chain of stores, and each boutique employs three salespersons.

 • Determine which salesperson made the most sales in the group.

Print the salesperson number and the boutique number. Assume there is only one salesperson with the highest number of sales.

• Print the average number of sales for each boutique with the boutique number.

• Print the boutique number of the boutique that had the lowest average sales. Assume there is only one boutique with the lowest average.

• Print the value of the fewest average sales made by a boutique.

• Calculate the overall average sales for the boutiques and print this figure. •

 Print the total number of sales made by all the boutiques.

 The number of sales for each salesperson must be loaded into an array. put comments in each line of code

**PROPOSED SOLUTION**

# Initialize variables

num\_boutiques = 6

num\_salespersons = 3

sales = [] # 2D array to hold sales data

total\_sales = 0

avg\_sales\_boutiques = [0] \* num\_boutiques # list to hold average sales for each boutique

min\_avg\_sales\_boutique = 0

min\_avg\_sales = 0

max\_salesperson = 0

max\_salesperson\_boutique = 0

# Loop through each boutique and salesperson

for i in range(num\_boutiques):

    sales.append([]) # add an empty list for the current boutique

    for j in range(num\_salespersons):

        # Get input for current salesperson's sales

        sales[i].append(int(input("Enter sales for salesperson {} in boutique {}: ".format(j+1, i+1))))

        # Update total sales and average sales for the current boutique

        total\_sales += sales[i][j]

        avg\_sales\_boutiques[i] += sales[i][j]

        # Check if current salesperson has the highest sales

        if sales[i][j] > sales[max\_salesperson\_boutique][max\_salesperson]:

            max\_salesperson = j

            max\_salesperson\_boutique = i

    # Calculate average sales for the current boutique

    avg\_sales\_boutiques[i] /= num\_salespersons

    # Check if the current boutique has the lowest average sales

    if avg\_sales\_boutiques[i] < avg\_sales\_boutiques[min\_avg\_sales\_boutique]:

        min\_avg\_sales\_boutique = i

        min\_avg\_sales = avg\_sales\_boutiques[i]

# Print results

print("Salesperson {} in boutique {} made the most sales with {} sales.".format(max\_salesperson+1, max\_salesperson\_boutique+1, sales[max\_salesperson\_boutique][max\_salesperson]))

for i in range(num\_boutiques):

    print("The average number of sales for boutique {} is {:.2f}.".format(i+1, avg\_sales\_boutiques[i]))

print("Boutique {} had the lowest average sales with {:.2f} sales.".format(min\_avg\_sales\_boutique+1, min\_avg\_sales))

print("The fewest average sales made by a boutique was {:.2f}.".format(min(avg\_sales\_boutiques)))

print("The overall average sales for all boutiques is {:.2f}.".format(total\_sales / (num\_boutiques \* num\_salespersons)))

print("The total number of sales made by all the boutiques is {}.".format(total\_sales))

**7.** A computer training college has two campuses, Campus 1 and Campus 2.

Each campus can accommodate a maximum of 100 students and each   student has the choice of studying towards one of two diploma courses – Computer Science or Computer Technology.

 The course taken by each student at each campus must be entered into an array.

 Determine and print:

 •Which campus has the most students studying Computer Science.

 • Which campus has the fewest students studying Computer Technology.

• The average number of students per campus. Do not include error checks.

**PROPOSED SOLUTION**

# Define an array to store the number of students studying each course at each campus

# The first dimension of the array represents the campus (2 campuses), and the second dimension represents the course (2 courses)

num\_students = [[0, 0], [0, 0]]

# Loop through each campus

for campus in range(2):

    # Loop through each student

    for student in range(100):

        # Prompt the user to enter the course of the student at the current campus

        course = int(input(f"Enter the course of student {student+1} at Campus {campus+1} (1 for Computer Science, 2 for Computer Technology): "))

        # Increment the count of students for the selected course at the selected campus

        num\_students[campus][course-1] += 1

# Calculate the total number of students at each campus

total\_students = [sum(num\_students[campus]) for campus in range(2)]

# Determine which campus has the most students studying Computer Science

most\_cs\_campus = 1 if num\_students[1][0] > num\_students[0][0] else 2

# Determine which campus has the fewest students studying Computer Technology

fewest\_ct\_campus = 1 if num\_students[1][1] < num\_students[0][1] else 2

# Calculate the average number of students per campus

avg\_students = sum(total\_students) / len(total\_students)

# Print the results

print(f"Campus {most\_cs\_campus} has the most students studying Computer Science.")

print(f"Campus {fewest\_ct\_campus} has the fewest students studying Computer Technology.")

print(f"The average number of students per campus is {avg\_students}.")

**8.** A company supplies 100 products from 10 different suppliers.

The price of each product from a supplier is entered into a two-dimensional array.

 Determine the following:

 • The most expensive product.

 • The lowest average product price from a supplier.

• The average price of all products from all suppliers.

**PROPOSED SOLUTION**

# Define a two-dimensional array to store the prices of the products from each supplier

# The first dimension of the array represents the supplier (10 suppliers), and the second dimension represents the product (100 products)

product\_prices = [[0 for product in range(100)] for supplier in range(10)]

# Loop through each supplier

for supplier in range(10):

    # Loop through each product

    for product in range(100):

        # Prompt the user to enter the price of the product from the current supplier

        price = float(input(f"Enter the price of Product {product+1} from Supplier {supplier+1}: "))

        # Store the price in the array

        product\_prices[supplier][product] = price

# Determine the most expensive product

max\_price = 0

max\_price\_product = None

for supplier in range(10):

    for product in range(100):

        if product\_prices[supplier][product] > max\_price:

            max\_price = product\_prices[supplier][product]

            max\_price\_product = (supplier, product)

print(f"The most expensive product is Product {max\_price\_product[1]+1} from Supplier {max\_price\_product[0]+1} at a price of {max\_price}.")

# Determine the lowest average product price from a supplier

lowest\_avg\_price = float('inf')

lowest\_avg\_price\_supplier = None

for supplier in range(10):

    avg\_price = sum(product\_prices[supplier]) / len(product\_prices[supplier])

    if avg\_price < lowest\_avg\_price:

        lowest\_avg\_price = avg\_price

        lowest\_avg\_price\_supplier = supplier

print(f"The supplier with the lowest average product price is Supplier {lowest\_avg\_price\_supplier+1} with an average price of {lowest\_avg\_price}.")

# Determine the average price of all products from all suppliers

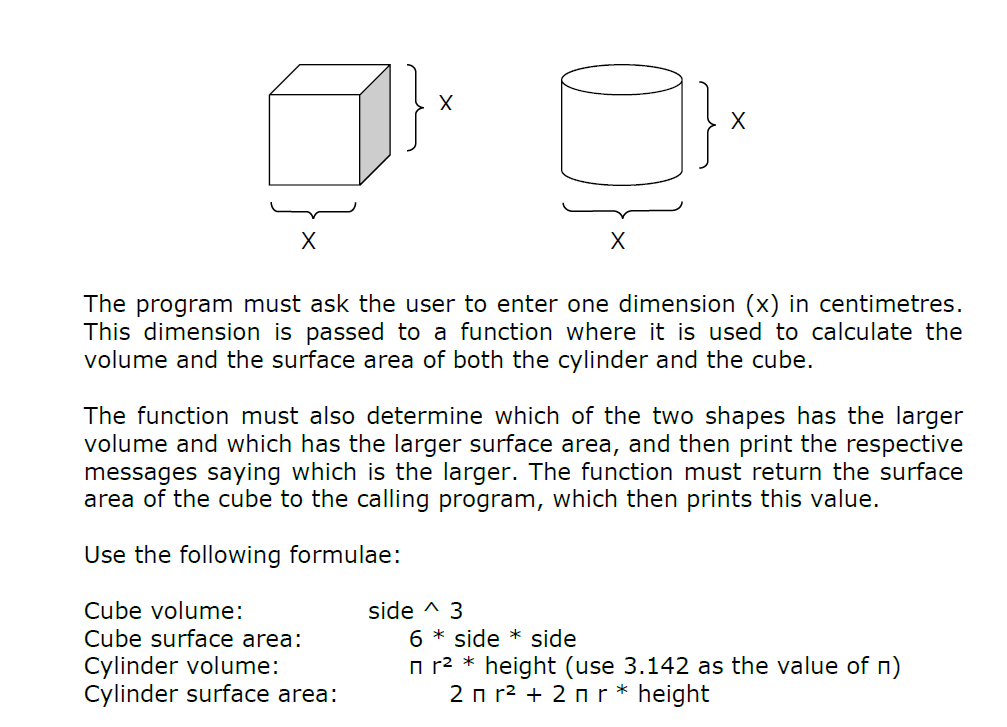
total\_price = sum(sum(product\_prices[supplier]) for supplier in range(10))

num\_products = len(product\_prices) \* len(product\_prices[0])

avg\_price = total\_price / num\_products

print(f"The average price of all products from all suppliers is {avg\_price}.")

**9.** Calculate the volume and surface area of a cube and a solid cylinder as illustrated:

****

**PROPOSED SOLUTION**

import math

# Define a function to calculate the volume and surface area of a cube and cylinder

def calculate\_shapes(x):

    # Calculate cube volume and surface area

    cube\_volume = x\*\*3

    cube\_surface\_area = 6 \* x\*\*2

    # Calculate cylinder volume and surface area

    cylinder\_volume = math.pi \* (x/2)\*\*2 \* x

    cylinder\_surface\_area = 2 \* math.pi \* (x/2)\*\*2 + 2 \* math.pi \* (x/2) \* x

    # Determine which shape has larger volume and surface area

    if cube\_volume > cylinder\_volume:

        print("Cube has larger volume")

    else:

        print("Cylinder has larger volume")

    if cube\_surface\_area > cylinder\_surface\_area:

        print("Cube has larger surface area")

    else:

        print("Cylinder has larger surface area")

    # Return surface area of cube

    return cube\_surface\_area

# Main program

x = float(input("Enter dimension in cm: "))

cube\_surface\_area = calculate\_shapes(x)

print("Surface area of cube is:", cube\_surface\_area)

**10.** A company allocates golfing hours to its directors at the end of each month.

 Write a program into which the current number of directors in the company is entered.

 This number of directors is then passed to a function, which requests the names and the total number of golfing hours accumulated by the start of the current month for each director.

During the month, the number of hours worked during a day by each director is added to a list. Assume no two directors work the same number of hours in a month.

These lists are used as input to the function to determine the total number of hours worked per director during the month (hours = 0 indicates the end of each director’s list).

Directors are given one hour’s golfing for every eight hours work they submit.

 The function prints a report for each director showing the director’s name, golfing hours accumulated by the start of the month, golfing hours earned during the month, and the total number of golfing hours accumulated at the end of the month.

 A summary at the end of the report shows the name and number of golfing hours of the director who has accumulated the most golfing hours. The number of directors who have more than 50 hours accumulated golfing time is returned to the main program, which prints this number. Showing your answer in one IDE code environment and putting comments in each line of code

**PROPOSED SOLUTION**

def calculate\_golfing\_hours(num\_directors):

    # Initialize variables

    max\_hours = 0

    max\_director = ""

    num\_directors\_over\_50\_hours = 0

    # Loop through each director and their hours

    for i in range(num\_directors):

        name = input(f"Enter the name of director {i+1}: ")

        total\_hours\_start\_month = int(input("Enter the total number of golfing hours accumulated by the start of the month: "))

        # Initialize daily hours list

        daily\_hours = []

        hours = int(input("Enter the number of hours worked on the first day (or 0 to end): "))

        while hours > 0:

            daily\_hours.append(hours)

            hours = int(input("Enter the number of hours worked on the next day (or 0 to end): "))

        # Calculate total hours worked in the month

        total\_hours\_month = sum(daily\_hours)

        golfing\_hours = total\_hours\_month // 8

        # Update max golfing hours

        if golfing\_hours > max\_hours:

            max\_hours = golfing\_hours

            max\_director = name

        # Count directors with more than 50 accumulated golfing hours

        if total\_hours\_start\_month // 8 > 50:

            num\_directors\_over\_50\_hours += 1

        # Print report for the director

        print(f"\n{name}'s Report:")

        print(f"Golfing hours accumulated by the start of the month: {total\_hours\_start\_month}")

        print(f"Golfing hours earned during the month: {golfing\_hours}")

        print(f"Total golfing hours accumulated at the end of the month: {total\_hours\_start\_month + golfing\_hours}")

    # Print summary report

    print(f"\nSummary Report:")

    print(f"Director with the most golfing hours: {max\_director} ({max\_hours} hours)")

    print(f"Number of directors with more than 50 accumulated golfing hours: {num\_directors\_over\_50\_hours}")

# Example usage:

calculate\_golfing\_hours(num\_directors=3)

**11.**  The government requires a program to compute the annual cost of running an embassy in a foreign country.

The following values for each embassy are entered: country, total monthly salary of all staff, monthly electricity bill, weekly entertainment allowance, annual building maintenance and the annual cost of security.

The government wants to know how many embassies cost them over R10 million a year.

 At the end of processing, the government also wants an average figure for the annual cost of running an embassy.

Processing must continue until a country of “ZZZ” is entered

**PROPOSED SOLUTION**

# Initialize variables

count = 0  # Counter for embassies costing over R10 million

total\_cost = 0  # Total cost of all embassies

num\_embassies = 0  # Number of embassies processed

# Loop to process embassies

while True:

    # Prompt user for embassy details

    country = input("Enter country name (or ZZZ to quit): ")

    if country == "ZZZ":

        break

    salary = float(input("Enter total monthly salary of all staff: "))

    electricity = float(input("Enter monthly electricity bill: "))

    entertainment = float(input("Enter weekly entertainment allowance: "))

    maintenance = float(input("Enter annual building maintenance cost: "))

    security = float(input("Enter annual cost of security: "))

    # Calculate annual cost of embassy

    monthly\_total = salary + electricity + (entertainment \* 4)  # multiply weekly entertainment by 4 to get monthly total

    annual\_total = (monthly\_total \* 12) + maintenance + security

    # Check if embassy cost over R10 million

    if annual\_total > 10000000:

        count += 1

    # Update total cost and number of embassies

    total\_cost += annual\_total

    num\_embassies += 1

# Calculate average cost of embassy

if num\_embassies > 0:

    avg\_cost = total\_cost / num\_embassies

else:

    avg\_cost = 0

# Print results

print(f"Number of embassies costing over R10 million: {count}")

print(f"Average annual cost of embassy: R{avg\_cost:.2f}")

**12.**  A retail store wants to calculate discount percentages for its store card members.

 The following data is input: card number, name, years of membership, purchase total Each member’s discount percentage must be calculated and a discount given on the following basis:

 • Members with a membership period of less than one year – 10% is deducted from their purchase total.

• Members with a membership period between one and four years – 15% is deducted from their purchase total, unless the purchase total is more than R1 000 – then 20% is deducted.

 • Members with a membership period of more than four years are allocated a 20% deduction.

 Print each customer’s name and card number and the final purchase total.

The printout should be in the following format: R1 000 \* 1 R100 \* 3 R10 \* 5 R1 \* 2 (For an amount which is R1 352). Processing continues until a name of “ZZZ” is entered.

**PROPOSED SOLUTION**

while True:

    # Input data for each customer

    card\_number = input("Enter card number (or 'ZZZ' to quit): ")

    if card\_number == "ZZZ":

        break

    name = input("Enter customer name: ")

    years\_of\_membership = int(input("Enter years of membership: "))

    purchase\_total = float(input("Enter purchase total: "))

    # Calculate discount percentage based on membership period and purchase total

    if years\_of\_membership < 1:

        discount\_percentage = 0.1

    elif 1 <= years\_of\_membership <= 4:

        if purchase\_total > 1000:

            discount\_percentage = 0.2

        else:

            discount\_percentage = 0.15

    else:

        discount\_percentage = 0.2

    # Calculate final purchase total after discount

    discount\_amount = purchase\_total \* discount\_percentage

    final\_total = purchase\_total - discount\_amount

    # Print customer information and final purchase total in the specified format

    print(f"\nName: {name}\nCard number: {card\_number}")

    print("Final purchase total:")

    for denom in [1000, 100, 10, 1]:

        count = int(final\_total / denom)

        final\_total -= count \* denom

        print(f"R{denom} \* {count}")

**13.** The Moscow Circus wants to know more about the attendance of their shows in South Africa.

They perform for three weeks, and have eight shows per week.

The number of people attending each show must be stored in an array.

 Determine the following:

 • The total number of people attending each week.

• The average number of people attending each week.

 • Overall average number of people per show.

 • The shows that had the best and worst attendance.

Print the week number and show number.

**PROPOSED SOLUTION**

# Initialize the attendance array with zeroes

attendance = [[0 for j in range(8)] for i in range(3)]

# Input the attendance numbers for each show

for week in range(3):

    for show in range(8):

        attendance[week][show] = int(input(f"Enter attendance for Week {week+1}, Show {show+1}: "))

# Calculate the total and average attendance for each week

weekly\_totals = [sum(attendance[week]) for week in range(3)]

weekly\_averages = [total/8 for total in weekly\_totals]

overall\_average = sum(sum(attendance, [])) / (3\*8)

# Find the best and worst shows based on attendance

best\_show = worst\_show = (0, 0)

for week in range(3):

    for show in range(8):

        if attendance[week][show] > attendance[best\_show[0]][best\_show[1]]:

            best\_show = (week, show)

        if attendance[week][show] < attendance[worst\_show[0]][worst\_show[1]]:

            worst\_show = (week, show)

# Print the results

print(f"Total attendance per week: {weekly\_totals}")

print(f"Average attendance per week: {weekly\_averages}")

print(f"Overall average attendance per show: {overall\_average:.2f}")

print(f"Best show: Week {best\_show[0]+1}, Show {best\_show[1]+1} with {attendance[best\_show[0]][best\_show[1]]} attendees")

print(f"Worst show: Week {worst\_show[0]+1}, Show {worst\_show[1]+1} with {attendance[worst\_show[0]][worst\_show[1]]} attendees")

**14.** Read, from a file, records containing the following fields: name, suburb, dog owner (yes/no) and dog licence (yes/no).

 The records of those people who live in Parkview or Parkhurst and who are dog owners without a dog licence must be written to a new file.

The number of records written to the new file must be accumulated and then printed.

 After all the records in the first file have been processed, print the contents of the new file.

**PROPOSED SOLUTION**

# Open the input file for reading

with open('input.txt', 'r') as input\_file:

    # Open the output file for writing

    with open('output.txt', 'w') as output\_file:

        # Initialize the counter for the number of records written to the new file

        count = 0

        # Loop over each line in the input file

        for line in input\_file:

            # Split the line into fields

            fields = line.strip().split(',')

            # Check if the person lives in Parkview or Parkhurst and is a dog owner without a dog licence

            if (fields[1] == 'Parkview' or fields[1] == 'Parkhurst') and fields[2] == 'yes' and fields[3] == 'no':

                # Write the line to the output file

                output\_file.write(line)

                # Increment the counter

                count += 1

        # Print the number of records written to the new file

        print(f"{count} records written to the output file.")

    # After the 'with' block, the output file is automatically closed

# After the 'with' block, the input file is automatically closed

# Open the output file for reading and print its contents

with open('output.txt', 'r') as output\_file:

    print(output\_file.read())

**15.** Read, from a file, records containing the following fields: name, rate and hours.

 • Calculate the gross pay using the formula: gross pay = rate \* hours

• If the gross pay is greater than R5 000, the name, write the rate and hours to a new file, otherwise print the record and the gross pay.

• After processing, print the records in the first file, the contents of the new file and the total number of records in the new file.

• Processing must continue until the end of the file. Showing your answer in one code IDE with full detailed comments per each line of code

**PROPOSED SOLUTION**

# Open the input file for reading

with open("input\_file.txt", "r") as input\_file:

    # Open the output file for writing

    with open("output\_file.txt", "w") as output\_file:

        # Initialize the counter for the new file

        new\_file\_counter = 0

        # Loop over each line in the input file

        for line in input\_file:

            # Split the line into the fields

            fields = line.strip().split(",")

            name = fields[0]

            rate = float(fields[1])

            hours = float(fields[2])

            # Calculate the gross pay

            gross\_pay = rate \* hours

            # Check if the gross pay is greater than R5 000

            if gross\_pay > 5000:

                # Write the record to the new file

                output\_file.write(f"{name},{rate},{hours}\n")

                # Increment the counter for the new file

                new\_file\_counter += 1

            else:

                # Print the record and the gross pay

                print(f"{name}, Gross pay: {gross\_pay}")

        # Print the total number of records in the new file

        print(f"Total number of records in the new file: {new\_file\_counter}")

# Close the input and output files

input\_file.close()

output\_file.close()