def q1(t1, t2):

# 1 pt

# Task: Calculate the Euclidean distance between two 2D points.

# Steps:

# 1. Extract the x and y coordinates from each tuple `t1` and `t2`.

x1, y1 = t1

x2 , y2 = t2

# 2. Compute the square of the difference between the x-coordinates and the y-coordinates.

x\_diff = (x2 - x1) \*\* 2

y\_diff = (y2 - y1) \*\* 2

# 3. Add these squared differences together.

diff\_sum = x\_diff + y\_diff

# 4. Take the square root of the sum to get the Euclidean distance.

sqrt = diff\_sum \*\* .5

return sqrt

# Example:

# Input: t1 = (0, 0), t2 = (3, 4)

# Output: 5.0 (The Euclidean distance between (0, 0) and (3, 4) is 5)

pass

def q1(t1, t2):

# 1 pt

# Task: Calculate the Manhattan distance between two 2D points.

# Steps:

# 1. Extract the x and y coordinates from each tuple `t1` and `t2`.

x1, y1 = t1

x2, y2 = t2

# 2. Compute the absolute difference between the x-coordinates and the y-coordinates.

x\_diff = abs(x2 - x1)

y\_diff = abs(y2 - y1)

# 3. Add these absolute differences together to get the Manhattan distance.

manhattan\_distance = x\_diff + y\_diff

return manhattan\_distance

# Example:

# Input: t1 = (2, 3), t2 = (5, 7)

# Output: 7

pass

def q1(t1, t2, t3):

# 1 pt

# Task: Calculate the area of a triangle given its three vertices in a 2D space.

# Steps:

# 1. Extract the x and y coordinates from each vertex (t1, t2, t3).

x1, y1 = t1

x2, y2 = t2

x3, y3 = t3

# 2. Use the formula to calculate the area of a triangle:

# Area = 0.5 \* abs(x1\*(y2-y3) + x2\*(y3-y1) + x3\*(y1-y2)).

area = 0.5 \* abs(x1 \* (y2 - y3) + x2 \* (y3 - y1) + x3 \* (y1 - y2))

# 3. Return the calculated area.

return area

# Example:

# Input: t1 = (0, 0), t2 = (4, 0), t3 = (0, 3)

# Output: 6.0 (The area of the triangle is 6)

def q3(list\_of\_integers):

# 1 pt

# Task: Find the maximum value in a list of integers using Pandas.

# Steps:

# 1. Convert the list of integers into a Pandas `Series`.

pandas\_series = pd.Series(list\_of\_integers)

# 2. Use the `.max()` method to return the largest integer in the series.

largest\_integer = pandas\_series.max()

return largest\_integer

# Example:

# Input: list\_of\_integers = [3, 5, 7, 2, 8]

# Output: 8 (The largest value in the list is 8)

pass

def q3(list\_of\_numbers):

# 1 pt

# Task: Calculate the sum of all even numbers in a list using Pandas.

# Steps:

# 1. Convert the list of numbers into a Pandas `Series`.

pandas\_series = pd.Series(list\_of\_numbers)

# 2. Use boolean indexing to filter even numbers and compute their sum.

even\_sum = pandas\_series[pandas\_series % 2 == 0].sum()

return even\_sum

# Example:

# Input: list\_of\_numbers = [1, 2, 3, 4, 5, 6]

# Output: 12

pass

def q6(df, column):

# 3 pts

# Task: Check if a specified column in a Pandas DataFrame contains any missing (`NaN`) values.

# Steps:

# 1. Use the `.isna()` method to check for missing values in the specified column.

missing\_values = df[column].isna()

# 2. Use `.sum()` to count how many `NaN` values exist.

sum\_missing\_values = missing\_values.sum()

# 3. If the count is greater than 0, return `True`; otherwise return `False`.

if sum\_missing\_values > 0:

return True

else:

return False

# Example:

# Input: df = pd.DataFrame({'A': [1, 2, None, 4]}), column = 'A'

# Output: True (The column 'A' has a NaN value)

pass

def q2(a, b, c):

# 1 pt

# Task: Check if the three integers are all even and if the first integer is the smallest.

# Steps:

# 1. Check if `a` is smaller than both `b` and `c`.

smaller = (a < b) and (a < c)

# 2. Verify if `a`, `b`, and `c` are all divisible by 2 (even numbers).

even\_numbers = (a % 2 == 0) and (b % 2 == 0) and (c % 2 == 0)

# 3. If both conditions are true, return `True`, otherwise return `False`.

return smaller and even\_numbers

# Example:

# Input: a = 2, b = 4, c = 6

# Output: True (2 is the smallest, and all values are even)

pass

def q4(list1, list2):

# 2 pts

# Task: Find the common elements between two lists.

# Steps:

# 1. Create an empty list to store the common elements.

common = []

# 2. Loop through each element in the first list.

for item in list1:

# 3. If the item is also in the second list, add it to the common list.

if item in list2:

common.append(item)

# 4. Return the list of common elements.

return list(set(common))

# Example:

# Input: list1 = [1, 2, 3, 4], list2 = [3, 4, 5, 6]

# Output: [3, 4] (The common elements between the lists)

pass

def q5(list\_of\_integers):

# 3 pts

# Task: Return a list of perfect squares from the input list of integers.

# Steps:

# 1. Define a helper function `is\_perfect\_square` to check if a number is a perfect square.

def is\_perfect\_square(num):

if num < 0:

return False

root = int(num\*\*0.5)

return root \* root == num

# 2. Use the helper function to filter perfect squares from the list.

perfect\_squares = [num for num in list\_of\_integers if is\_perfect\_square(num)]

return perfect\_squares

# Example:

# Input: list\_of\_numbers = [1, 4, 8, 9, 16, 20]

# Output: [1, 4, 9, 16]

pass

def q2(s):

# 1 pt

# Task: Check if a given string is a palindrome (reads the same forwards and backwards).

# Steps:

# 1. Remove all non-alphanumeric characters and convert the string to lowercase.

cleaned = ''

for char in s:

if char.isalnum():

cleaned += char.lower()

# 2. Compare the cleaned string to its reverse to check if it is a palindrome.

is\_palindrome = cleaned == cleaned[::-1]

# 3. Return True if it is a palindrome, otherwise return False.

return is\_palindrome

# Example:

# Input: s = "A man, a plan, a canal, Panama"

# Output: True (The string is a palindrome)

pass

def q10(str1, str2):

# 3 pts

# Task: Check if two strings are anagrams of each other (contain the same characters in any order).

# Steps:

# 1. Remove all non-alphanumeric characters and convert both strings to lowercase.

clean\_str1 = ''.join(c.lower() for c in str1 if c.isalnum())

clean\_str2 = ''.join(c.lower() for c in str2 if c.isalnum())

# 2. Sort the characters in both strings.

sorted\_str1 = sorted(clean\_str1)

sorted\_str2 = sorted(clean\_str2)

# 3. Compare the sorted strings and return `True` if they are equal, otherwise `False`.

return sorted\_str1 == sorted\_str2

# Example:

# Input: str1 = "listen", str2 = "silent"

# Output: True (The strings are anagrams)

pass

class A:

def \_\_init\_\_(self, a=None, b=None, c=None):

# 0 pts

# Task: Initialize an object with attributes `a`, `b`, and `c`.

# If any attribute is not provided, assign a random integer value between 1 and 10.

if a is None:

self.a = r.randint(1, 10)

else:

self.a = a

if b is None:

self.b = r.randint(1, 10)

else:

self.b = b

if c is None:

self.c = r.randint(1, 10)

else:

self.c = c

pass

def f1(self, other):

# 1 pt

# Task: Compare two instances of `A` to see if the current instance's values of `a`, `b`, and `c` are all greater than the other's values.

# Steps:

# 1. Compare `self.a`, `self.b`, and `self.c` with `other.a`, `other.b`, and `other.c`.

if (self.a > other.a) and (self.b > other.b) and (self.c > other.c):

return True

else:

return False

pass

def f4(self, others):

# 2 pts

# Task: Create a CSV file that contains the `a`, `b`, and `c` attributes of `self` and other instances.

# Steps:

# 1. Add `self` to the list of `others`.

all\_instances = [self] + others

# 2. Create a list of lists with the `a`, `b`, and `c` attributes of all instances.

all\_instances\_list = []

for i in all\_instances:

attributes = [i.a, i.b, i.c]

all\_instances\_list.append(attributes)

# 3. Convert this list into a DataFrame and save it as a CSV file named 'output.csv'.

df = pd.DataFrame(all\_instances\_list, columns=['a', 'b', 'c'])

df.to\_csv('f4\_output.csv', index=False)

pass

def computer\_choose():

"""

Randomly select a move for the computer.

"""

return r.choice([ROCK, PAPER, SCISSORS])

def player\_choose(automated, choice=None):

"""

Allow the player to choose a move or use automated mode for testing.

"""

if automated:

if choice is None:

return computer\_choose()

return choice

else:

try:

player\_input = int(input("Choose your move (1 for ROCK, 2 for PAPER, 3 for SCISSORS): "))

if player\_input in [ROCK, PAPER, SCISSORS]:

return player\_input

else:

return INVALID

except ValueError:

return INVALID

def determine\_winner(computer\_choice, player\_choice):

"""

Determine the winner based on the rules of Rock-Paper-Scissors.

"""

choices\_map = {ROCK: "ROCK", PAPER: "PAPER", SCISSORS: "SCISSORS"}

print(f"Player selected {choices\_map.get(player\_choice, 'INVALID')} and computer selected {choices\_map.get(computer\_choice, 'INVALID')}.")

if player\_choice == computer\_choice:

return TIE

elif (player\_choice == ROCK and computer\_choice == SCISSORS) or \

(player\_choice == PAPER and computer\_choice == ROCK) or \

(player\_choice == SCISSORS and computer\_choice == PAPER):

return PLAYER\_WINS

else:

return PLAYER\_LOSES

def check\_for\_another\_game(automated, answer=1):

"""

Ask if the player wants to continue playing or quit.

"""

if automated:

return answer

else:

try:

player\_input = int(input("Continue (1), or Quit (2)? "))

if player\_input in [CONTINUE, QUIT]:

return player\_input

else:

return INVALID

except ValueError:

return INVALID

def play\_game(automated, max\_wins=3):

"""

Main function to run the game loop.

"""

games = 0

player\_wins = 0

computer\_wins = 0

while player\_wins < max\_wins and computer\_wins < max\_wins:

computer\_choice = computer\_choose()

player\_choice = player\_choose(automated)

if player\_choice == INVALID:

print("Invalid choice. Please try again.")

continue

result = determine\_winner(computer\_choice, player\_choice)

games += 1

if result == PLAYER\_WINS:

print("Player wins this round!")

player\_wins += 1

elif result == PLAYER\_LOSES:

print("Computer wins this round!")

computer\_wins += 1

else:

print("It's a tie!")

print(f"Score: Player {player\_wins}, Computer {computer\_wins}, Games Played: {games}")

if player\_wins < max\_wins and computer\_wins < max\_wins:

continue\_game = check\_for\_another\_game(automated)

if continue\_game == QUIT:

print("Game ended by player.")

break

elif continue\_game == INVALID:

print("Invalid input. Ending the game.")

break

print("Game Over!")

if player\_wins > computer\_wins:

print("Player is the overall winner!")

elif computer\_wins > player\_wins:

print("Computer is the overall winner!")

else:

print("The game ends in a tie!")

if \_\_name\_\_ == "\_\_main\_\_":

"""

Start the game in manual mode if executed directly.

"""

play\_game(automated=False)

def q2(a, b):

# 2 pts

# Task: Safely divide two numbers and handle division by zero.

# Steps:

# 1. Use a try-except block to handle ZeroDivisionError.

# 2. If `b` is zero, return "Division by zero is not allowed."

# 3. Otherwise, return the result of a / b.

try:

result = a / b

except ZeroDivisionError:

return "Division by zero is not allowed"

return result

# Example:

# Input: a = 10, b = 0

# Output: "Division by zero is not allowed"

pass

def q4(df, col1, col2):

# 3 pts

# Task: Check if two columns in a DataFrame are inversely correlated.

# Steps:

# 1. Calculate the correlation coefficient between the two columns using `.corr()`.

# 2. Check if the coefficient is less than -0.5.

# 3. Return True if they are inversely correlated, otherwise return False.

correlation = df[col1].corr(df[col2])

return correlation < -0.5

# Example:

# Input: df = pd.DataFrame({'A': [1, 2, 3], 'B': [3, 2, 1]}), col1 = 'A', col2 = 'B'

# Output: True

pass

def q5(list\_of\_integers):

# 3 pts

# Task: Return a list of prime numbers from the input list of integers.

# Steps:

# 1. Define a helper function `is\_prime` to check if a number is a prime.

def is\_prime(num):

if num <= 1:

return False

for i in range(2, int(num\*\*0.5) + 1):

if num % i == 0:

return False

return True

# 2. Use the helper function to filter prime numbers from the list.

return [num for num in list\_of\_integers if is\_prime(num)]

# Example:

# Input: list\_of\_integers = [2, 3, 4, 5, 6, 5]

# Output: [2, 3, 5, 5]

pass

def q8(n):

# 3 pts

# Task: Generate the first `n` numbers in the Fibonacci sequence.

# Steps:

# 1. Create a list with the first two numbers of the sequence (0 and 1).

fib = [0, 1]

# 2. Use a loop to calculate the remaining numbers.

for \_ in range(2, n):

next\_number = fib[-1] + fib[-2]

fib.append(next\_number)

# 3. Return the first `n` numbers from the list.

return fib[:n]

# Example:

# Input: n = 6

# Output: [0, 1, 1, 2, 3, 5] (The first 6 Fibonacci numbers)

pass

def q5(words\_list):

# 3 pts

# Task: Find the longest word in a list of strings.

# Steps:

# 1. Iterate through the list and find the word with the maximum length.

# 2. Handle the case where there are ties by returning all longest words as a list.

max\_length = max(len(word) for word in words\_list)

longest\_words = [word for word in words\_list if len(word) == max\_length]

return longest\_words

# Example:

# Input: ["cat", "elephant", "dog", "dinosaur"]

# Output: ["elephant", "dinosaur"]

pass

def q1(t1, t2):

# 1 pt

# Task: Calculate the slope of the line passing through two points in 2D space.

# Steps:

# 1. Extract the x and y coordinates from each tuple `t1` and `t2`.

x1, y1 = t1

x2, y2 = t2

# 2. Calculate the change in y and the change in x.

dy = y2 - y1

dx = x2 - x1

# 3. Compute the slope using the formula slope = dy / dx.

# Note: Handle the case where dx = 0 to avoid division by zero.

if dx == 0:

return "undefined" # Vertical line

slope = dy / dx

# 4. Return the calculated slope.

return slope

# Example:

# Input: t1 = (1, 2), t2 = (4, 6)

# Output: 1.3333

pass

def q1(t1, t2):

# 1 pt

# Task: Calculate the midpoint between two points in 2D space.

# Steps:

# 1. Extract the x and y coordinates from each tuple `t1` and `t2`.

x1, y1 = t1

x2, y2 = t2

# 2. Calculate the midpoint coordinates using the formula:

# Midpoint = ((x1 + x2) / 2, (y1 + y2) / 2).

mid\_x = (x1 + x2) / 2

mid\_y = (y1 + y2) / 2

# 3. Return the midpoint as a tuple (mid\_x, mid\_y).

return (mid\_x, mid\_y)

# Example:

# Input: t1 = (0, 0), t2 = (4, 6)

# Output: (2.0, 3.0)

pass

def q1(t1, t2, t3):

# 1 pt

# Task: Determine if three points in 2D space are collinear.

# Steps:

# 1. Extract the x and y coordinates from each tuple `t1`, `t2`, and `t3`.

x1, y1 = t1

x2, y2 = t2

x3, y3 = t3

# 2. Use the slope formula to check if the slopes between pairs of points are equal:

# If (y2-y1)/(x2-x1) == (y3-y2)/(x3-x2), the points are collinear.

# Note: Handle the case where the denominator is zero (vertical line).

if (x2 - x1) == 0 or (x3 - x2) == 0:

return (x2 - x1) == (x3 - x2) # True if all are vertical

# 3. Calculate the slopes between points and compare.

slope1 = (y2 - y1) / (x2 - x1)

slope2 = (y3 - y2) / (x3 - x2)

# 4. Return True if the slopes are equal, else False.

return slope1 == slope2

# Example:

# Input: t1 = (0, 0), t2 = (2, 2), t3 = (4, 4)

# Output: True (The points are collinear)

pass

def q1(point):

# 1 pt

# Task: Reflect a point over the x-axis in 2D space.

# Steps:

# 1. Extract the x and y coordinates from the input `point`.

x, y = point

# 2. Negate the y-coordinate to reflect over the x-axis.

reflected\_point = (x, -y)

# 3. Return the new reflected point.

return reflected\_point

# Example:

# Input: point = (3, 4)

# Output: (3, -4)

pass

def q1(t1, t2, t3):

# 1 pt

# Task: Calculate the perimeter of a triangle given its three vertices in 2D space.

# Steps:

# 1. Extract the coordinates of each vertex `t1`, `t2`, and `t3`.

x1, y1 = t1

x2, y2 = t2

x3, y3 = t3

# 2. Compute the distances between each pair of vertices using the Euclidean distance formula.

d1 = ((x2 - x1) \*\* 2 + (y2 - y1) \*\* 2) \*\* 0.5

d2 = ((x3 - x2) \*\* 2 + (y3 - y2) \*\* 2) \*\* 0.5

d3 = ((x1 - x3) \*\* 2 + (y1 - y3) \*\* 2) \*\* 0.5

# 3. Add the distances to calculate the perimeter.

perimeter = d1 + d2 + d3

# 4. Return the calculated perimeter.

return perimeter

# Example:

# Input: t1 = (0, 0), t2 = (3, 0), t3 = (0, 4)

# Output: 12.0

pass

def q1(point):

# 1 pt

# Task: Rotate a point 90 degrees counterclockwise around the origin.

# Steps:

# 1. Extract the x and y coordinates from the input `point`.

x, y = point

# 2. Apply the rotation formula:

# New x = -y, New y = x.

rotated\_point = (-y, x)

# 3. Return the rotated point.

return rotated\_point

# Example:

# Input: point = (1, 2)

# Output: (-2, 1)

pass

def q1(t1, t2, t3):

# 1 pt

# Task: Calculate the centroid of a triangle given its three vertices in 2D space.

# Steps:

# 1. Extract the coordinates of each vertex `t1`, `t2`, and `t3`.

x1, y1 = t1

x2, y2 = t2

x3, y3 = t3

# 2. Calculate the centroid using the formula:

# Centroid = ((x1 + x2 + x3) / 3, (y1 + y2 + y3) / 3).

centroid\_x = (x1 + x2 + x3) / 3

centroid\_y = (y1 + y2 + y3) / 3

# 3. Return the centroid as a tuple (centroid\_x, centroid\_y).

return (centroid\_x, centroid\_y)

# Example:

# Input: t1 = (0, 0), t2 = (6, 0), t3 = (3, 6)

# Output: (3.0, 2.0)

pass

def q1(vector):

# 1 pt

# Task: Calculate the length (magnitude) of a vector in 2D space.

# Steps:

# 1. Extract the x and y components of the vector.

x, y = vector

# 2. Use the formula for vector length:

# Length = sqrt(x^2 + y^2).

length = (x \*\* 2 + y \*\* 2) \*\* 0.5

# 3. Return the calculated length.

return length

# Example:

# Input: vector = (3, 4)

# Output: 5.0

pass

def q1(list\_of\_numbers):

# 1 pt

# Task: Calculate the mean of a list of numbers using Pandas.

# Steps:

# 1. Convert the list of numbers into a Pandas `Series`.

pandas\_series = pd.Series(list\_of\_numbers)

# 2. Use the `.mean()` method to return the mean value.

mean\_value = pandas\_series.mean()

return mean\_value

# Example:

# Input: list\_of\_numbers = [10, 20, 30, 40, 50]

# Output: 30.0 (The mean of the list is 30.0)

pass

def q2(list\_of\_numbers):

# 1 pt

# Task: Calculate the median of a list of numbers using Pandas.

# Steps:

# 1. Convert the list of numbers into a Pandas `Series`.

pandas\_series = pd.Series(list\_of\_numbers)

# 2. Use the `.median()` method to return the median value.

median\_value = pandas\_series.median()

return median\_value

# Example:

# Input: list\_of\_numbers = [10, 20, 30, 40, 50]

# Output: 30.0 (The median of the list is 30.0)

pass

def q3(list\_of\_values):

# 1 pt

# Task: Count the frequency of each value in a list using Pandas.

# Steps:

# 1. Convert the list of values into a Pandas `Series`.

pandas\_series = pd.Series(list\_of\_values)

# 2. Use the `.value\_counts()` method to count the frequency of each unique value.

frequency\_counts = pandas\_series.value\_counts()

# 3. Return the frequency counts as a Pandas `Series`.

return frequency\_counts

# Example:

# Input: list\_of\_values = [1, 2, 2, 3, 3, 3]

# Output:

# 3 3

# 2 2

# 1 1

# dtype: int64

pass

def q4(list\_of\_numbers, threshold):

# 1 pt

# Task: Filter the values in a list that are greater than a given threshold using Pandas.

# Steps:

# 1. Convert the list of numbers into a Pandas `Series`.

pandas\_series = pd.Series(list\_of\_numbers)

# 2. Use a Boolean condition to filter values greater than the `threshold`.

filtered\_values = pandas\_series[pandas\_series > threshold]

# 3. Return the filtered values as a Pandas `Series`.

return filtered\_values

# Example:

# Input: list\_of\_numbers = [10, 20, 30, 40, 50], threshold = 25

# Output:

# 2 30

# 3 40

# 4 50

# dtype: int64

pass

def q5(list\_of\_numbers):

# 1 pt

# Task: Calculate the sum of all values in a list using Pandas.

# Steps:

# 1. Convert the list of numbers into a Pandas `Series`.

pandas\_series = pd.Series(list\_of\_numbers)

# 2. Use the `.sum()` method to calculate the sum of the values.

total\_sum = pandas\_series.sum()

return total\_sum

# Example:

# Input: list\_of\_numbers = [1, 2, 3, 4, 5]

# Output: 15 (The sum of the list is 15)

pass

import pandas as pd

import re

import math

def q1(mass, acceleration):

# 1 pt

# Task: Calculate the force acting on an object using the formula F = mass \* acceleration.

# Step 1: Multiply the `mass` and `acceleration` parameters.

# Example: If mass = 10 and acceleration = 9.8:

force = mass \* acceleration # 10 \* 9.8 = 98.0

# Step 2: Return the calculated force.

return force

# Example Output: q1(10, 9.8) = 98.0

def q2(value):

# 1 pt

# Task: Safely convert a value to a float.

# Step 1: Use a try block to attempt converting the input value to a float.

try:

float\_value = float(value) # Example: If value = "3.14", float\_value = 3.14.

return float\_value

except ValueError:

# Step 2: If a ValueError occurs, return "Conversion failed".

# Example: If value = "abc", the conversion fails.

return "Conversion failed"

# Example Output: q2("3.14") = 3.14, q2("abc") = "Conversion failed"

def q3(radius):

# 1 pt

# Task: Calculate the volume of a sphere.

# Step 1: Use the formula V = (4/3) \* pi \* radius³.

# Example: If radius = 3:

# volume = (4/3) \* math.pi \* (3 \*\* 3).

volume = (4 / 3) \* math.pi \* radius \*\* 3 # = 113.097

# Step 2: Return the calculated volume.

return volume

# Example Output: q3(3) = 113.097

def q4(list1, list2):

# 2 pts

# Task: Compare two lists of integers.

# Step 1: Check if the lengths of both lists are equal.

# Example: If list1 = [1, 2, 3] and list2 = [4, 5]:

same\_length = len(list1) == len(list2) # False

# Step 2: Check if the sum of all elements in `list1` is greater than the sum of elements in `list2`.

# Example: If sum(list1) = 6 and sum(list2) = 9:

list1\_sum\_greater = sum(list1) > sum(list2) # False

# Step 3: Return a dictionary with both results.

return {'same\_length': same\_length, 'list1\_sum\_greater': list1\_sum\_greater}

# Example Output: q4([1, 2, 3], [4, 5]) = {'same\_length': False, 'list1\_sum\_greater': False}

def q5(numbers):

# 3 pts

# Task: Filter even numbers from a list.

# Step 1: Use a loop to check each number in the input list.

# Example: If numbers = [1, 2, 3, 4, 5, 6], iterate through each number.

even\_numbers = [num for num in numbers if num % 2 == 0] # [2, 4, 6]

# Step 2: Return the new list containing only the even numbers.

return even\_numbers

# Example Output: q5([1, 2, 3, 4, 5, 6]) = [2, 4, 6]

def q6(string):

# 1 pt

# Task: Remove leading and trailing spaces from a string.

# Step 1: Use the .strip() method on the input string.

# Example: If string = " Hello, World! ":

stripped\_string = string.strip() # "Hello, World!"

# Step 2: Return the modified string.

return stripped\_string

# Example Output: q6(" Hello, World! ") = "Hello, World!"

def q7(email):

# 2 pts

# Task: Check if an email address is valid.

# Step 1: Define a regular expression pattern for valid email addresses.

pattern = r'^[\w\.-]+@[\w\.-]+\.\w+$'

# Step 2: Use re.match to check if the email matches the pattern.

# Example: If email = "user@example.com", the pattern matches.

is\_valid = bool(re.match(pattern, email)) # True

# Step 3: Return True if valid, otherwise False.

return is\_valid

# Example Output: q7("user@example.com") = True, q7("invalid\_email.com") = False

def q8(df, col):

# 3 pts

# Task: Replace all negative values in a DataFrame column with 0.

# Step 1: Use Pandas to access the specified column.

# Step 2: Replace negative values with 0 using the apply function.

# Example: If df = pd.DataFrame({'A': [1, -2, 3], 'B': [-4, 5, 6]}) and col = 'A':

df[col] = df[col].apply(lambda x: max(x, 0)) # Replace -2 with 0 in column 'A'.

# Step 3: Return the modified DataFrame.

return df

# Example Output:

# q8(pd.DataFrame({'A': [1, -2, 3], 'B': [-4, 5, 6]}), 'A') =

# A B

# 0 1 -4

# 1 0 5

# 2 3 6

def q9(volume):

# 2 pts

# Task: Calculate the buoyant force acting on an object.

# Step 1: Define constants for the formula.

density = 1000 # kg/m³ (default density of water)

gravity = 9.8 # m/s² (gravitational acceleration)

# Step 2: Multiply the parameters to calculate the force.

# Example: If volume = 2:

force = density \* volume \* gravity # 1000 \* 2 \* 9.8 = 19600.0

# Step 3: Return the calculated buoyant force.

return force

# Example Output: q9(2) = 19600.0

def q10(df, col1, col2):

# 2 pts

# Task: Check if two columns in a DataFrame are equal.

# Step 1: Use Pandas to compare values in `col1` and `col2`.

# Example: If df = pd.DataFrame({'A': [1, 2, 3], 'B': [1, 2, 4]}):

columns\_equal = df[col1].equals(df[col2]) # False

# Step 2: Return True if all values match, otherwise False.

return columns\_equal

# Example Output: q10(pd.DataFrame({'A': [1, 2, 3], 'B': [1, 2, 4]}), 'A', 'B') = False