Name: _____
Surname: _____
Date: _____

Debugging (15 points each)

Instructions

The functions below contain one or more errors. You have to find these errors, explain why they happen, and suggest a fix to them.

Questions

1. This code aims to count the number of unique words in a string, ignoring case sensitivity. However, it doesn't produce the correct count. Find the error and correct it.

```python
def count_unique_words(text):
    words = text.lower().split(" ")
    unique_words = set()
    for word in words:
        unique_words.add(word)
    return len(words)

sentence = "Hello world hello"
print(count_unique_words(sentence))  # Output: 3, Expected: 2
```

2. The following function is intended to find and return the maximum even number from a given list of integers. However, it sometimes returns incorrect results. Identify the issue(s) and propose a fix.

```python
def max_even_number(numbers: list[int]) -> int:
    max_even = None
    for num in numbers:
        if num % 2 == 0 and (max_even is None or num > max_even):
            max_even = num
        if max_even is None:
            return "No even number found"
        else:
            return max_even

nums = [1, 2, 3, 4]
print(max_even_number(nums))  # Prints "No even number found"

nums = [2, 4, 6]
print(max_even_number(nums))  # Prints 2
```

3. The following code should join three arrays together to form the first nine letters of the abc. When you run it, the last line returns an error: "`AttributeError: "numpy.ndarray" object has no attribute "concatenate"`". Explain the issue(s) and fix it.

```
import numpy as np

nm = np.array(["a", "b", "c"])
nn = np.array(["d", "e", "f"])
np = np.array(["g", "h", "i"])

nq = np.concatenate([nm, nn, np])
```

4. The function below should extract the antidiagonal of any square matrix. The code shows abnormal behaviour: it returns a result when a matrix is not square, and fails when the matrix is square. Explain the issue(s) and correct it.

```
import numpy as np

def antidiagonal(x: np.array) -> np.array:
    if x.ndim != 2:
        print("Array is not 2D")
    elif x.shape[0] != x.shape[1]:
        print("Matrix is not squared")

    result = []
    size = x.shape[0]

    for i in range(size):
        j = size - i
        result.append(x[i, j])

    return np.array(result)

b = np.array(
    [[1, 2, 3],
     [4, 5, 6]]
)
print(antidiagonal(b))
# Expected: Matrix is not squared
# Output (two lines):
# Matrix is not squared
# [3 5]

a = np.array(
    [[1, 2],
     [3, 4]]
)
print(antidiagonal(a))
# Expected: [2, 3]
# Output:
# IndexError: index 2 is out of bounds for axis 1 with size 2
```

Pseudo-Code

Instructions

In the following exercises, you will be asked to write Python functions by hand. Please follow these guidelines:

- Any build-in function you need is in the cheat-sheet provided.
- Clearly indicate each level of indentation using a right arrow (⇒).
- Ensure your code works correctly for any possible example.
- If your code has errors, it will be evaluated based on the provided docstring, comments, and explanations.

Full points will be awarded for code that performs as expected under all conditions.

5. Removing Characters (20 points)

Create the `remove_first_char()` function to take a string and return a new version. This version should remove all occurrences of the first character, except the first one itself. For example, "sun shines" becomes "sun hine" (all "s" except the first were removed). Note that the function treats lowercase and uppercase as distinct characters.

6. Circular Motion Coordinates (20 points)

Your task is to simulate the circular motion of an object by computing its coordinates. For a circle with radius R and an object moving along its circumference, you need to determine the object's x and y positions corresponding to a list of angles α (in degrees).

The coordinates are determined by the following formulas:

$$x = R \cdot \cos(\alpha)$$

$$y = R \cdot \sin(\alpha)$$

Part 1 (10 points): Implement the function `circular_motion(radius, angles)`:

- `radius`: A scalar value denoting the radius of the circle, R.
- `angles`: A list of angles in **degrees**, α.

The function should compute and return two NumPy arrays: `x_coords` and `y_coords`. These arrays represent the x and y coordinates on the circle for each angle in `angles`. Round this values to two decimals.

Part 2 (10 points): Implement a function `verify_coordinates(radius, x_coords, y_coords)` to verify the accuracy of your circular motion function. For each coordinate pair (x, y) in `x_coords` and `y_coords`, check if the equation $\sqrt{x^2 + y^2}$ equals the radius R within a reasonable tolerance (e.g., 0.01). This validation ensures that each calculated point lies on the circle. The function should return "True" if all points are on the circle and "False" otherwise.