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. The function below is designed to detect palindromes. It uses an auxiliary function inside. When we run the code, it raises an error on line 10: "`NameError: name "lower_and_join" is not defined`". Identify and fix the issue(s).

```
1  def is_palindrome(input_str: str) -> bool:
2
3      def lower_and_join(input_str: str) -> str:
4          return input_str.lower().replace(" ", "")
5
6      input_str = lower_and_join(input_str)
7      return input_str == input_str[::-1]
8
9  sentence = "Was it a car or a cat I saw"
10 print(lower_and_join(sentence))  # Expected: wasitacaroracatisaw
11 print(is_palindrome(sentence))  # Expected: True
```

2. The code below is intended to accumulate values in a list every time the function is called. However, it behaves unexpectedly. Identify and fix the issue(s).

```
1  def accumulate(value: int, lst: list=[]) -> list:
2      lst.append(value)
3      return lst
4
5  print(accumulate(1))  # Output: [1], Expected: [1]
6  print(accumulate(2))  # Output: [1, 2], Expected: [2]
7  print(accumulate(3, lst=[4]))  # Output: [4, 3], Expected: [4, 3]
```

3. The code below is intended to count the frequency of each character in a string. However, it raises a "`KeyError: "h"`" on the last line. Identify and fix the issue(s).

```python
def char_frequency(s: str) -> dict:
    freq = {}
    for char in s.lower():
        freq[char] += 1
    return freq

print(char_frequency("hello"))
# Expected: {"h": 1, "e": 1, "l": 2, "o": 1}
```

4. This function is intended to increment the first element of an array by 1, the second by 2, the third by 3, etc. However, it doesn't return the expected result. Identify the problem(s) and fix the code.

```python
import numpy as np

def increment_array(arr: np.array):
    i = 1
    for element in arr:
        element = element + i
        i += 1

numbers = np.array([1, 0, 1])
numbers = increment_array(numbers)
print(numbers)
# Expected Output: [2, 2, 4]
# Actual Output: None
```

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. Approximating Pi with Leibniz (20 points)

Write a function `approximate_pi` to estimate π using the Leibniz formula. This function should take one integer `n`, representing how many terms of the series to use. The function returns the estimated value of π.

$$\pi \approx 4 \times \sum_{k=0}^{n} \frac{(-1)^k}{2k+1}$$

6. Culinary Preferences in Spain (20 points)

During your class's culinary trip around Spain, you have a list of foods to be tasted in each city. However, some classmates have dietary restrictions or preferences and have listed foods they won't eat. Your task is to accommodate everyone's preferences as much as possible.
Create a function `foods_for_all()`. It takes two inputs:

- `foods_by_city` a dictionary with city names and their food lists.
- `food_restrictions` a list of tuples with students' names and the foods they avoid.

The function `foods_for_all()` returns a dictionary with each city's food options that all students can eat. *(15 points)*
If a city ends up with no options for everyone, it is not included in the output. *(3 points)*
Also, ensure the original inputs remain unchanged after running your function. *(2 points)*

```
# Sample input
# Remember: your function should work with any other possibility!
foods_by_city = {
  "Madrid": ["Squid", "Omelette", "Churros", "Soup"],
  "Valencia": ["Paella", "Horchata", "Fartons"],
  "Cadiz": ["Fried fish", "Potato salad", "Prawns"]}

food_restrictions = [
("Alice", ["Squid", "Paella", "Prawns"]),
("Bob", ["Churros", "Fartons"]),
("Charlie", ["Prawns"]),
("Diana", ["Fried fish", "Paella"]),
("Eve", ["Squid", "Fartons"]),
("Fred", ["Churros", "Omelette", "Fartons"]),
("Gwen", ["Fried fish", "Prawns"]),
("Hilda", ["Squid", "Potato salad"]),
]
```