Name: _____
Surname: _____
Date: _____

Instructions

Always assume each code is run independently, so variables do not persist between exercises.

Questions 1-6: Multiple Choice (5 points each)

On the table below, mark the correct answer for each question. Use an X to mark your choice. If you want to change your answer, fill the entire box and mark a new one. If multiple boxes are marked for the same question, no points will be awarded. See the examples below. Each question has **only one** correct answer. A correct answer awards 5 points. There is no penalty for wrong choices.

Example	A	B	C	D
Chosen A	X	☐	☐	☐
Chosen C, cancelled A	■	☐	X	☐
Chosen B, cancelled A and C	■	X	■	☐
Invalid (multiple choices)	X	X	☐	☐
Invalid (no choices)	■	☐	☐	☐

Questions 7-9: Debugging (10 points each)

These exercises show code cells with one or more errors. You have to find these errors, explain why they happen, and suggest a fix to them.

Questions 10-11: Open Problems (20 points each)

These are open-ended tasks. Explain your approach, outline algorithmic steps, and optionally provide compact working code snippets.

Mark your answers for Questions 1–6 below

Question	A	B	C	D
Q1	☐	☐	☐	☐
Q2	☐	☐	☐	☐
Q3	☐	☐	☐	☐
Q4	☐	☐	☐	☐
Q5	☐	☐	☐	☐
Q6	☐	☐	☐	☐

1. Which of the following statements about for and while loops in Python is TRUE?

   A. A while loop must always terminate after a fixed number of iterations.

   B. A while loop continues until its condition becomes `False`, regardless of how many iterations it takes.

   C. A for loop cannot be exited early using `break`.

   D. A for loop can be used only when the number of iterations is known in advance.

2. After running the following code, what will be the final value of variable "str2"?

```
str1 = "I love strings"

str_list = [char.upper() for char in str1 if char != " "]

str2 = "".join(str_list)
```

   A. `str2 = ‘‘I L O V E S T R I N G S’`

   B. `str2 = ‘‘I LOVE STRINGS’’`

   C. The code will raise an error.

   D. `str2 = ‘‘ILOVESTRINGS’’`

3. Why will the following code cell fail?

```
1  import numpy as np
2
3  arr = np.arange(4)
4  # Return evenly spaced values within a given interval.
5
6  arr_bool = np.array([True, False, True, False, True])
7  selected = arr[arr_bool]
```

   A. (Line 7) We cannot index an array (`arr`) with a boolean array (`arr_bool`).

   B. (Line 3) Invalid input for function `np.arange()`.

   C. (Line 6) Arrays cannot be initialized using boolean values.

   D. (Line 7) Boolean array does not match the indexed array.

4. Which of the following array declarations will cause an error?

```
1  import numpy as np
2
3  a = np.zeros(3).astype(float)
4  b = np.zeros(3, 3)
5  c = np.ones(5, dtype=bool)
6  d = np.ones((5, 5))
```

   A. (Line 5) Array `c`

   B. (Line 6) Array `d`

   C. (Line 3) Array `a`

   D. (Line 4) Array `b`

5. Which of the following lines will raise an error?

```
1 data = ("apple", ["x", "y"], ("red", "green"))
2
3 data[1].append("z")           # Line 2
4 print(data[2][1])             # Line 3
5 data[0][2] = "p"              # Line 4
```

   A. Line 4.

   B. None of them.

   C. Line 2.

   D. Line 3.

6. What is going to be the final value of "x"?

```
1  import numpy as np
2
3  x = np.array(
4    [[1, 2],
5    [3, 4]]
6  )
7  x[1, :] = 0
```

   A. $\begin{bmatrix} 1 & 2 \\ 0 & 4 \end{bmatrix}$

   B. $\begin{bmatrix} 1 & 0 \\ 3 & 0 \end{bmatrix}$

   C. $\begin{bmatrix} 0 & 2 \\ 1 & 4 \end{bmatrix}$

   D. $\begin{bmatrix} 1 & 2 \\ 0 & 0 \end{bmatrix}$

7. This function is designed to remove duplicate elements from a guest list for each event organized by a specific agency. Everything is contained in a dictionary where the keys are the events and the values are the guest lists. However, it does not remove them; it returns the same dictionary. Identify the problem(s) and fix the code.

```
def remove_duplicate_guests(agency_info: dict)->dict[list[str]]:
  for event, guests in agency_info.items():
    guests_removed_dupl = tuple(guests)

    agency_info[event] = guests

    return agency_info

agency_info = {
    "Summer Gala": [
        "Alice Johnson", "Mark Davis", "Sophia Lee",
        "Alice Johnson", "Mark Davis"
    ],
    "Tech Conference": [
        "Daniel Kim", "Olivia Brown", "Emma Wilson",
        "Daniel Kim", "Emma Wilson"
    ]
}

print(remove_duplicate_guests(agency_info))
#Expected Ouput: {"Summer Gala": ["Alice Johnson", "Mark Davis", "Sophia
    Lee"],
#"Tech Conference": ["Daniel Kim", "Olivia Brown", "Emma Wilson"]}

#Ouput: {"Summer Gala": ["Alice Johnson", "Mark Davis", "Sophia Lee", "
    Alice Johnson", "Mark Davis"],
#"Tech Conference": ["Daniel Kim", "Olivia Brown", "Emma Wilson", "Daniel
     Kim", "Emma Wilson"]}
```

8. The following function should mutliply the elements of an array by 2. However the code shows an unexpected result. Explain the issue and fix the code.

```
import numpy as np

def double_array(arr: np.array) -> np.array:
        # Multiply each element by 2
        arr_double = arr * 2
        return arr_double

print(double_array([1, 2, 3]))
# Expected output: [2, 4, 6]
# Actual output:   [1, 2, 3, 1, 2, 3]
```

How would `double_array()` work on NumPy arrays of 2 or more dimensions?

9. The following code is intended to replace all negative numbers in the array "`data`" with their absolute values using a boolean mask. However, the code raises an error `ValueError: NumPy boolean array indexing assignment cannot assign 7 input values to the 3 output values where the mask is true`. Explain the issue and fix the code.

```
import numpy as np

data = np.array([5, -3, 7, -1, 0, -6, 8])
mask = data < 0
data[mask] = -data
print(data)
# Expected output: [5, 3, 7, 1, 0, 6, 8]
# Actual output: ValueError
```

10. 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.

    If a city ends up with no options for everyone, it is not included in the output.

    Also, ensure the original inputs remain unchanged after running your function.

```python
# 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"]),
]
```

Continue exercise 11

11. You are given a list of player names and a fixed number of steps X. Each player begins at position 0 and performs a random walk of X steps. Write a function `random_walks()` that simulates this process. At each step, the player should either move $+1$ or -1, chosen uniformly at random. After completing the walk for each player, return a dictionary where each key is a player's name, and the corresponding value is that player's final position.

    Your function should accept the following arguments:

    - `players` (positional): A list or tuple of player names.
    - `num_steps` (keyword): The number of steps for each player's random walk (same number for all of them), defaulting to 10.
    - `random_seed` (keyword): If provided, set the random seed to ensure reproducible results.

    Additionally, if a player's name appears multiple times in the input list, append a "-" to subsequent occurrences to ensure that all dictionary keys are unique. For example, if "Alex" appears twice, the keys should be "Alex" and "Alex−". If "Bob" appears three times, the keys should be "Bob", "Bob−", and "Bob−−".

    See the code below for an example of input and output:

```
players = ["Alex", "Alex", "Bob", "Bob", "Bob", "Chia"]
x = 10

print(random_walks(players, num_steps=x))
# Example output (actual results will vary due to randomness):
# {"Alex": 2, "Alex-": 0, "Bob": -1, "Bob-": -3, "Bob--": 0, "Chia": 4}
```

Continue exercise 11