Name: _____
Surname: _____
Date: _____

Multiple Choice (10 points each)

Instructions

Circle the correct answer for each question, as each question has **only one** correct answer.
Provide reasoning for your answers in the space next to each question.
A correct answer always awards full points (+10 pts).
An incorrect answer with sensible reasoning awards half points (+5 pts).
An incorrect answer with faulty or no reasoning subtracts points (-3 pts).
No answer gives zero points (neither adds nor subtracts points).
Assume each code cell is **run in isolation**, meaning no other code has been run before it.

Questions

1. After running this code, what will be the final value of the **variable "a"**?

```
a = 4 - 4 / 0
a = abs(a)
```

   (a) `a` = ∞
   (b) `a = 0`
   (c) The code will raise an error.

2. Is there any difference between the commands in lines 2 and 3?

```
x = 3.14
y = str(x)  # Line 2
z = f"{x}"  # Line 3
```

   (a) Both commands produce the same output (meaning that `y = z`).
   (b) The commands produce a different output (meaning that `y` $\neq$ `z`).
   (c) One of them raises an error.

3. Which of the following lines of code will **raise an error**?

```
sports = ("football", "chess", "dancing")
sports[1] = "basketball"        # Line 2
print(f"I love {sports[-1]}")   # Line 3
print(sports[0][4])             # Line 4
```

   (a) Line 2.
   (b) Line 3.
   (c) Line 4.

4. Which message will **never be printed**, no matter the value of "name"?

```
name = "Daniel"  # Anything goes here

if name[0] == "d":
    print("Your name starts with d")
elif name[0].lower() == "d":
    print("Your name starts with D")
elif (type(name) == int) or (type(name) == float):
    print("A number is not a string!")
else:
    print("Your name starts with a letter other than d")
```

(a) "Your name starts with D"

(b) "A number is not a string!"

(c) "Your name starts with a letter other than d"

5. After running this code, what will be the final value of the **variable "count_up"**?

```
count_up = 1

while count_up <= 4:
    count_up += count_up
```

(a) `count_up = 4`

(b) `count_up = 8`

(c) The code will raise an error / never stops.

6. After running this code, what will be the length of "ls_end"?

```
def fibonacci(n: int, ls: list=[1, 1]) -> list:
    # Add 'n' more values to the list using Fibonacci logic
    # where x_(i) = x_(i-2) + x_(i-1)
    for _ in range(n):
        val = ls[-2] + ls[-1]
        ls.append(val)
    return ls

ls_test1 = fibonacci(3)
ls_test2 = fibonacci(3, ls=[1, 1, 2])
ls_end = fibonacci(3)
```

(a) `len(ls_end) <= 5`

(b) `len(ls_end) > 5`

(c) The code will raise an error.

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.

7. Safe Division (20 points)

Write a function named `safe_divide(numerator, denominator)`. This function must be able to handle specific edge cases and return values in specific formats as described below:

- If `denominator` is 0, the function should **print** the warning message `"Warning: Division by zero!"` and **return** 10^{10}. *(5 points)*
- If the result of the division is an integer (e.g., 10 divided by 5), the function should return an `int` value. *(10 points)*
- For all other cases, the function should return a `float` value representing the result of the division. *(5 points)*

8. Planning App (20 points)

You have a dictionary that lists people and their hobbies, as shown below:

```
dict_people = {
    "Amy": ["cinema", "dancing", "health", "history", "travel"],
    "Martha": ["health", "running", "travel"],
    "Rory": ["cinema", "dancing", "history", "running"]
}
```

Write a function named `match_people(dict_people)`. This function should take a dictionary (like the one above) as input and print all pairs of people who share at least one hobby. *(12 points)* For example, given the dictionary above, the output should be:

```
"Amy and Martha: health, travel"
"Amy and Rory: cinema, dancing, history"
"Martha and Rory: running"
```

Ensure the function handles errors gracefully. If the input is not a dictionary with lists of strings, the function should stop and print the message "Input is not the expected format". *(4 points)*
Avoid redundancy in the output. For instance, if "Amy and Martha" is printed, do not print "Martha and Amy". *(4 points)*