Name: _____
Surname: _____
Date: _____

Intructions

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

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

Circle the correct answer for each question, as each question has **only one** correct answer. A correct answer awards 5 points. There is no penalty for wrong 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)

In these exercises, you will encounter problems that are more open-ended and may not have a single correct answer. Your task is to explain how you would approach each problem by breaking it down into smaller, manageable steps.
Describe how you would structure your code, which logical steps are needed, and what kind of Python tools or constructs (e.g., loops, conditionals, functions, lists, etc.) you would use. You are not required to write full code, but doing so is encouraged if it helps clarify your ideas.

---

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

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

   (a) `a =` ∞
   (b) `a = 0`
   (c) `a =` $-\infty$
   (d) The code will raise an error.

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

```
1 x = 3.14
2 y = str(x)  # Line 2
3 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) Line 2 (y) raises an error.
   (d) Line 3 (z) 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.

   (d) None of them.

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".

   (d) All of them can be printed depending on "name".

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.

   (d) The code 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) `len(ls_end) > 5`

(d) The code will raise an error.

7. The following Python code is intended to print the factorials of numbers starting from 3, incrementing by 1 each time, and continuing until the user decides to stop. However, the code contains **three errors** that prevent it from functioning correctly. Identify and fix these errors.

   - Line 4 raises the error `SyntaxError: invalid syntax`.
   - The loop should print "`3!=6`" but instead prints "`True`".
   - After fixing the previous errors, the loop should continue with "`4!=24`", then print "`5!=120`", etc., but instead it prints "`3!=18`", then "`3!=54`"...

```
1  # Initialize the loop control and factorial variables
2  number = 3
3  factorial = 6
4  continue = True
5
6  while continue:
7    factorial = factorial * number
8    print(f"{number!=factorial}")
9    ask = input("Continue? (yes/no): ")
10   # If the user says "yes", proceed to the next number
11   if ask == "yes":
12     continue = True
13   else:
14     continue = False
```

8. The following code is intended to create two sets: one containing the original members of a group, and another containing both the original and new members. It then displays the difference between these sets. However, the output is not as expected. Explain the issue and fix the code.

```
group_before = {"Luke", "Han", "Leia"}
# Duplicate the original group and add the new members
group_now = group_before
for new_member in ["Ben", "Arturo"]:
  group_now.add(new_member)

# Compare the groups
print(group_now - group_before)
# Expected output: {"Arturo", "Ben"}
# Actual output: set()  <--- Empty set
```

9.

10. 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}.
    - If the result of the division is an integer (e.g., 10 divided by 5), the function should return an `int` value.
    - For all other cases, the function should return a `float` value representing the result of the division.

Continue exercise 10

11.

Continue exercise 11