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 we run the code below, what will be the value of "x"?

```
s = "1.5 to the power of 2 is 2.25"
x = 10 * float(s[-4:])
```

   (a) `x = -40.0`
   (b) `x = 15.0`
   (c) `x = 22.5`
   (d) The code will raise an error.

2. Which statement is **FALSE** regarding keyword arguments in Python?

   (a) Keyword arguments are defined by specifying the parameter name followed by an equal sign and the value.
   (b) Keyword arguments are always required.
   (c) You are not allowed to put keyword arguments before positional arguments.
   (d) Keyword arguments are used to set default parameter values.

3. After we run the code below, what will be the value of "x"?

```
s = "1.5 to the power of 2 is 2.25"
count = 0
for x in s:
  if x in "123456789":
    count += 1
```

(a) 0

(b) 5

(c) 6

(d) The code will raise an error.

4. Consider the following block of code. What will be the console's output?

```
def exercise(x: list) -> tuple:
  # Reverse the list
  rev = x[::-1]
  return tuple(rev)

result = exercise("hola")
```

(a) Python will print "hol".

(b) Python will print "aloh".

(c) The code runs fine but the user won't see any output.

(d) The code will raise an error.

5. Which one of these statements is **FALSE** about tuples?

(a) You can create a tuple with mixed data types.

(b) Tuples are ordered and can be indexed.

(c) Tuples are mutable.

(d) Tuples can contain duplicated values.

6. After we run the code below, what will be the value of "ls_new"?

```
ls = [1, 2, 3]
ls_new = []
while len(ls) > 0:
  x = ls[0]
  ls_new.extend([x, x])
  ls.remove(x)
```

(a) `ls_new = []`

(b) `ls_new = [1, 2, 3]`

(c) `ls_new = [1, 1, 2, 2, 3, 3]`

(d) The code will never end (infinite loop).

7. 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
```

8. 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).

```
def accumulate(value: int, lst: list=[]) -> list:
  lst.append(value)
  return lst

print(accumulate(1))  # Output: [1], Expected: [1]
print(accumulate(2))  # Output: [1, 2], Expected: [2]
print(accumulate(3, lst=[4]))  # Output: [4, 3], Expected: [4, 3]
```

9. 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
```

10. The mathematical constant e, approximately equal to 2.71828, is fundamental in many areas of mathematics. It can be defined in various ways, two of which are:

    (a) **Limit Definition**:

    $$e = \lim_{n \to N} \left(1 + \frac{1}{n}\right)^n$$

    (b) **Series Expansion** (Maclaurin Series for e^x at $x = 1$):

    $$e = \sum_{n=0}^{N} \frac{1}{n!} = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \dots$$

    Where $N \to \infty$. Your task is to implement two functions, one for each of the definitions above. Both functions will utilize the argument `n` as the upper limit for approximation (N in the equations above).

Continue exercise 10

11. 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"]
}
```

Design 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. 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"
```

Continue exercise 11