Group 1 | Group Assignment 1

Fahid Ansar | Gurmeet Deshwal | Abdul | Yaswanth Kumar

Part A

1. A function that takes a number and returns a list of its all possible factors (e.g., for 10, these are [1, 2, 5, 10]). Name this function "factors".

In [4]:
```
def factors (x):
    list_factors = []
    divisor = 1

    while divisor <= x:
        if x % divisor == 0:
            list_factors.append(divisor)
        divisor += 1
        list_factors.sort()

    return list_factors
factors(10)
```

Out[4]:
```
[1, 2, 5, 10]
```

2. A function that takes a number and returns True if the number is prime and False if the number is not prime (reuse the previous function!), e.g., for 7 it should return True, for 8 it should return False. Name this function "is_prime".

In [24]:
```
def is_prime(x):
    if len(factors(x)) > 2:
        return False
    else:
        return True

is_prime(7)
```

Out[24]:
```
True
```

3. A function that takes two numbers and returns their Highest Common Factor (e.g., for 12 and 18, it is 6). Name this function "hcf".

In [10]:
```
def hcf(a,b):
    return [max(item for item in factors(a) if item in factors(b))]
hcf(12,18)
```

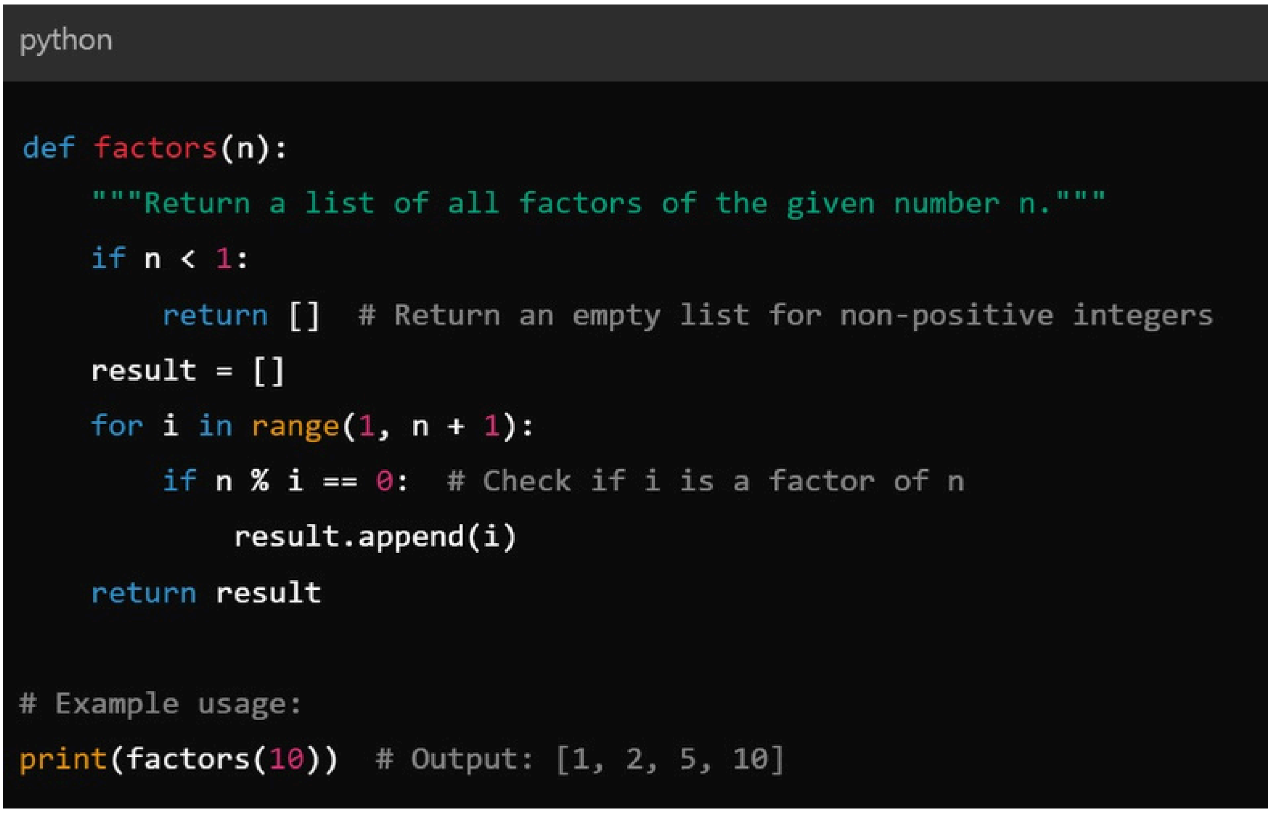
Out[10]:
```
[6]
```

Part B

Use a generative AI tool, e.g., ChatGPT, to generate a Python program that provides the same implementation as that you wrote in Part A above, then write a critical review of up-to 400 words (please use bullet points) comparing the quality of your own code against the program generated by the AI tool. In your comments, highlight the ways that the auto-generated code is different from yours, its strengths and weaknesses and the level and type of programs that these generative models may work well and may not.

1. A function that takes a number and returns a list of its all possible factors (e.g., for 10, these are [1, 2, 5, 10]). Name this function "factors".

- search keywords: please write a simple function that takes a number and returns a list of its all possible factors (e.g., for 10, these are [1, 2, 5, 10]). Name this function "factors".
- auto-generated codes:



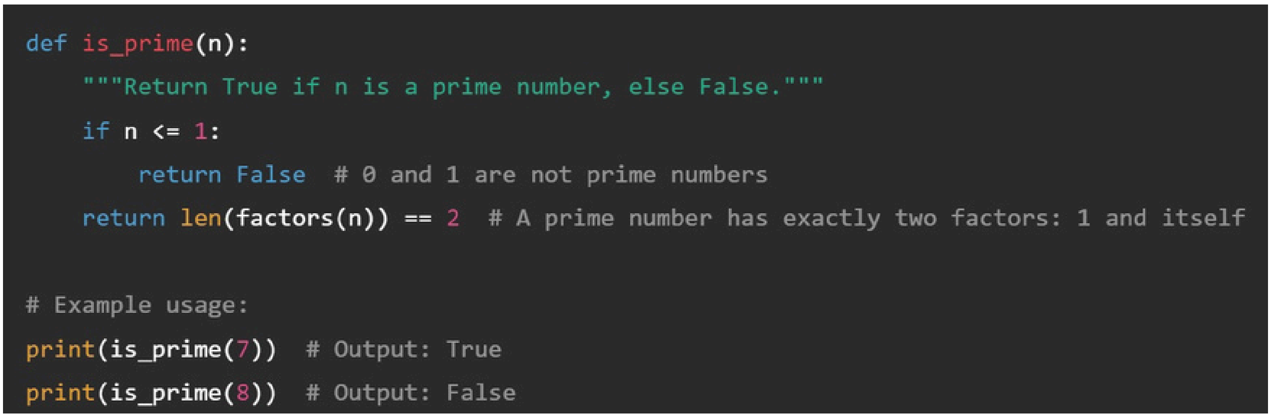
```python
def factors(n):
    """Return a list of all factors of the given number n."""
    if n < 1:
        return []  # Return an empty list for non-positive integers
    result = []
    for i in range(1, n + 1):
        if n % i == 0:  # Check if i is a factor of n
            result.append(i)
    return result

# Example usage:
print(factors(10))  # Output: [1, 2, 5, 10]
```

- evaluation and comparison commentsble
  1. Clarity: Our code uses x and divisor. ChatGPT uses n and i. Both are clear and serve the same purpose. But shorter names like n and i are more common.
  2. Performance: Our code uses a while loop manually increment divisor after each iteration. ChatGPT uses a for loop that automatically iterates through the numbers from 1 to n. Our code uses list_factors.sort() inside the loop after each iteration. Chatgpt doesn't have sorting because the factors are already naturally added in order due to the increasing loop variable i.

.

2. A function that takes a number and returns True if the number is prime and False if the number is not prime (reuse the previous function!), e.g., for 7 it should return True, for 8 it should return False. Name this function "is_prime".

- search keywords: please write a simple function that takes a number and returns True if the number is prime and False if the number is not prime (reuse the previous function!), e.g., for 7 it should return True, for 8 it should return False. Name this function "is_prime".
- auto-generated codes:

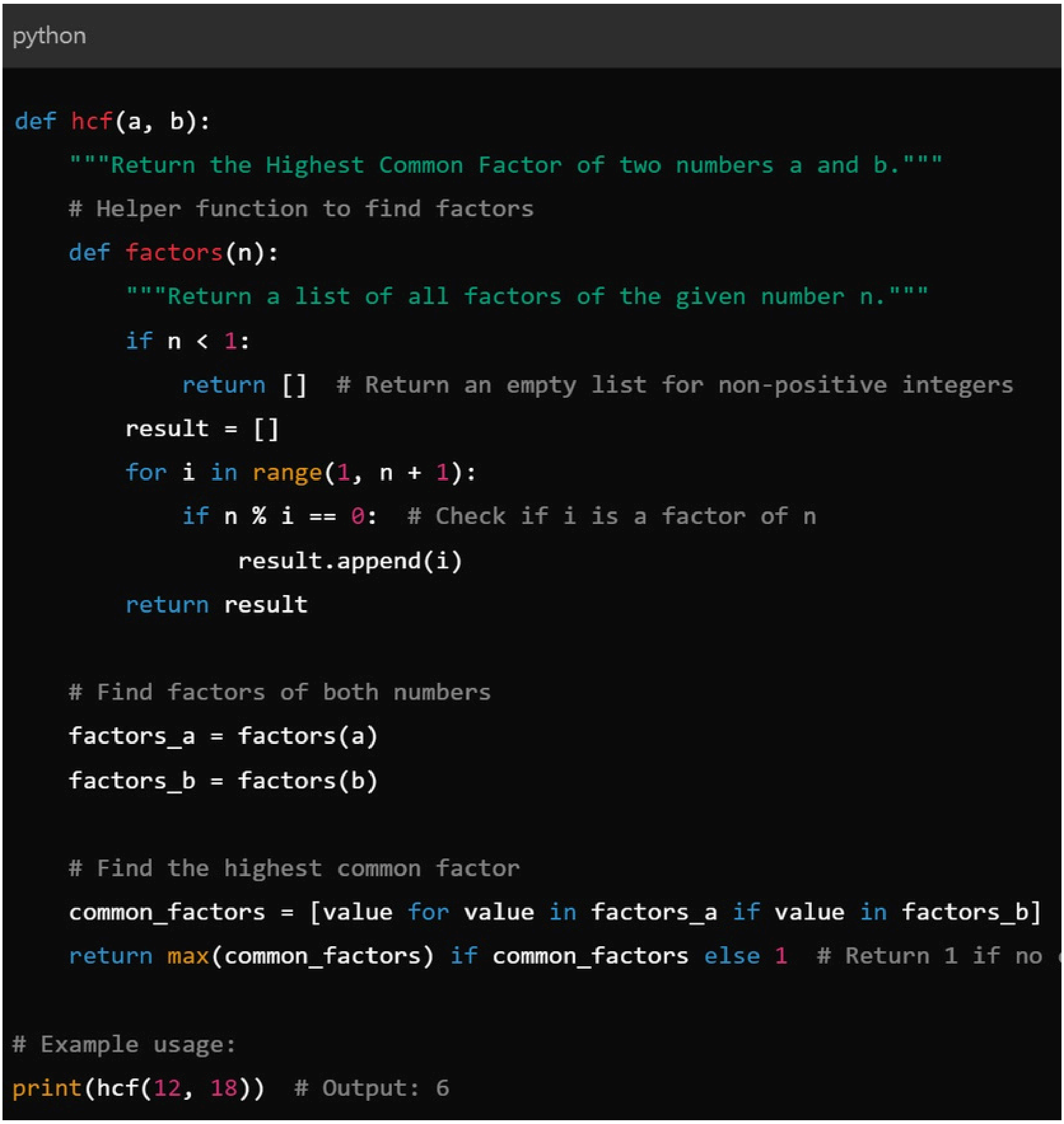


```python
def is_prime(n):
    """Return True if n is a prime number, else False."""
    if n <= 1:
        return False  # 0 and 1 are not prime numbers
    return len(factors(n)) == 2  # A prime number has exactly two factors: 1 and itself

# Example usage:
print(is_prime(7))  # Output: True
print(is_prime(8))  # Output: False
```

- evaluation and comparison comments:
  1. Clarity: ChatGPT is more explicit and clear. The use of a well-named factors function and clear comments make it easier to understand. Our code is shorter but relies on the assumption of a defined factors function without much explanation.
  2. Performance: Our code could have slightly better performance due to its shorter logic. It returns False as soon as more than two factors are found, avoiding unnecessary checks. ChatGPT always calculates all factors, even if more than two are found early on, which can slightly degrade performance for non-prime numbers.

3. A function that takes two numbers and returns their Highest Common Factor (e.g., for 12 and 18, it is 6). Name this function "hcf".

- search keywords:please write a simple function that takes two numbers and returns their Highest Common Factor (e.g., for 12 and 18, it is 6). Name this function "hcf".
- auto-generated codes:



python

```python
def hcf(a, b):
    """Return the Highest Common Factor of two numbers a and b."""
    # Helper function to find factors
    def factors(n):
        """Return a list of all factors of the given number n."""
        if n < 1:
            return []  # Return an empty list for non-positive integers
        result = []
        for i in range(1, n + 1):
            if n % i == 0:  # Check if i is a factor of n
                result.append(i)
        return result

    # Find factors of both numbers
    factors_a = factors(a)
    factors_b = factors(b)

    # Find the highest common factor
    common_factors = [value for value in factors_a if value in factors_b]
    return max(common_factors) if common_factors else 1  # Return 1 if no

# Example usage:
print(hcf(12, 18))  # Output: 6
```

- evaluation and comparison comments
  1. Clarity: Our Code clearly separates the factor calculation from the hcf function, making it easy to read and understand. ChatGPT contains comments and unnecessary parts that explain the code but can make it longer and more complex.
  2. Performance: Our Code uses a simple loop to find factors, which is straightforward. ChatGPT uses similar performance but includes extra checks that may slow down the execution process for larger numbers.

Common features in comparison:

1. Error Handling: Our code doesn't have a check for x < 1. If a negative number or zero is passed, the while loop would behave strangely or continue indefinitely. ChatGPT includes an edge case check for numbers less than 1. If n < 1, it returns an empty list.
2. Best Use Cases: Our code best for small projects where the factors function is already defined and is ideal for quick solutions that don't need any extra details. ChatGPT is suitable for teaching the code to someone due to clear explanations and error handling.