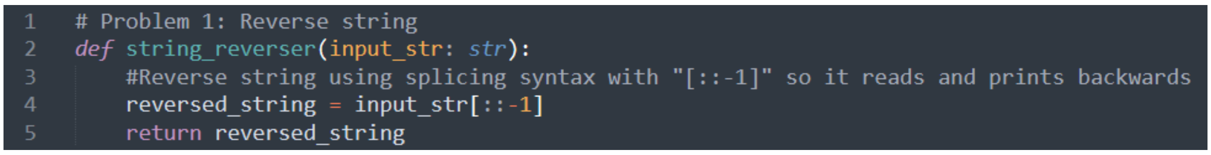
Mara Muda Pohan

Technical Test Task: Logic and Algorithm Implementation

Problem 1: Reverse a String

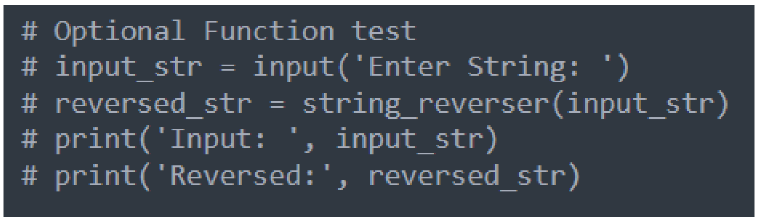
In this problem, I used the built in splicer syntax from python with ‘[::-1]’ to reverse the string.



```python
# Problem 1: Reverse string
def string_reverser(input_str: str):
    #Reverse string using splicing syntax with "[::-1]" so it reads and prints backwards
    reversed_string = input_str[::-1]
    return reversed_string
```

This function will return the results of the reversed string.

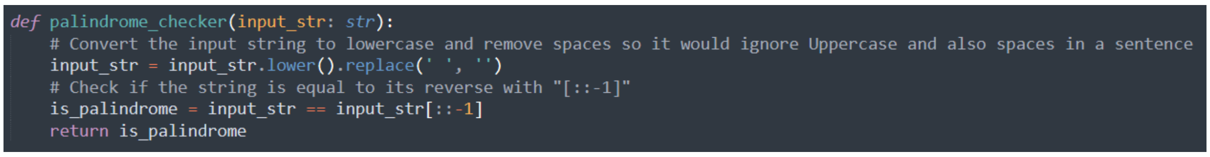
I also provided an optional function checker code that asks for user input. To use it simply highlight the code under ‘# Optional Function test’ and remove the comment or ‘#’ and run it with the function.



```python
# Optional Function test
# input_str = input('Enter String: ')
# reversed_str = string_reverser(input_str)
# print('Input: ', input_str)
# print('Reversed:', reversed_str)
```

Problem 2: Palindrome Check

In this problem I also used the splicer syntax with ‘[::-1]’ to read and print the input backwards. But the difference here is I normalized the data by making the string lowercase and removing the space.

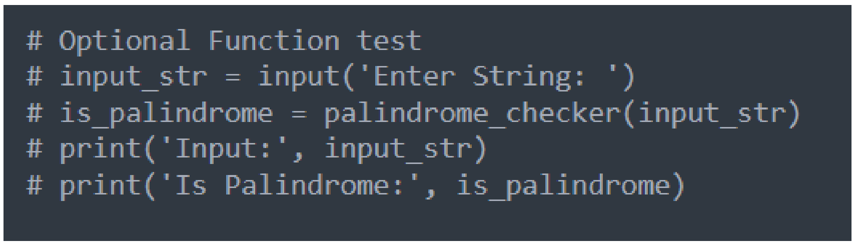


```python
def palindrome_checker(input_str: str):
    # Convert the input string to lowercase and remove spaces so it would ignore Uppercase and also spaces in a sentence
    input_str = input_str.lower().replace(' ', '')
    # Check if the string is equal to its reverse with "[::-1]"
    is_palindrome = input_str == input_str[::-1]
    return is_palindrome
```

Turning the string into a lowercase is done to anticipate situations such as “Madam” where the uppercase “M” is not equal to “m”.
Removing the spaces from a string is done to anticipate situations for longer sentence such as “Was it a car or a cat I saw” is not equal to “was I tac a ro rac a ti saW”.

This function will return “True” or “False”.

I also provided an optional function checker code that asks for user input. To use it simply highlight the code under '# Optional Function test' and remove the comment or '#' and run it with the function.



```
# Optional Function test
# input_str = input('Enter String: ')
# is_palindrome = palindrome_checker(input_str)
# print('Input:', input_str)
# print('Is Palindrome:', is_palindrome)
```

Problem 3: Prime Number Generator

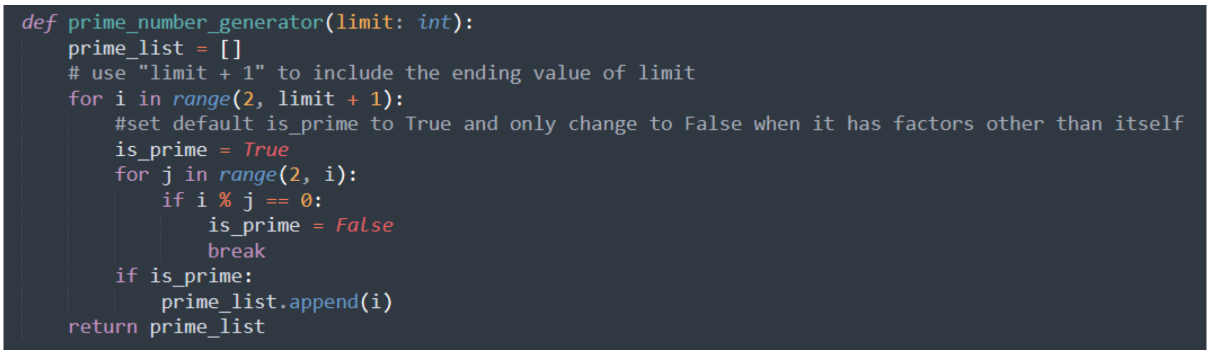
In this Problem, I opted for a *nested loop* where it uses modulus(%) to check if the numbers are divisible by numbers other than itself In the range.

I used this method because it's the simplest and safest method to check a prime number.

First, I set "is_prime" value to "True"

In the first *for loop* I included a "+1" in the limit so it will include the end value of the limit.

In the second *for loop* I did not use a "+1" so it won't include the end value in "if i % j == 0" condition. Whenever "if i % j == 0" condition is met, it will turn "is_prime" into "False" and stop the operation and move on to the next number. If the condition isn't fulfilled until the range is completed then it will append the result to the "prime_list".



```
def prime_number_generator(limit: int):
    prime_list = []
    # use "limit + 1" to include the ending value of limit
    for i in range(2, limit + 1):
        #set default is_prime to True and only change to False when it has factors other than itself
        is_prime = True
        for j in range(2, i):
            if i % j == 0:
                is_prime = False
                break
        if is_prime:
            prime_list.append(i)
    return prime_list
```

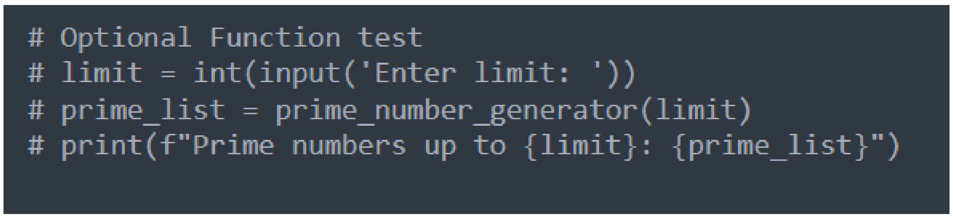
This function will return a list of prime numbers up to a given limit.

Although it is not the most efficient method, it is the safest method terms of accurateness. The other method is by using the 'math' library from python and use 'math.sqrt' function.

This is because if we use the 'math.sqrt' function from the 'math' library, we can eliminate half the range, But using square root in python will result in rounding down the square root which can potentially miss prime factors even if it is faster and more efficient.

In Addition to this, I also provided an optional function checker code that asks for user input. To use it simply highlight the code under '# Optional Function test' and remove the comment or '#' and run it with the function.

Mara Muda Pohan



```python
# Optional Function test
# limit = int(input('Enter limit: '))
# prime_list = prime_number_generator(limit)
# print(f"Prime numbers up to {limit}: {prime_list}")
```