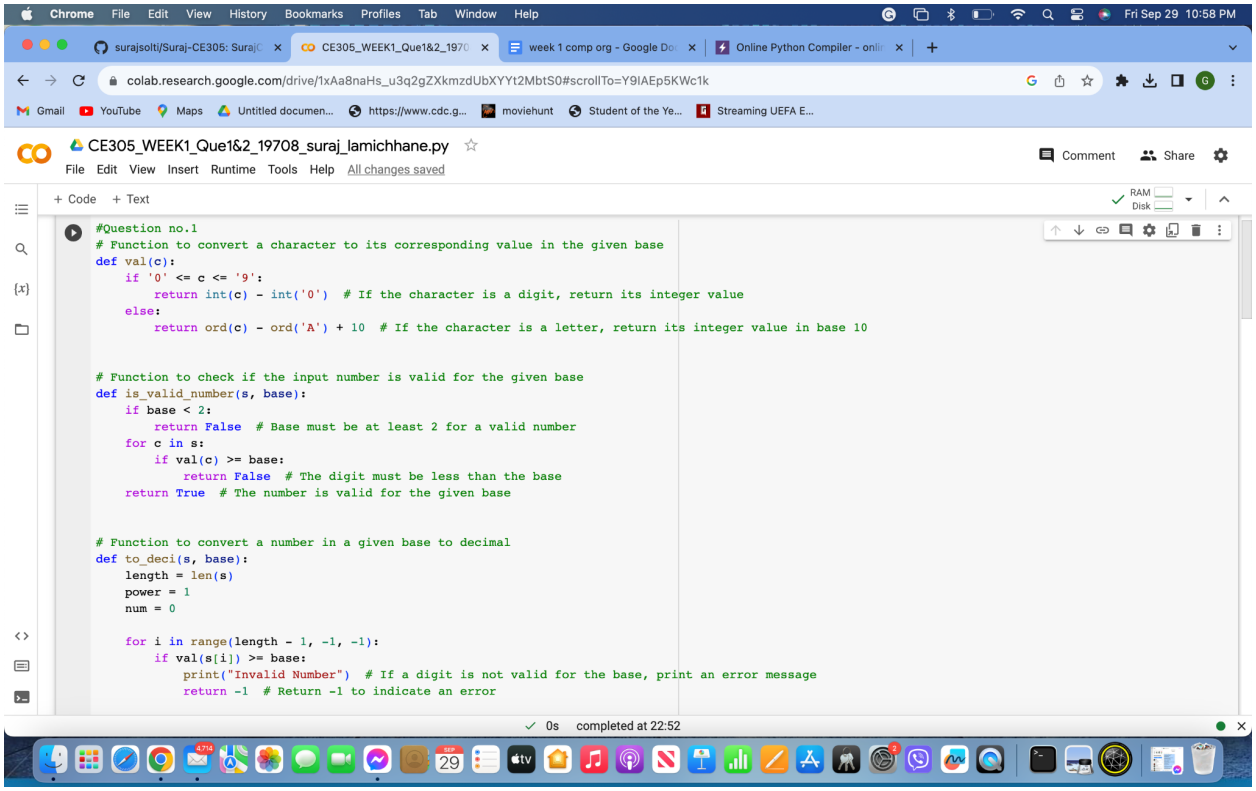
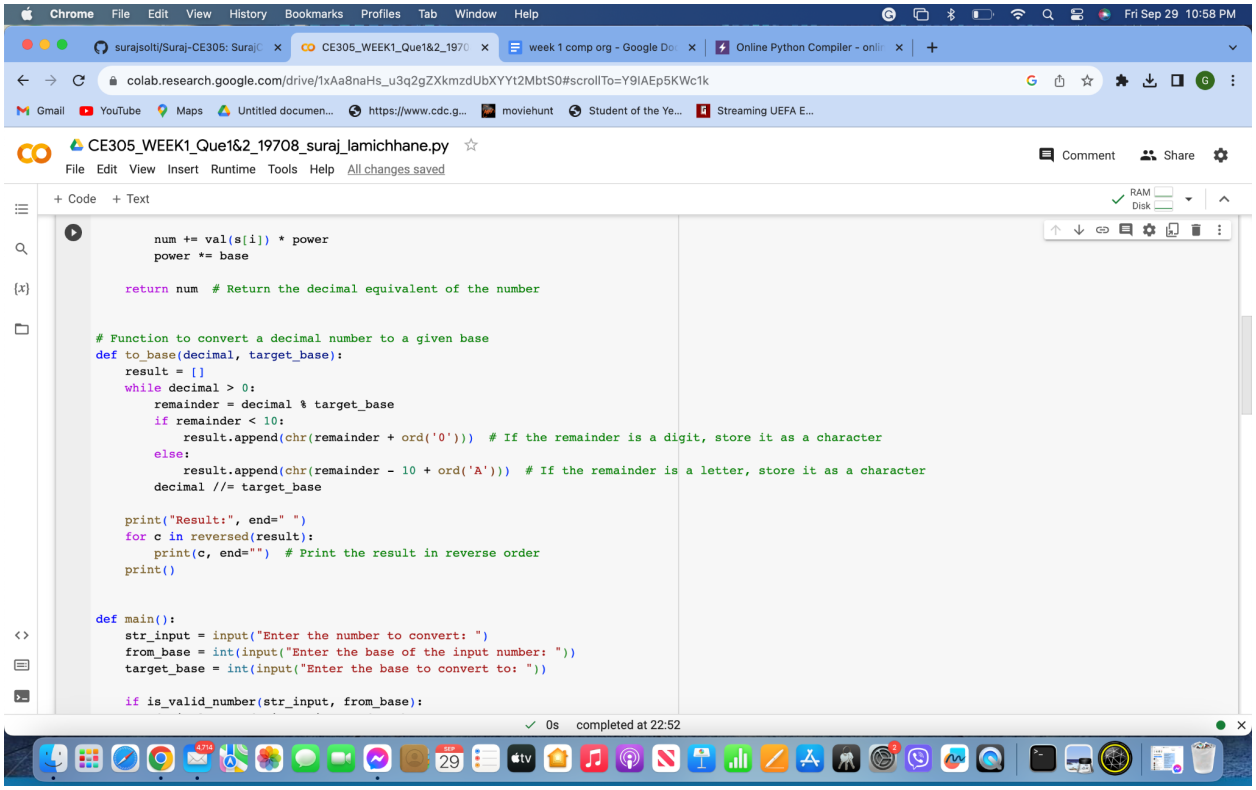
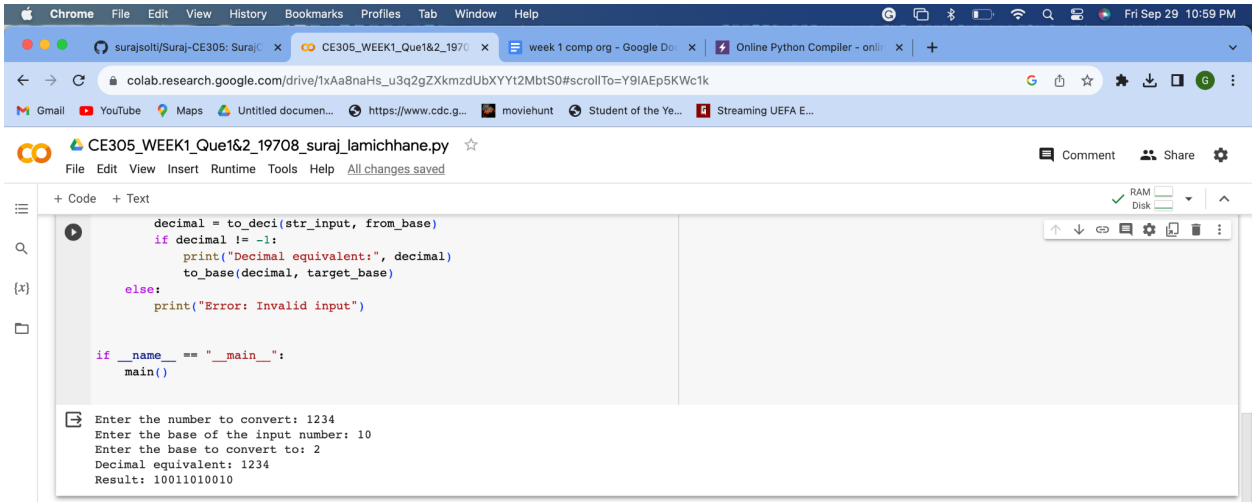
Qno.1)



```python
#Question no.1
# Function to convert a character to its corresponding value in the given base
def val(c):
    if '0' <= c <= '9':
        return int(c) - int('0')  # If the character is a digit, return its integer value
    else:
        return ord(c) - ord('A') + 10  # If the character is a letter, return its integer value in base 10


# Function to check if the input number is valid for the given base
def is_valid_number(s, base):
    if base < 2:
        return False  # Base must be at least 2 for a valid number
    for c in s:
        if val(c) >= base:
            return False  # The digit must be less than the base
    return True  # The number is valid for the given base


# Function to convert a number in a given base to decimal
def to_deci(s, base):
    length = len(s)
    power = 1
    num = 0

    for i in range(length - 1, -1, -1):
        if val(s[i]) >= base:
            print("Invalid Number")  # If a digit is not valid for the base, print an error message
            return -1  # Return -1 to indicate an error
```



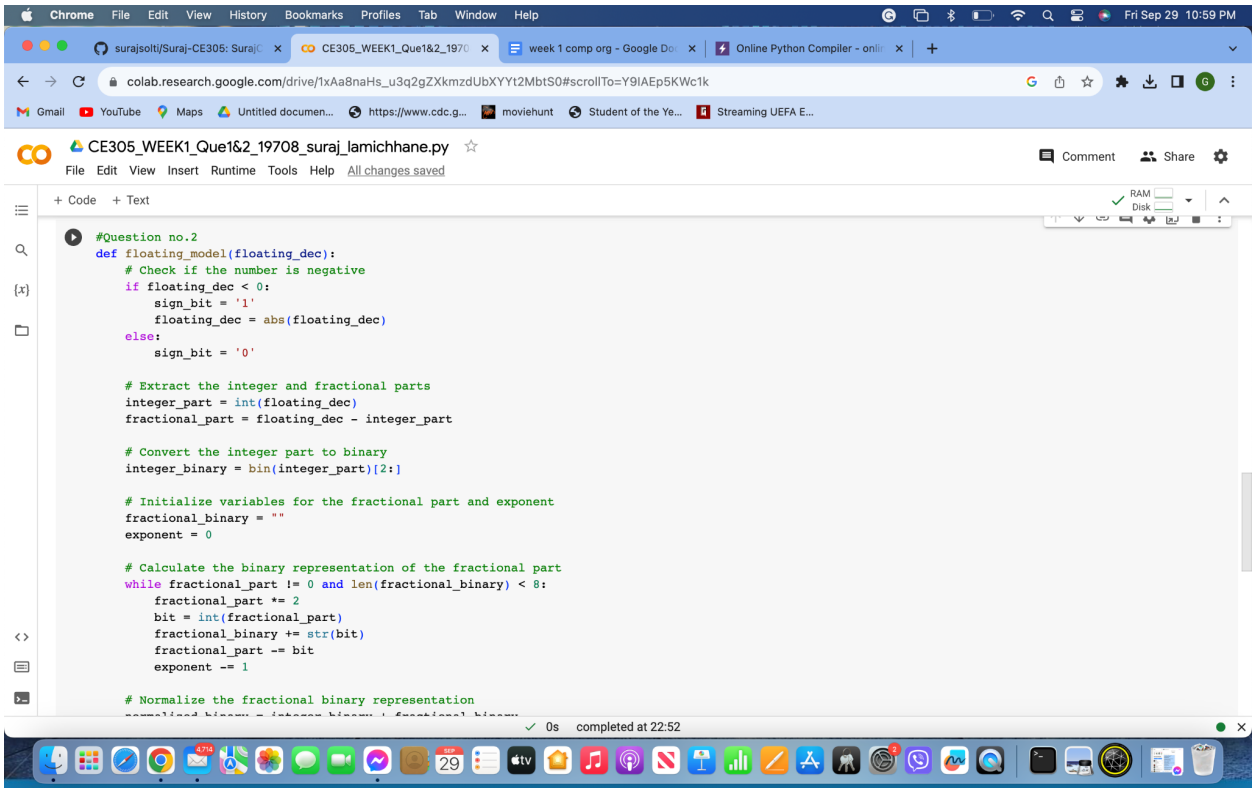
```
        num += val(s[i]) * power
        power *= base

    return num  # Return the decimal equivalent of the number


# Function to convert a decimal number to a given base
def to_base(decimal, target_base):
    result = []
    while decimal > 0:
        remainder = decimal % target_base
        if remainder < 10:
            result.append(chr(remainder + ord('0')))  # If the remainder is a digit, store it as a character
        else:
            result.append(chr(remainder - 10 + ord('A')))  # If the remainder is a letter, store it as a character
        decimal //= target_base

    print("Result:", end=" ")
    for c in reversed(result):
        print(c, end="")  # Print the result in reverse order
    print()


def main():
    str_input = input("Enter the number to convert: ")
    from_base = int(input("Enter the base of the input number: "))
    target_base = int(input("Enter the base to convert to: "))

    if is_valid_number(str_input, from_base):
```

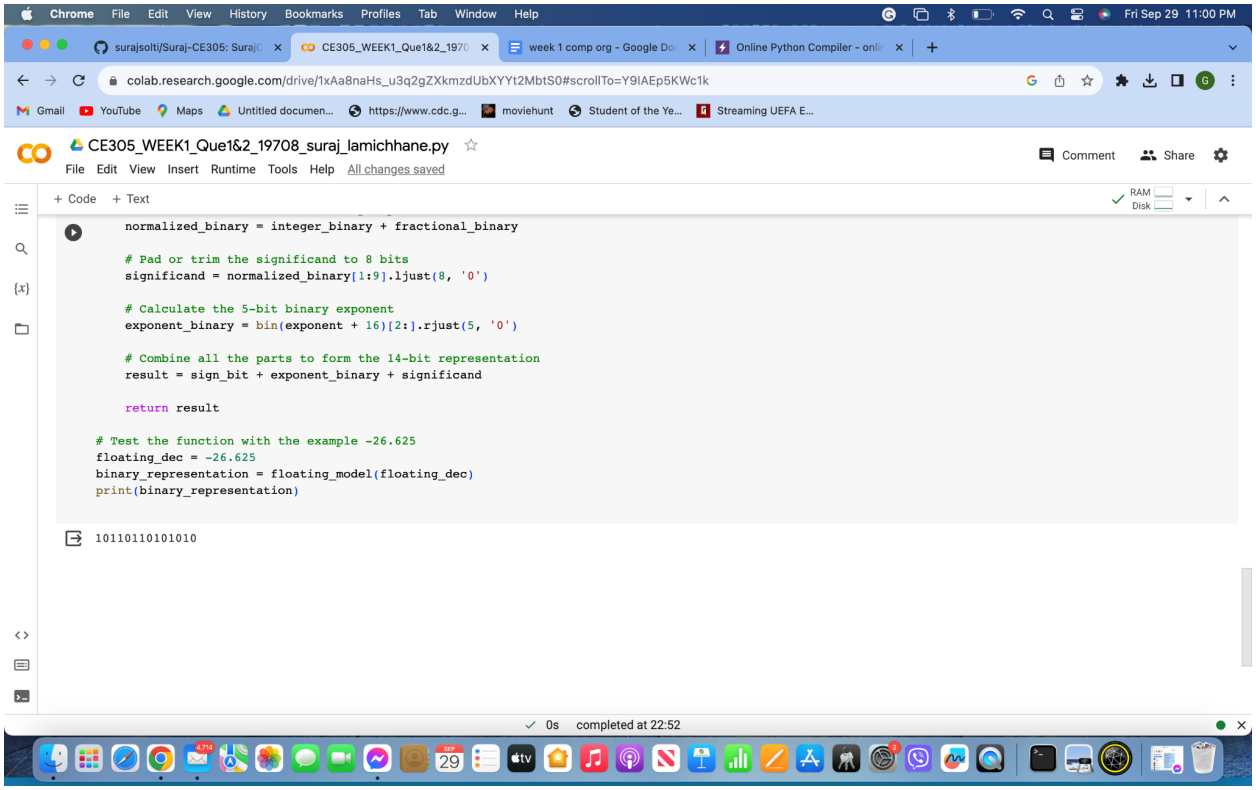


```
        decimal = to_deci(str_input, from_base)
        if decimal != -1:
            print("Decimal equivalent:", decimal)
            to_base(decimal, target_base)
    else:
        print("Error: Invalid input")


if __name__ == "__main__":
    main()
```

```
Enter the number to convert: 1234
Enter the base of the input number: 10
Enter the base to convert to: 2
Decimal equivalent: 1234
Result: 10011010010
```

Qno.2)


CE305_WEEK1_Que1&2_19708_suraj_lamichhane.py
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code
+ Text
#Question no.2
def floating_model(floating_dec):
# Check if the number is negative
if floating_dec < 0:
sign_bit = '1'
floating_dec = abs(floating_dec)
else:
sign_bit = '0'
# Extract the integer and fractional parts
integer_part = int(floating_dec)
fractional_part = floating_dec - integer_part
# Convert the integer part to binary
integer_binary = bin(integer_part)[2:]
# Initialize variables for the fractional part and exponent
fractional_binary = ""
exponent = 0
# Calculate the binary representation of the fractional part
while fractional_part != 0 and len(fractional_binary) < 8:
fractional_part *= 2
bit = int(fractional_part)
fractional_binary += str(bit)
fractional_part -= bit
exponent -= 1
Normalize the fractional binary representation
0s completed at 22:52


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Fri Sep 29 11:00 PM
colab.research.google.com/drive/1xAa8naHs_u3q2gZXkmzdUbXYYt2MbtS0#scrollTo=Y9lAEp5KWc1k
Gmail YouTube Maps Untitled documen... https://www.cdc.g... moviehunt Student of the Ye... Streaming UEFA E...
CE305_WEEK1_Que1&2_19708_suraj_lamichhane.py
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
normalized_binary = integer_binary + fractional_binary
# Pad or trim the significand to 8 bits
significand = normalized_binary[1:9].ljust(8, '0')
# Calculate the 5-bit binary exponent
exponent_binary = bin(exponent + 16)[2:].rjust(5, '0')
# Combine all the parts to form the 14-bit representation
result = sign_bit + exponent_binary + significand
return result
# Test the function with the example -26.625
floating_dec = -26.625
binary_representation = floating_model(floating_dec)
print(binary_representation)
10110110101010
0s completed at 22:52