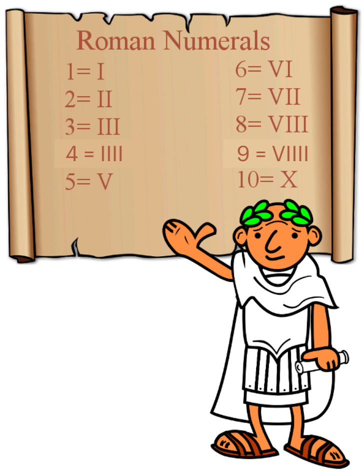
Computing module: CARM 601

Semester 1 Assessment 2

ROMAN NUMERALS



Sarina Saiyed

7 November 2023

Requirements:

1. Allow user to input two numbers in Arabic decimal form (input cannot be over 2,499)
2. Convert input number into additive roman numeral form
3. Output the converted values for the user to see
4. Allow user the choice to pick between adding the two input numbers or subtracting it
5. When chosen, display the results in both the Arabic decimal form and roman numeral form.

Measurable requirements:

1. Assign a variable for one of the input Arabic decimal number from the user
2. Assign another variable for the other Arabic decimal number from the user
3. Set a stored variable of both the roman numeral values (M, D, C, L, X, V, I) and the Arabic decimal values (1000, 500, 100, 50, 10, 5, 1)
4. Take the first assigned variable and go down the index of the stored variable of the Arabic decimal values and divide each value until the results are more than zero
5. When the result is more than zero, it will show how many of that roman value will be needed
    a. e.g ... 49 // 50 => 0, 49 // 10 => 4
    b. Therefore, will need to represent 10 + 10 + 10 + 10 => XXXX

6. To check if there are more character that need to be converted, take the first assigned variable and take the modulus operation of it with the index of the stored variable of the Arabic decimal values
    a. e.g 49% 10 => 9
7. If the given number is not in the the stored variable of the roman numeral values, repeat the process of splitting the unit values of the modulus results
    a. ... 9 // 10 => 0, 9 // 5 => 1
    b. Roman value: XXXXV
    c. 9% 1 => 1
    d. Roman value: XXXXVI
8. After repeating until the entire value is converted to additive roman numeral, repeat the steps again for the second input value
9. Output the converted input values for the user
10. Give the user the option to add or subtract the two numbers
    a. If the user chooses to add both numbers,
    b. Add the Arabic decimal form of there two input numbers and than convert to roman numeral
    c. If the user chooses to subtract from the two numbers,
    d. Subtract the Arabic decimal form of the two input numbers
    e. If the result if less than or equal to zero
        a. Tell the user this is result is not possible
        b. Set the user back to where they can choose to either add or subtract their two numbers
    f. If the result is more than zero, covert the result into roman numerals
11. Finally, display to the user the result in both Arabic decimal and Roman numeral form.

Stepwise refinement:

1. Set an input variable for the user to enter their first Arabic decimal number
2. Set another input variable for the user to enter their second Arabic decimal number
3. Define a function called RomanConverter that allow the input variables to pass through the parameters
4. If the input variables exceed 2499, return to the start of the function
5. Set a stored variable,
   i. Set one, named RomanNumerals, if data structure is a dictionary, in descending value
   ii. Set two, named RomanNumerals and ArabicDecimal, if data structure is a list, in descending value
6. Set an empty string variable called Roman
7. Set the index, I, to zero
8. Loop while the input variables are greater than zero
   i. Set a variable, called Divide, that divides the input variables with a given Arabic value that is smaller than the given input
      I) To check the range of the given Arabic decimal values, divide each of the values in the stored variable, starting with 1000
      II) Start a new integrated while loop, while Divide exceeds zero
      III) Set the variable Roman, the empty string, so that the string is adding on the the items that are allocated to the index given by Divide
      IV) To execute the while loop body again set the variable Divide to be subtracted by 1
      V) Repeat until Divide = 0
   ii. Increment the index, I, by 1

iii. This allows to got though all the values in the stored variable RomanNumerals

iv. Under the variable Divide, set the input variables to find the modulus of the variables with the Arabic decimal values

9. Return the value in the variable Roman
10. Define a new function called calculator that allow the input variables to pass through the parameters
11. Create an if statement that allows the user to choose whether they would like to add both of the input variables or to subtract both of the input variables
12. If the user chooses add
    i. Add together the integer values of the Arabic decimal numbers
    ii. Convert the answer into roman numerals using the method before
        I) Dividing the answer with each integer until the value returned is more than zero
        II) Find the modulus using the answer and the value found previously
        III) Continue until fully converted
    iii. Display the addition result in both Roman numeral and Arabic decimal form
13. If the user decided to subtract
    i) Subtract the two values from one another
    ii) If the result from the subtraction is less than or equal to zero,
    iii) Display to the user that this subtraction is not possible
        I) Return the user back to the start of the Calculator function
    iv. Else, display the subtractive results in both Roman numeral and Arabic decimal form.

Pseudocode:

```
     DEF RomanConverter1(DecimalInt1, ArabicDecimal,
RomanNumerals)
     IF DecimalInt1 > 2499:
               PRINT("Your input numbers are too big")
               RETURN
     Roman1 = ""
     i = 0
     WHILE DecimalInt1  > 0:
               Divide1 = DecimalInt1 // ArabicDecimal[i]
               DecimalInt1 = DecimalInt1 % ArabicDecimal[i]
               WHILE Divide1:
                    Roman1 = Roman1 + RomanNumerals[i]
                    Divide1 -=1
               END WHILE
               i +=1
     END WHILE
     RETURN Roman1
     END IF


     DEF RomanConverter2(DecimalInt2, ArabicDecimal,
     RomanNumerals)
     IF DecimalInt2 > 2499:
               PRINT("Your second input numbers are too big")
               RETURN
     Roman2 = ""
     i = 0
     WHILE DecimalInt2  > 0:
```

```
                Divide2 = DecimalInt2 // ArabicDecimal[i]
                DecimalInt2 = DecimalInt2 % ArabicDecimal[i]
                WHILE Divide2:
                        Roman2 = Roman2 + RomanNumerals[i]
                        Divide2 -=1
                END WHILE
                i +=1
        END WHILE
        RETURN Roman2
        END IF

        DEF
calculator(DecimalInt1,DecimalInt2,ArabicDecimal,RomanNumerals):
        option = INPUT("Would you like to Add or Subtract your numbers?
\n")
        IF option == "Add":
                RomanResultA = ""
                i = 0
                resultA = DecimalInt1 + DecimalInt2
                divideA = resultA // ArabicDecimal[i]
                resultA = resultA % ArabicDecimal[i]
                WHILE divideA:
                        RomanResultA = RomanResultA +
RomanNumerals[i]
                        divideA -=1
                END WHILE
                i += 1
             PRINT(resultA, RomanResultA)
        ELIF option == "Subtract":
                RomanResultS = ""
                i = 0
                resultS = DecimalInt1 + DecimalInt2
                IF results <= 0:
                    PRINT("Answer cannot be less than or equal to zero")
                        RETURN
                divideS = resultS // ArabicDecimal[i]
                resultS = resultS % ArabicDecimal[i]
                WHILE divideS:
                        RomanResultS = RomanResultS +
RomanNumerals[i]
                        divideS -=1
                END WHILE
                i += 1
            PRINT(resultS, RomanResultS)
    END IF


    ArabicDecimal = [1000, 500, 100, 50, 10, 5, 1]
    RomanNumerals = ["M", "D", "C", "L", "X", "V", "I"]

    PRINT("The numbers you are about to enter cannot exceed 2499.")
    DecimalInt1 = INT(INPUT("Enter your first number: "))
    DecimalInt2 = INT(INPUT("Enter your second number: "))
```

Code body:

```
# program: ROMAN NUMERALS.py
# author: Sarina Saiyed
# email: 2338323@students.carmel.ac.uk
# student number: 2338323
#
# A manufacturing company wishes to make a new calculator that allows
# the user to input two numbers in Arabic decimal form (each input
# number must not exceed 2499) and displays the numbers in Roman numeral
# form.  The calculator then allows a user to add or subtract the two
# numbers, displaying the result in both Arabic decimal and Roman
# numeral form.


##################################################
# Design

# Assign a variable for one of the input Arabic decimal number from the user
# Assign another variable for the other  Arabic decimal number from the user
# Set a stored variable of both the roman numeral values (M, D, C, L, X, V, I)
#   and the Arabic decimal values (1000, 500, 100, 50, 10, 5, 1)
# Take the first assigned variable and go down the index of the stored
#   variable of the Arabic decimal values and divide each value until the
#   results are more than zero
# When the result is more than zero, it will show how many of that roman
#   value will be needed
#       e.g … 49 // 50 => 0, 49 // 10 => 4
# Therefore, will need to represent 10 + 10 + 10 + 10 => XXXX
# To check if there are more character that need to be converted, take
#   the first assigned variable and take the modulus operation of it
#   with the index of the stored variable of the Arabic decimal values
#       e.g 49 % 10 => 9
# If the given number is not in the the stored variable of the roman
#   numeral values, repeat the process of splitting the unit values
#   of the modulus results
#       … 9 // 10 => 0, 9 // 5 => 1
# Roman value: XXXXV
# 9 % 1 => 1
# Roman value: XXXXVI
# After repeating until the entire value is converted to additive roman
#   numeral, repeat the steps again for the second input value
# Output the converted input values for the user
# Give the user the option to add or subtract the two numbers
# If the user chooses to add both numbers,
# Add the Arabic decimal form of there two input numbers and than convert
#   to roman numeral
# If the user chooses to subtract from the two numbers,
# Subtract the Arabic decimal form of the two input numbers
# If the result if less than or equal to zero
# Tell the user this is result is not possible
# Set the user back to where they can choose to either add or subtract
#   their two numbers
# If the result is more than zero, covert the result into roman numerals
# Finally, display to the user the result in both Arabic decimal and
#   Roman numeral form.


#################################################
# Pseudocode

# DEF RomanConverter1(DecimalInt1, ArabicDecimal, RomanNumerals)
#       IF DecimalInt1 > 2499:
#               PRINT("Your input numbers are too big")
#               RETURN
#
#       Roman1 = ""
#       i = 0
#       WHILE DecimalInt1  > 0:
#               Divide1 = DecimalInt1 // ArabicDecimal[i]
#               DecimalInt1 = DecimalInt1 % ArabicDecimal[i]
#               WHILE Divide1:
#                       Roman1 = Roman1 + RomanNumerals[i]
#                       Divide1 -=1
#               END WHILE
#               i +=1
#       END WHILE
#       RETURN Roman1
#       END IF

#
# DEF RomanConverter2(DecimalInt2,ArabicDecimal, RomanNumerals)
#       IF DecimalInt2 > 2499:
#               PRINT("Your second input numbers are too big")
#               RETURN]

#       Roman2 = ""
#       i = 0
#       WHILE DecimalInt2  > 0:
#               Divide2 = DecimalInt2 // ArabicDecimal[i]
#               DecimalInt2 = DecimalInt2 % ArabicDecimal[i]
#               WHILE Divide2:
#                       Roman2 = Roman2 + RomanNumerals[i]
#                       Divide2 -=1
#               END WHILE
#               i +=1
#       END WHILE
#       RETURN Roman2
#       END IF
#
# DEF calculator(DecimalInt1,DecimalInt2,ArabicDecimal,RomanNumerals):
#       option = INPUT("Would you like to Add or Subtract your numbers? \n")
#       IF option == "Add":
#               RomanResultA = ""
#               i = 0
#               resultA = DecimalInt1 + DecimalInt2
#               divideA = resultA // ArabicDecimal[i]
#               resultA = resultA % ArabicDecimal[i]
#               WHILE divideA:
#                       RomanResultA = RomanResultA + RomanNumerals[i]
#                       divideA -=1
#               END WHILE
#               i += 1
#               PRINT(resultA, RomanResultA)
#       ELIF option == "Subtract":
```

```
#               RomanResultS = ""
#               i = 0
#               resultS = DecimalInt1 + DecimalInt2
#               IF results <= 0:
#                       PRINT("Answer cannot be less than or equal to zero, choose
addtion")
#                       RETURN
#               divideS = resultS // ArabicDecimal[i]
#               resultS = resultS % ArabicDecimal[i]
#               WHILE divideS:
#                       RomanResultS = RomanResultS + RomanNumerals[i]
#                       divideS -=1
#               END WHILE
#               i += 1
#       PRINT(resultS, RomanResultS)
#       END IF
#
#
# ArabicDecimal = [1000, 500, 100, 50, 10, 5, 1]
# RomanNumerals = ["M", "D", "C", "L", "X", "V", "I"]
#
# PRINT("The numbers you are about to enter cannot exceed 2499.")
# DecimalInt1 = INT(INPUT("Enter your first number: "))
# DecimalInt2 = INT(INPUT("Enter your second number: "))
#


##################################################
# Variables

# INT DecimalInt1
# INT DecimalInt2
# INT ArabicDecimal
# STR RomanNumerals
# STR Roman1
# STR Roman2
# INT Divide1
# INT Divide2
# STR option
# STR RomanResultA
# STR RomanResultS
# INT resultA
# INT resultS
# INT divideA
# INT divideS


##################################################
# Functions

# RomanConverter1
# RomanConverter2
# addcalculator
# subcalculator

##################################################
# Main


def RomanConverter1(DecimalInt1):
    if DecimalInt1 > 2499: # if statement that ends the funtion
        print("Your first input number is too big:")# if the first value is greater than
2499
        return
    Roman1 = "" #empty string variable, store final result
    i = 0 # index is set to zero
    while DecimalInt1 > 0: #while loop that continues when the variable
'Decimal1' is more than 0
        Divide1 = DecimalInt1 // ArabicDecimal[i] # the variable is divided with
the value of each index in the stored variable
        DecimalInt1 = DecimalInt1 % ArabicDecimal[i]# the answer from before is
used to find the modulus using the same stored variable
                                                  # This will continue untill the variable is 0
        while Divide1: # while the variable is not zero, will execute body untill
Divide1 = 0
            Roman1 = Roman1 + RomanNumerals[i] # the empty string is added
onto using the index of the roman stored variable
```

```
                Divide1 -= 1 # takes the value away from 1; so when variable = 0 the
body will end
            i += 1 # increments the index of the stored variables to indicate that the
largest value as been checked and will be going to the next largest value untill the
end
        return Roman1 # ends the program and allows to return the final converted
input

    def RomanConverter2(DecimalInt2): # same code as the first function
        if DecimalInt2 > 2499:
            print("Your second input number is too big: ")
            return
        Roman2 = ""
        i = 0
        while DecimalInt2 > 0:
            Divide2 = DecimalInt2 // ArabicDecimal[i]
            DecimalInt2 = DecimalInt2 % ArabicDecimal[i]
            while Divide2:
                Roman2 = Roman2 + RomanNumerals[i]
                Divide2 -= 1
            i += 1
        return Roman2


    def addcalculator(DecimalInt1,DecimalInt2):
        RomanResultA = ""
        i = 0
        resultA = DecimalInt1 + DecimalInt2 # gives the Arabic decimal addtion value
        while resultA > 0: # converts the results into roman numerals like in the
convertion functions
            divideA = resultA // ArabicDecimal[i]
            resultA = resultA % ArabicDecimal[i]
            while divideA:
                RomanResultA = RomanResultA + RomanNumerals[i]
                divideA -=1
            i += 1
        resultA = DecimalInt1 + DecimalInt2 #repeating this line of code to be able
to output both values
```

```
    return RomanResultA, resultA # returns both the arabic decimal and roman numeral form


def subcalculator(DecimalInt1,DecimalInt2):
    RomanResult = ""
    i = 0
    resultS = DecimalInt1 - DecimalInt2 # gives the subtraction
    if resultS <= 0: # ends the function if the result is zero or negative
        print("Answer cannot be less than or equal to zero")
    return
    while resultS > 0: # converts answer to roman numeral
        divideS = resultS // ArabicDecimal[i]
        resultS = resultS % ArabicDecimal[i]
        while divideS:
            RomanResult = RomanResult + RomanNumerals[i]
            divideS -=1
        i += 1
    resultS = DecimalInt1 - DecimalInt2
    return RomanResult, resultS



#Varibles are below the functions so that they are able to be passed through the parameters or used as arguments

ArabicDecimal = [1000, 500, 100, 50, 10, 5, 1] #stored Arabic decimal variable
RomanNumerals = ["M", "D", "C", "L", "X", "V", "I"] #stored Roman numeral variable


print("The numbers you are about to enter cannot exceed 2499.") # tells user the input limit

DecimalInt1 = int(input("Enter your first number: ")) #input variable
DecimalInt2 = int(input("Enter your second number: ")) #input variable

print("Input one:",RomanConverter1(DecimalInt1)) # prints out the roman convertion of first input
```

```
    print("Input two:",RomanConverter2(DecimalInt2)) # prints out the roman
convertion of second input


    if DecimalInt1 <= 2499 and DecimalInt2 <= 2499: # if statement to allow user
access to addtion and subtraction if both inputs are less than 2499
        option = input("Would you like to add or subtract? \n")
        if option == "add":
            addcalculator(DecimalInt1 ,DecimalInt2) #send the user to the
addcalculator funtion
            print(addcalculator(DecimalInt1 ,DecimalInt2)) # prints out the return
values after the variables have fully passed the function
        elif option == "subtract":
            subcalculator(DecimalInt1,DecimalInt2)
            print(subcalculator(DecimalInt1,DecimalInt2))
        else:
            print("Please enter either 'add' or 'subtract'")
            quit #end the program
```

Code without annotations:

```
def RomanConverter1(DecimalInt1):
    if DecimalInt1 > 2499:
        print("Your first input number is too big:")
    Roman1 = ""
    i = 0
    while DecimalInt1 > 0:
        Divide1 = DecimalInt1 // ArabicDecimal[i]
        DecimalInt1 = DecimalInt1 % ArabicDecimal[i]
        while Divide1:
            Roman1 = Roman1 + RomanNumerals[i]
            Divide1 -= 1
        i += 1
    return Roman1

def RomanConverter2(DecimalInt2):
    if DecimalInt2 > 2499:
        print("Your second input number is too big: ")
        return
    Roman2 = ""
    i = 0
    while DecimalInt2 > 0:
        Divide2 = DecimalInt2 // ArabicDecimal[i]
        DecimalInt2 = DecimalInt2 % ArabicDecimal[i]
        while Divide2:
            Roman2 = Roman2 + RomanNumerals[i]
            Divide2 -= 1
        i += 1
    return Roman2


def addcalculator(DecimalInt1,DecimalInt2):
    RomanResultA = ""
    i = 0
    resultA = DecimalInt1 + DecimalInt2 # gives the Arabic decimal addtion value
    while resultA > 0:
        divideA = resultA // ArabicDecimal[i]
        resultA = resultA % ArabicDecimal[i]
        while divideA:
            RomanResultA = RomanResultA + RomanNumerals[i]
            divideA -=1
        i += 1
    resultA = DecimalInt1 + DecimalInt2
    return RomanResultA, resultA


def subcalculator(DecimalInt1,DecimalInt2):
    RomanResult = ""
    i = 0
    resultS = DecimalInt1 - DecimalInt2
    if resultS <= 0:
        print("Answer cannot be less than or equal to zero")
    return
    while resultS > 0:
        divideS = resultS // ArabicDecimal[i]
        resultS = resultS % ArabicDecimal[i]
        while divideS:
            RomanResult = RomanResult + RomanNumerals[i]
            divideS -=1
        i += 1
    resultS = DecimalInt1 - DecimalInt2
    return RomanResult, resultS



ArabicDecimal = [1000, 500, 100, 50, 10, 5, 1]
RomanNumerals = ["M", "D", "C", "L", "X", "V", "I"]


print("The numbers you are about to enter cannot exceed 2499.")

DecimalInt1 = int(input("Enter your first number: "))
DecimalInt2 = int(input("Enter your second number: "))

print("Input one:",RomanConverter1(DecimalInt1))
print("Input two:",RomanConverter2(DecimalInt2))


if DecimalInt1 <= 2499 and DecimalInt2 <= 2499:
    option = input("Would you like to add or subtract? \n")
    if option == "add":
        addcalculator(DecimalInt1 ,DecimalInt2)
        print(addcalculator(DecimalInt1 ,DecimalInt2))
    elif option == "subtract":
        subcalculator(DecimalInt1,DecimalInt2)
        print(subcalculator(DecimalInt1,DecimalInt2))
    else:
        print("Please enter either 'add' or 'subtract'")
        quit
```

Testing table:

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome

1	Checking if the user enters an input value that is higher than 2499	`if DecimalInt1 > 2499:`
		
		`print("Your first input number is too big: ")`
		`        return`
			“None”	Nothing

2	Checking if the user enters an input value that is higher than 2499	`if DecimalInt1 > 2499:`
		`        print("Your first input number is too big: ")`
		`        return`
		
		`print(RomanConverter1(DecimalInt1))`
			“None”	“None”

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome

3	Testing the conversion of the first Arabic decimal input	`Roman1 = ""`
		`    i = 0`
		`    while DecimalInt1 > 0:`
		`        Divide1 = DecimalInt1 // ArabicDecimal[i]`
		`        DecimalInt1 = DecimalInt1 % ArabicDecimal[i]`
		`        while Divide1:`
		`            Roman1 = Roman1 = RomanNumerals[i]`
		`            Divide1 -= 1`
		`        i += 1`
		`    return Roman1`
		
		`ArabicDecimal = [1000, 500, 100, 50, 10, 5, 1]`
		`RomanNumerals = ["M", "D", "C", "L", "X", "V", "I"]`
		
		`print("The numbers you are about to enter cannot exceed 2499.")`
		`DecimalInt1 = int(input("Enter your first number: "))`
		`#DecimalInt2 = int(input("Enter your second number: "))`
		
		`print(RomanConverter1(DecimalInt1))`
			Input: 5
			
			Output: V
				V

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome

4	Testing the conversion of the first Arabic decimal input	Roman1 = ""
		    i = 0
		    while DecimalInt1 > 0:
		        Divide1 = DecimalInt1 // ArabicDecimal[i]
		        DecimalInt1 = DecimalInt1 % ArabicDecimal[i]
		        while Divide1:
		            Roman1 = Roman1 = RomanNumerals[i]
		            Divide1 -= 1
		        i += 1
		    return Roman1
		
		ArabicDecimal = [1000, 500, 100, 50, 10, 5, 1]
		RomanNumerals = ["M", "D", "C", "L", "X", "V", "I"]
		
		print("The numbers you are about to enter cannot exceed 2499.")
		DecimalInt1 = int(input("Enter your first number: "))
		#DecimalInt2 = int(input("Enter your second number: "))
		
		print(RomanConverter1(DecimalInt1))
			Input: 60
			
			Output: LX
				X

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome

5	Testing the conversion of the first Arabic decimal input	`Roman1 = ""`
		`    i = 0`
		`    while DecimalInt1 > 0:`
		`        Divide1 = DecimalInt1 // ArabicDecimal[i]`
		`        DecimalInt1 = DecimalInt1 % ArabicDecimal[i]`
		`        while Divide1:`
		`            Roman1 = Roman1 + RomanNumerals[i]`
		`            Divide1 -= 1`
		`        i += 1`
		`    return Roman1`
		
		`ArabicDecimal = [1000, 500, 100, 50, 10, 5, 1]`
		`RomanNumerals = ["M", "D", "C", "L", "X", "V", "I"]`
		
		`print("The numbers you are about to enter cannot exceed 2499.")`
		`DecimalInt1 = int(input("Enter your first number: "))`
		`#DecimalInt2 = int(input("Enter your second number: "))`
		
		`print(RomanConverter1(DecimalInt1))`
			Input: 60
			
			Output: LX
				LX

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome

6	Testing to see if the second conversion function works with the first one	def RomanConverter2(DecimalInt2):
		    if DecimalInt2 > 2499:
		        print("Your first input number is too big: ")
		        return
		    Roman2 = ""
		    i = 0
		    while DecimalInt2 > 0:
		        Divide2 = DecimalInt2 // ArabicDecimal[i]
		        DecimalInt2 = DecimalInt2 % ArabicDecimal[i]
		
		        while Divide2:
		
		            Roman2 = Roman2 + RomanNumerals[i]
		            Divide2 -= 1
		        i += 1
		    return Roman2
		
		ArabicDecimal = [1000, 500, 100, 50, 10, 5, 1]
		RomanNumerals = ["M", "D", "C", "L", "X", "V", "I"]
		
		print("The numbers you are about to enter cannot exceed 2499.")
		DecimalInt1 = int(input("Enter your first number: "))
		DecimalInt2 = int(input("Enter your second number: "))
		
		print(RomanConverter1(DecimalInt1))
		print(RomanConverter2(DecimalInt2))
			Input 1: 5 or 60
			Input 2: 10 or 40
			
			Output: V or LX
			
			Output: X or XXXX
				V
				LX
				
				Or
				
				X
				XXXX

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome

7	Testing the adding option in the calculator function	option = input("Would you like to add or subtract? \n")
		    if option == "add":
		        RomanResult = ""
		        i = 0
		        result = DecimalInt1 + DecimalInt2
		        while result > 0:
		            divideA = resultA // ArabicDecimal[i]
		            resultA = resultA % ArabicDecimal[i]
		            while divideA:
		                RomanResult = RomanResult + RomanNumerals[i]
		                divideA -=1
		            i += 1
			Input: 10, 40
			Add
			
			Output: 50, L
				Error

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome

8	Testing the adding option in the calculator function	`option = input("Would you like to add or subtract? \n")`
		`    if option == "add":`
		`        RomanResult = ""`
		`        i = 0`
		`        result = DecimalInt1 + DecimalInt2`
		`        while result > 0:`
		`            divideA = result // ArabicDecimal[i]`
		`            result = result % ArabicDecimal[i]`
		`            while divideA:`
		`                RomanResult = RomanResult + RomanNumerals[i]`
		`                divideA -=1`
		`            i += 1`
			Input: 10, 40
			Add
			
			Output:
			50, L
				0

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome

9	Testing new way for the adding option	`def addcalculator(DecimalInt1,DecimalInt2):`
		    `RomanResult = ""`
		    `i = 0`
		    `result = DecimalInt1 + DecimalInt2`
		    `return result`
		    `while result > 0:`
		        `divideA = result // ArabicDecimal[i]`
		        `result = result % ArabicDecimal[i]`
		        `while divideA:`
		            `RomanResult = RomanResult + RomanNumerals[i]`
		            `divideA -=1`
		        `i += 1`
		    `return RomanResult`
		
		`...`
		`print(addcalculator(DecimalInt1 ,DecimalInt2))`
		
		`option = input("Would you like to add or subtract? \n")`
		`if option == "add":`
		    `addcalculator()`
		`elif option == "subract":`
		    `subcalculator()`
		`else:`
		    `print("nope")`
			Input: 10, 40
			Add
			
			Output: 50, L
				Enter your first number: 10
				Enter your second number: 40
				X
				XXXX
				L
				Would you like to add or subtract?
				add
				ERROR

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome

10	Testing new way for the adding option	def addcalculator(DecimalInt1,DecimalInt2):
		    RomanResult = ""
		    i = 0
		    result = DecimalInt1 + DecimalInt2
		    return result
		    while result > 0:
		        divideA = result // ArabicDecimal[i]
		        result = result % ArabicDecimal[i]
		        while divideA:
		            RomanResult = RomanResult + RomanNumerals[i]
		            divideA -=1
		        i += 1
		    return RomanResult
		
		...
		option = input("Would you like to add or subtract? \n")
		if option == "add":
		    addcalculator()
		elif option == "subract":
		    subcalculator()
		else:
		    print("nope")
		
		print(addcalculator(DecimalInt1 ,DecimalInt2))
			Input: 10, 40
			Add
			
			Output: 50, L
				Enter your first number: 10
				Enter your second number: 40
				X
				XXXX
				Would you like to add or subtract?
				add
				ERROR

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome

11	Testing new way for the adding option	def addcalculator(DecimalInt1,DecimalInt2):
		    RomanResult = ""
		    i = 0
		    result = DecimalInt1 + DecimalInt2
		    return result
		    while result > 0:
		        divideA = result // ArabicDecimal[i]
		        result = result % ArabicDecimal[i]
		        while divideA:
		            RomanResult = RomanResult + RomanNumerals[i]
		            divideA -=1
		        i += 1
		    return RomanResult
		…
		option = input("Would you like to add or subtract? \n")
		if option == "add":
		
		addcalculator(DecimalInt1 ,DecimalInt2)
		elif option == "subract":
		    subcalculator()
		else:
		    print("nope")
		
		print(addcalculator(DecimalInt1 ,DecimalInt2))
			Input: 10, 40
			Add
			
			Output: 50, L
				Enter your first number: 10
				Enter your second number: 40
				X
				XXXX
				Would you like to add or subtract?
				add
				50

12	Testing new way for the adding option	def addcalculator(DecimalInt1,DecimalInt2):
		    RomanResult = ""
		    i = 0
		    result = DecimalInt1 + DecimalInt2
		    #return result
		    while result > 0:
		        divideA = result // ArabicDecimal[i]
		        result = result % ArabicDecimal[i]
		        while divideA:
		            RomanResult = RomanResult + RomanNumerals[i]
		            divideA -=1
		        i += 1
		    return RomanResult
			Input: 10, 40
			Add
			
			Output: 50, L
				Enter your first number: 10
				Enter your second number: 40
				X
				XXXX
				Would you like to add or subtract?
				add
				L

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome

13	Testing to get both the Arabic decimal value and the roman numeral value	def addcalculator(DecimalInt1,DecimalInt2):
		    RomanResult = ""
		    i = 0
		    result = DecimalInt1 + DecimalInt2
		    #return result
		    while result > 0:
		        divideA = result // ArabicDecimal[i]
		        result = result % ArabicDecimal[i]
		        while divideA:
		            RomanResult = RomanResult + RomanNumerals[i]
		            divideA -=1
		        i += 1
		    return RomanResult, result
			Input: 10, 40
			Add
			
			Output: 50, L
				Enter your first number: 10
				Enter your second number: 40
				X
				XXXX
				Would you like to add or subtract?
				add
				(‘L’, 0)

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome

14	Testing to get both the Arabic decimal value and the roman numeral value	`def addcalculator(DecimalInt1,DecimalInt2):`
		`    RomanResultA = ""`
		`    i = 0`
		`    resultA = DecimalInt1 + DecimalInt2`
		`    while resultA > 0:`
		`        divideA = resultA // ArabicDecimal[i]`
		`        resultA = resultA % ArabicDecimal[i]`
		`        while divideA:`
		`            RomanResultA = RomanResultA + RomanNumerals[i]`
		`            divideA -=1`
		`        i += 1`
		`    resultA = DecimalInt1 + DecimalInt2`
		`    return resultA, RomanResultA`
			Input: 10, 40
			Add
			
			Output: 50, L
				Enter your first number: 10
				Enter your second number: 40
				X
				XXXX
				Would you like to add or subtract?
				add
				(‘L’, 50)

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome

15	Testing out the sub calculator function with what I have figured out with the add calculator function	def subcalculator(DecimalInt1,DecimalInt2):
		    RomanResult = ""
		    i = 0
		    resultS = DecimalInt1 - DecimalInt2
		    if resultS <= 0:
		        print("Answer cannot be less than or equal to zero")
		        return
		    while resultS > 0:
		        divideS = resultS // ArabicDecimal[i]
		        resultS = resultS % ArabicDecimal[i]
		        while divideS:
		            RomanResult = RomanResult + RomanNumerals[i]
		            divideS -=1
		        i += 1
		    resultS = DecimalInt1 - DecimalInt2
		    return RomanResult, resultS
		
		option = input("Would you like to add or subtract? \n")
		if option == "add":
		
		addcalculator(DecimalInt1 ,DecimalInt2)
		
		print(addcalculator(DecimalInt1 ,DecimalInt2))
		elif option == "subtract":
		
		subcalculator(DecimalInt1,DecimalInt2)
		
		print(subcalculator(DecimalInt1,DecimalInt2))
		else:
		    print("nope")
			Input: 40, 10
			Subtract
			
			Output:
			(‘XXX’, 30)
				(‘XXX’, 30)

16	Testing the input of 10 and 40 and adding the two numbers together	Input: 10 Input: 40 Input: add	X XXXX (‘L’,50)	The numbers you are about to enter cannot exceed 2499. Enter your first number: 10 Enter your second number: 40 X XXXX Would you like to add or subtract? add ('L', 50)

Test Num ber	Description of the test	Test Data	Expected Outcome	Actual outcome

17	Testing when one of the inputs in larger than 2499 and when the other is the correct input
	
	(With addition)
		`Input: 3000`
		`Input: 40`
			Your first input number is too big:
			
			None
				The numbers you are about to enter cannot exceed 2499.
				Enter your first number: 3000
				Enter your second number: 40
				Your first input number is too big:
				None
				XXXX
				Would you like to add or subtract?
				add
				('MMMXXXX', 3040)

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome

18	Testing when one of the inputs in larger than 2499 and when the other is the correct input	Input: 3000
		Input: 40
		
		if (DecimalInt1 or DecimalInt2) < 2499:
		    option = input("Would you like to add or subtract? \n")
		    if option == "add":
		        addcalculator(DecimalInt1 ,DecimalInt2)
		        print(addcalculator(DecimalInt1 ,DecimalInt2))
		    elif option == "subtract":
		        subcalculator(DecimalInt1,DecimalInt2)
		        print(subcalculator(DecimalInt1,DecimalInt2))
		    else:
		        print("Please enter either 'add' or 'subtract'")
		        quit
			Your first input number is too big:
			
			None
			XXXX
				The numbers you are about to enter cannot exceed 2499.
				Enter your first number: 3000
				Enter your second number: 40
				Your first input number is too big:
				Input one: None
				Input two: XXXX

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome
19	Testing when the second inputs in larger than 2499 and when the first one is the correct input	Input: 40 Input: 3000	Your second number is too big: XXXX None	The numbers you are about to enter cannot exceed 2499. Enter your first number: 40 Enter your second number: 3000 Input one: XXXX Your first input number is too big: Input two: None Would you like to add or subtract?

20	Testing when the second inputs in larger than 2499 and when the first one is the correct input	Input: 40
		Input: 3000
		
		`if DecimalInt1 < 2499 and DecimalInt2 < 2499:`
		`    option = input("Would you like to add or subtract? \n")`
		`    if option == "add":`
		`        addcalculator(DecimalInt1 ,DecimalInt2)`
		`        print(addcalculator(DecimalInt1 ,DecimalInt2))`
		`    elif option == "subtract":`
		`        subcalculator(DecimalInt1,DecimalInt2)`
		`        print(subcalculator(DecimalInt1,DecimalInt2))`
		`    else:`
		`        print("Please enter either 'add' or 'subtract'")`
		`        quit`
			Your second number is too big:
			
			XXXX
			None
				The numbers you are about to enter cannot exceed 2499.
				Enter your first number: 40
				Enter your second number: 3000
				Input one: XXXX
				Your second input number is too big:
				Input two: None

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome
21	Inputting values larger than 2499 in both inputs	Input: 3000 input: 3000	Your first input is too big: None Your second input is too big: None	The numbers you are about to enter cannot exceed 2499. Enter your first number: 3000 Enter your second number: 3000 Your first input number is too big: Input one: None Your second input number is too big: Input two: None
22	Testing 2499 in first input	Input: 2499 Input: 10 Input: add	MMCCCCLXXXXVIIII X ("MMDVIIII", 2509)	The numbers you are about to enter cannot exceed 2499. Enter your first number: 2499 Enter your second number: 10 Input one: MMCCCCLXXXXVIIII Input two: X

	Testing 2499 in first input	Input: 2499
		Input: 10
		
		Input: add
		
		`if DecimalInt1 <= 2499 and DecimalInt2 <= 2499:`
		`    option = input("Would you like to add or subtract? \n")`
		`    if option == "add":`
		`        addcalculator(DecimalInt1 ,DecimalInt2)`
		`        print(addcalculator(DecimalInt1 ,DecimalInt2))`
		`    elif option == "subtract":`
		`        subcalculator(DecimalInt1,DecimalInt2)`
		`        print(subcalculator(DecimalInt1,DecimalInt2))`
		`    else:`
		`        print("Please enter either 'add' or 'subtract'")`
		`        quit`
			MMCCCCLXXXXVIIII
			
			X
			("MMDVIIII", 2509)
				The numbers you are about to enter cannot exceed 2499.
				Enter your first number: 2499
				Enter your second number: 10
				Input one:
				MMCCCCLXXXXVIIII
				Input two: X
				Would you like to add or subtract?
				add
				('MMDVIIII', 2509)

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome
23	Testing 2500 in first input	Input: 2500 Input: 10	Your first input number is too big: None X	The numbers you are about to enter cannot exceed 2499. Enter your first number: 2500 Enter your second number: 10 Your first input number is too big: Input one: None Input two: X
24	Testing 2499 in second input	Input: 10 Input: 2499 Input: add	X MMCCCCLXXXXVIIII ("MMDVIIII", 2509)	The numbers you are about to enter cannot exceed 2499. Enter your first number: 10 Enter your second number: 2499 Input one: X Input two: MMCCCCLXXXXVIIII Would you like to add or subtract? add ('MMDVIIII', 2509)
25	Testing 2500 in second input	Input: 10 Input: 2500	X Your second input number is too big: None	The numbers you are about to enter cannot exceed 2499. Enter your first number: 10 Enter your second number: 2500 Input one: X Your second input number is too big: Input two: None

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome
25.5	Testing add two of the same values	Input: 10 Input: 10 Input: add	X X (“XX”, 20)	The numbers you are about to enter cannot exceed 2499. Enter your first number: 10 Enter your second number: 10 Input one: X Input two: X Would you like to add or subtract? add ('XX', 20)
26	Testing the subtraction function when the first value is bigger than the second (Answer > 0)	Input: 40 Input: 10 Input: subtraction	XXXX X (“XXX”, 30)	The numbers you are about to enter cannot exceed 2499. Enter your first number: 40 Enter your second number: 10 Input one: XXXX Input two: X Would you like to add or subtract? subtract ('XXX', 30)

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome
27	Testing the subtraction function when the first value is 2499 (testing limit)	Input: 2499 Input: 10 Input: subtraction	MMCCCCLXXXXVIIII X ('MMCCCCLXXXVIIII', 2489)	The numbers you are about to enter cannot exceed 2499. Enter your first number: 2499 Enter your second number: 10 Input one: MMCCCCLXXXXVIIII Input two: X Would you like to add or subtract? subtract ('MMCCCCLXXXVIIII', 2489)
28	Testing when the first value is 2500	Input: 2500 Input: 10 Input: subtraction	Your first input is too big: None X	The numbers you are about to enter cannot exceed 2499. Enter your first number: 2500 Enter your second number: 10 Your first input number is too big: Input one: None Input two: X

Test Number	Description of the test	Test Data	Expected Outcome	Actual outcome
29	Testing when the second number is bigger than the first when subtracting	Input: 10 Input: 40 Input: subtraction	XXXX X Answer cannot be less than or equal to zero	The numbers you are about to enter cannot exceed 2499. Enter your first number: 10 Enter your second number: 40 Input one: X Input two: XXXX Would you like to add or subtract? subtract Answer cannot be less than or equal to zero Answer cannot be less than or equal to zero None
30	Testing subtraction two of the same inputs	Input: 10 Input: 10 Input: subtract	X X Answer cannot be less than or equal to zero	The numbers you are about to enter cannot exceed 2499. Enter your first number: 10 Enter your second number: 10 Input one: X Input two: X Would you like to add or subtract? subtract Answer cannot be less than or equal to zero Answer cannot be less than or equal to zero None

References:

Arabic decimal form to Roman Numerals

Returning multiple values in a function