Recitation 05b - Functions

Exercise 1 - (Gaddis 5.17) Prime Numbers

A prime number is a number that is only evenly divisible by itself and 1. For example, the number 5 is prime because it can only be evenly divided by 1 and 5. The number 6, however, is not prime because it can be divided evenly by 1, 2, 3, and 6.

Write a Boolean function named `is_prime` which takes an integer as an argument and returns `True` if the argument is a prime number, or `False` otherwise. Use the function in a program that prompts the user to enter a number and then displays a message indicating whether the number is prime.

Exercise 2 - (Gaddis 5.18) Prime Number List

This exercise assumes that you have already written the `is_prime` function in Exercise 6. Write another program that calls the `is_prime` function in a loop to display all prime numbers between 1 to 100.

Exercise 3 - Days in a Month

Write a function named `Days_in_a_month`that determines how many days there are in a particular month. Your function will take two parameters: The month as an integer between 1 and 12, and the year as a four digit integer. Ensure that your function reports the correct number of days in February for leap years. Include a main program that reads a month and year from the user and displays the number of days in that month. You may find useful to write a function named `Leap_year` that determines whether the year is a leap year.

Exercise 4 - Rotate

Write a function named `rotate` that takes as its input a string of lower case letters, a-z, and spaces, and an integer n, and returns the string constructed by shifting each of the letters n steps, and leaving the spaces unchanged. Note that ’a’ follows ’z’. You can use an additional procedure if you choose to as long as rotate returns the correct string.
Note that n can be positive, negative or zero.
Note: use ord() and chr() functions.

You can use the parameters below to test your code
# string = ”zw pfli tfuv nfibj tfiivtkcp pfl jyflcu sv rscv kf ivru kyzj”
# shift = -17

Exercise 5 - Reduce a Fraction to Lowest Terms

Write a function named `reduce_fraction` that takes two positive integers that represent the numerator and denominator of a fraction as its only two parameters. This should call another function named `greastest_common_divisor` and use the returned value to reduce the fraction to its lowest terms and then return both the numerator and denominator of the reduced fraction as its result. The main program should ask the user to enter a numerator and denominator. Then your program should display the reduced fraction. You can assume the user enters valid integers.