Check if a given number is positive or negative:

```
num = float(input("Enter a number: "))
if num > 0:
   print("The number is positive.")
elif num < 0:
   print("The number is negative.")
else:
   print("The number is zero.")
```

Determine if a person is eligible to vote based on their age:

```
age = int(input("Enter your age: "))
if age >= 18:
   print("You are eligible to vote.")
else:
   print("You are not eligible to vote.")
```

Find the maximum of two numbers using if-else statements:

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
if num1 > num2:
   print("The maximum number is", num1)
else:
   print("The maximum number is", num2)
```

Classify a given year as a leap year or not:

```
year = int(input("Enter a year: "))
if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
   print(year, "is a leap year.")
else:
   print(year, "is not a leap year.")
```

Check whether a character is a vowel or a consonant:

```
char = input("Enter a character: ").lower()
if char in 'aeiou':
   print("The character is a vowel.")
else:
   print("The character is a consonant.")
```

Determine whether a given number is even or odd:

```
num = int(input("Enter a number: "))
if num % 2 == 0:
   print("The number is even.")
else:
   print("The number is odd.")
```

Calculate the absolute value of a number without using the `abs()` function:

```
num = float(input("Enter a number: "))
if num >= 0:
   abs_value = num
else:
   abs_value = -num
print("The absolute value is", abs_value)
```

Determine the largest of three given numbers using if-else statements:

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))
if num1 >= num2 and num1 >= num3:
   largest = num1
elif num2 >= num1 and num2 >= num3:
   largest = num2
else:
```

```
    largest = num3
print("The largest number is", largest)
```

Check if a given string is a palindrome:

```
string = input("Enter a string: ")
if string == string[::-1]:
    print("The string is a palindrome.")
else:
    print("The string is not a palindrome.")
```

Calculate the grade based on a student's score:

```
score = float(input("Enter the score: "))
if score >= 90:
    grade = 'A'
elif score >= 80:
    grade = 'B'
elif score >= 70:
    grade = 'C'
elif score >= 60:
    grade = 'D'
else:
    grade = 'F'
print("The grade is", grade)
```

Nested If-Else Statements

Find the largest among three numbers using nested if-else statements:

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))

if num1 >= num2:
    if num1 >= num3:
        largest = num1
    else:
        largest = num3
else:
    if num2 >= num3:
        largest = num2
    else:
        largest = num3

print("The largest number is", largest)
```

Determine if a triangle is equilateral, isosceles, or scalene:

```
side1 = float(input("Enter the first side: "))
side2 = float(input("Enter the second side: "))
side3 = float(input("Enter the third side: "))

if side1 == side2 == side3:
    print("The triangle is equilateral.")
elif side1 == side2 or side2 == side3 or side1 == side3:
    print("The triangle is isosceles.")
else:
    print("The triangle is scalene.")
```

Check if a year is a leap year and also if it is a century year:

```
year = int(input("Enter a year: "))

if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    if year % 100 == 0:
```

```
        print(year, "is a leap year and a century year.")
    else:
        print(year, "is a leap year but not a century year.")
else:
    print(year, "is not a leap year.")
```

Determine if a number is positive, negative, or zero:

```
num = float(input("Enter a number: "))

if num > 0:
    print("The number is positive.")
elif num < 0:
    print("The number is negative.")
else:
    print("The number is zero.")
```

Check if a person is a teenager (between 13 and 19 years old):

```
age = int(input("Enter your age: "))

if age >= 13:
    if age <= 19:
        print("You are a teenager.")
    else:
        print("You are not a teenager.")
else:
    print("You are not a teenager.")
```

Determine the type of angle based on its measure (acute, obtuse, or right):

```
angle = float(input("Enter the angle in degrees: "))

if angle < 90:
    print("The angle is acute.")
elif angle == 90:
    print("The angle is right.")
else:
    print("The angle is obtuse.")
```

Calculate the roots of a quadratic equation:

```
import math

a = float(input("Enter coefficient a: "))
b = float(input("Enter coefficient b: "))
c = float(input("Enter coefficient c: "))

discriminant = b**2 - 4*a*c

if discriminant > 0:
    root1 = (-b + math.sqrt(discriminant)) / (2*a)
    root2 = (-b - math.sqrt(discriminant)) / (2*a)
    print("The roots are real and different:", root1, root2)
elif discriminant == 0:
    root = -b / (2*a)
    print("The roots are real and the same:", root)
else:
    real_part = -b / (2*a)
    imaginary_part = math.sqrt(-discriminant) / (2*a)
    print("The roots are complex and different:", real_part, "+", imaginary_part, "i and", real_part, "-",
imaginary_part, "i")
```

Determine the day of the week based on a user-provided number:

```
day_num = int(input("Enter a number (1-7): "))
```

```
if day_num == 1:
   day = "Monday"
elif day_num == 2:
   day = "Tuesday"
elif day_num == 3:
   day = "Wednesday"
elif day_num == 4:
   day = "Thursday"
elif day_num == 5:
   day = "Friday"
elif day_num == 6:
   day = "Saturday"
elif day_num == 7:
   day = "Sunday"
else:
   day = "Invalid number"

print("The day is", day)
```

Determine if a year is a leap year and also if it is evenly divisible by 400:

```
year = int(input("Enter a year: "))

if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
   if year % 400 == 0:
      print(year, "is a leap year and is evenly divisible by 400.")
   else:
      print(year, "is a leap year but not evenly divisible by 400.")
else:
   print(year, "is not a leap year.")
```

Check if a given number is prime or not using nested if-else statements:

```
num = int(input("Enter a number: "))

if num > 1:
   for i in range(2, int(num ** 0.5) + 1):
      if num % i == 0:
         print(num, "is not a prime number.")
         break
   else:
      print(num, "is a prime number.")
else:
   print(num, "is not a prime number.")
```

Elif Statements

Assign grades based on different ranges of scores using elif statements:

```
score = float(input("Enter the score: "))

if score >= 90:
   grade = 'A'
elif score >= 80:
   grade = 'B'
elif score >= 70:
   grade = 'C'
elif score >= 60:
   grade = 'D'
else:
   grade = 'F'
```

```
print("The grade is", grade)
```

Determine the type of a triangle based on its angles:

```
angle1 = float(input("Enter the first angle: "))
angle2 = float(input("Enter the second angle: "))
angle3 = float(input("Enter the third angle: "))

if angle1 + angle2 + angle3 == 180:
    if angle1 == 90 or angle2 == 90 or angle3 == 90:
        print("The triangle is a right triangle.")
    elif angle1 > 90 or angle2 > 90 or angle3 > 90:
        print("The triangle is an obtuse triangle.")
    else:
        print("The triangle is an acute triangle.")
else:
    print("The angles do not form a triangle.")
```

Categorize a given person's BMI into underweight, normal, overweight, or obese using elif statements:

```
weight = float(input("Enter your weight in kilograms: "))
height = float(input("Enter your height in meters: "))
bmi = weight / (height ** 2)

if bmi < 18.5:
    category = "Underweight"
elif 18.5 <= bmi < 24.9:
    category = "Normal weight"
elif 25 <= bmi < 29.9:
    category = "Overweight"
else:
    category = "Obesity"

print("Your BMI is", bmi, "and you are categorized as", category)
```

Determine whether a given number is positive, negative, or zero using elif statements:

```
num = float(input("Enter a number: "))

if num > 0:
    print("The number is positive.")
elif num < 0:
    print("The number is negative.")
else:
    print("The number is zero.")
```

Determine the type of a character (uppercase, lowercase, or special) using elif statements:

```
char = input("Enter a character: ")

if char.isupper():
    print("The character is uppercase.")
elif char.islower():
    print("The character is lowercase.")
else:
    print("The character is a special character.")
```

Calculate the discounted price based on different purchase amounts using elif statements:

```
purchase_amount = float(input("Enter the purchase amount: "))

if purchase_amount >= 1000:
```

```
    discount = purchase_amount * 0.1
elif purchase_amount >= 500:
    discount = purchase_amount * 0.05
else:
    discount = 0

discounted_price = purchase_amount - discount
print("The discounted price is", discounted_price)
```

Elif Statements

Calculate the electricity bill based on different consumption slabs using elif statements:

```
units = float(input("Enter the number of units consumed: "))

if units <= 100:
    bill = units * 1.5
elif units <= 200:
    bill = 100 * 1.5 + (units - 100) * 2.5
elif units <= 300:
    bill = 100 * 1.5 + 100 * 2.5 + (units - 200) * 4
else:
    bill = 100 * 1.5 + 100 * 2.5 + 100 * 4 + (units - 300) * 6

print("The electricity bill is", bill)
```

Determine the type of quadrilateral based on its angles and sides using elif statements:

```
side1 = float(input("Enter the first side: "))
side2 = float(input("Enter the second side: "))
side3 = float(input("Enter the third side: "))
side4 = float(input("Enter the fourth side: "))
angle1 = float(input("Enter the first angle: "))
angle2 = float(input("Enter the second angle: "))
angle3 = float(input("Enter the third angle: "))
angle4 = float(input("Enter the fourth angle: "))

if side1 == side2 == side3 == side4 and angle1 == angle2 == angle3 == angle4 == 90:
    print("The quadrilateral is a square.")
elif side1 == side3 and side2 == side4 and angle1 == angle2 == angle3 == angle4 == 90:
    print("The quadrilateral is a rectangle.")
elif side1 == side3 and side2 == side4:
    print("The quadrilateral is a rhombus.")
elif angle1 == angle3 and angle2 == angle4:
    print("The quadrilateral is a parallelogram.")
else:
    print("The quadrilateral is an irregular quadrilateral.")
```

Determine the season based on a user-provided month using elif statements:

```
month = input("Enter the month: ").lower()

if month in ["december", "january", "february"]:
    season = "Winter"
elif month in ["march", "april", "may"]:
    season = "Spring"
elif month in ["june", "july", "august"]:
    season = "Summer"
elif month in ["september", "october", "november"]:
    season = "Autumn"
```

```
else:
    season = "Invalid month"

print("The season is", season)
```

Determine the type of a year (leap or common) and month (30 or 31 days) using elif statements:

```
year = int(input("Enter a year: "))
month = input("Enter the month: ").lower()

if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    year_type = "Leap year"
else:
    year_type = "Common year"

if month in ["april", "june", "september", "november"]:
    days = 30
elif month == "february":
    if year_type == "Leap year":
        days = 29
    else:
        days = 28
elif month in ["january", "march", "may", "july", "august", "october", "december"]:
    days = 31
else:
    days = "Invalid month"

print(f"{year} is a {year_type} and {month.capitalize()} has {days} days.")
```

Basic Level

Check if a number is positive, negative, or zero:

```
def check_number(num):
    if num > 0:
        return "Positive"
    elif num < 0:
        return "Negative"
    else:
        return "Zero"

number = float(input("Enter a number: "))
print(check_number(number))
```

Determine if a person is eligible to vote based on their age:

```
def check_voting_eligibility(age):
    if age >= 18:
        return "Eligible to vote"
    else:
        return "Not eligible to vote"

age = int(input("Enter your age: "))
print(check_voting_eligibility(age))
```

Find the maximum of two given numbers using conditional statements:

```
def find_max(num1, num2):
    if num1 > num2:
        return num1
    else:
        return num2
```

```
number1 = float(input("Enter first number: "))
number2 = float(input("Enter second number: "))
print("The maximum number is:", find_max(number1, number2))
```

Calculate the grade of a student based on their exam score:

```
def calculate_grade(score):
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 70:
        return "C"
    elif score >= 60:
        return "D"
    else:
        return "F"

score = float(input("Enter the exam score: "))
print("The grade is:", calculate_grade(score))
```

Check if a year is a leap year or not:

```
def is_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False

year = int(input("Enter a year: "))
if is_leap_year(year):
    print(year, "is a leap year.")
else:
    print(year, "is not a leap year.")
```

Classify a triangle based on its sides' lengths:

```
def classify_triangle(a, b, c):
    if a == b == c:
        return "Equilateral"
    elif a == b or b == c or a == c:
        return "Isosceles"
    else:
        return "Scalene"

side1 = float(input("Enter the length of the first side: "))
side2 = float(input("Enter the length of the second side: "))
side3 = float(input("Enter the length of the third side: "))
print("The triangle is:", classify_triangle(side1, side2, side3))
```

Determine the largest of three given numbers:

```
def find_largest(num1, num2, num3):
    if num1 >= num2 and num1 >= num3:
        return num1
    elif num2 >= num1 and num2 >= num3:
        return num2
    else:
        return num3

number1 = float(input("Enter first number: "))
number2 = float(input("Enter second number: "))
number3 = float(input("Enter third number: "))
print("The largest number is:", find_largest(number1, number2, number3))
```

Check whether a character is a vowel or a consonant:

```
def check_vowel_or_consonant(char):
    vowels = "aeiouAEIOU"
    if char in vowels:
        return "Vowel"
    else:
        return "Consonant"

character = input("Enter a character: ")
print(character, "is a", check_vowel_or_consonant(character))
```

Calculate the total cost of a shopping cart based on discounts:

```
def calculate_total_cost(prices, discount):
    total = sum(prices)
    discounted_total = total - (total * discount / 100)
    return discounted_total

prices = [float(x) for x in input("Enter the prices of items separated by space: ").split()]
discount = float(input("Enter the discount percentage: "))
print("The total cost after discount is:", calculate_total_cost(prices, discount))
```

Check if a given number is even or odd:

```
def check_even_or_odd(num):
    if num % 2 == 0:
        return "Even"
    else:
        return "Odd"

number = int(input("Enter a number: "))
print(number, "is", check_even_or_odd(number))
```

Intermediate Level

Calculate the roots of a quadratic equation:

```
import math

def calculate_roots(a, b, c):
    discriminant = b**2 - 4*a*c
    if discriminant > 0:
        root1 = (-b + math.sqrt(discriminant)) / (2*a)
        root2 = (-b - math.sqrt(discriminant)) / (2*a)
        return root1, root2
    elif discriminant == 0:
        root = -b / (2*a)
        return root, root
    else:
        return "Complex roots"

a = float(input("Enter coefficient a: "))
b = float(input("Enter coefficient b: "))
c = float(input("Enter coefficient c: "))
roots = calculate_roots(a, b, c)
print("The roots are:", roots)
```

Determine the day of the week based on the day number (1-7):

```
def day_of_week(day_number):
    days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
    if 1 <= day_number <= 7:
        return days[day_number - 1]
```

```
    else:
        return "Invalid day number"

day_number = int(input("Enter day number (1-7): "))
print("The day is:", day_of_week(day_number))
```

Calculate the factorial of a given number using recursion:

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

number = int(input("Enter a number: "))
print("The factorial of", number, "is", factorial(number))
```

Find the largest among three numbers without using the max() function:

```
def find_largest(num1, num2, num3):
    if num1 >= num2 and num1 >= num3:
        return num1
    elif num2 >= num1 and num2 >= num3:
        return num2
    else:
        return num3

number1 = float(input("Enter first number: "))
number2 = float(input("Enter second number: "))
number3 = float(input("Enter third number: "))
print("The largest number is:", find_largest(number1, number2, number3))
```

Simulate a basic ATM transaction menu:

```
def atm_menu():
    balance = 1000  # Initial balance
    while True:
        print("\nATM Menu:")
        print("1. Check Balance")
        print("2. Deposit Money")
        print("3. Withdraw Money")
        print("4. Exit")
        choice = int(input("Enter your choice: "))

        if choice == 1:
            print("Your balance is:", balance)
        elif choice == 2:
            amount = float(input("Enter amount to deposit: "))
            balance += amount
            print("Amount deposited successfully. New balance is:", balance)
        elif choice == 3:
            amount = float(input("Enter amount to withdraw: "))
            if amount <= balance:
                balance -= amount
                print("Amount withdrawn successfully. New balance is:", balance)
            else:
                print("Insufficient balance.")
        elif choice == 4:
            print("Thank you for using the ATM. Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")
```

```
atm_menu()
```

Check if a given string is a palindrome or not:

```
def is_palindrome(s):
    return s == s[::-1]

string = input("Enter a string: ")
if is_palindrome(string):
    print(string, "is a palindrome.")
else:
    print(string, "is not a palindrome.")
```

Calculate the average of a list of numbers, excluding the smallest and largest values:

```
def average_excluding_extremes(numbers):
    if len(numbers) <= 2:
        return "Not enough numbers to exclude extremes."
    numbers.sort()
    return sum(numbers[1:-1]) / (len(numbers) - 2)

numbers = [float(x) for x in input("Enter numbers separated by space: ").split()]
print("The average excluding the smallest and largest values is:", average_excluding_extremes(numbers))
```

Convert a given temperature from Celsius to Fahrenheit:

```
def celsius_to_fahrenheit(celsius):
    return (celsius * 9/5) + 32

celsius = float(input("Enter temperature in Celsius: "))
print("Temperature in Fahrenheit is:", celsius_to_fahrenheit(celsius))
```

Simulate a basic calculator for addition, subtraction, multiplication, and division:

```
def calculator():
    print("Basic Calculator")
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))
    operation = input("Enter operation (+, -, *, /): ")

    if operation == '+':
        result = num1 + num2
    elif operation == '-':
        result = num1 - num2
    elif operation == '*':
        result = num1 * num2
    elif operation == '/':
        if num2 != 0:
            result = num1 / num2
        else:
            return "Error! Division by zero."
    else:
        return "Invalid operation."

    return "The result is: " + str(result)

print(calculator())
```

Determine the roots of a cubic equation using the Cardano formula:

```
import cmath

def solve_cubic(a, b, c, d):
    if a == 0:
        return "Not a cubic equation."
```

```
    f = ((3*c/a) - (b**2/a**2)) / 3
    g = ((2*b**3/a**3) - (9*b*c/a**2) + (27*d/a)) / 27
    h = (g**2 / 4) + (f**3 / 27)

    if h > 0:
        r = -(g/2) + cmath.sqrt(h)
        s = cmath.cbrt(r)
        t = -(g/2) - cmath.sqrt(h)
        u = cmath.cbrt(t)
        root1 = (s + u) - (b / (3*a))
        return root1
    elif f == 0 and g == 0 and h == 0:
        root = -cmath.cbrt(d/a)
        return root, root, root
    else:
        i = cmath.sqrt((g**2 / 4) - h)
        j = cmath.cbrt(i)
        k = cmath.acos(-(g / (2*i)))
        l = -j
        m = cmath.cos(k / 3)
        n = cmath.sqrt(3) * cmath.sin(k / 3)
        p = -(b / (3*a))
        root1 = 2*j*cmath.cos(k/3) - (b/(3*a))
        root2 = l * (m + n) + p
        root3 = l * (m - n) + p
        return root1, root2, root3

a = float(input("Enter coefficient a: "))
b = float(input("Enter coefficient b: "))
c = float(input("Enter coefficient c: "))
d = float(input("Enter coefficient d: "))
roots = solve_cubic(a, b, c, d)
print("The roots are:", roots)
```

Advanced Level

Calculate the income tax based on the user's income and tax brackets:

```
def calculate_tax(income):
    if income <= 250000:
        return 0
    elif income <= 500000:
        return (income - 250000) * 0.05
    elif income <= 1000000:
        return (income - 500000) * 0.2 + 12500
    else:
        return (income - 1000000) * 0.3 + 112500

income = float(input("Enter your annual income: "))
tax = calculate_tax(income)
print("Your income tax is:", tax)
```

Simulate a rock-paper-scissors game against the computer:

```
import random

def rock_paper_scissors(user_choice):
    choices = ["rock", "paper", "scissors"]
    computer_choice = random.choice(choices)
    print("Computer chose:", computer_choice)
```

```
    if user_choice == computer_choice:
        return "It's a tie!"
    elif (user_choice == "rock" and computer_choice == "scissors") or \
         (user_choice == "paper" and computer_choice == "rock") or \
         (user_choice == "scissors" and computer_choice == "paper"):
        return "You win!"
    else:
        return "You lose!"

user_choice = input("Enter rock, paper, or scissors: ").lower()
print(rock_paper_scissors(user_choice))
```

Generate a random password based on user preferences (length, complexity):

```
import random
import string

def generate_password(length, complexity):
    characters = string.ascii_letters
    if complexity >= 2:
        characters += string.digits
    if complexity >= 3:
        characters += string.punctuation

    password = ''.join(random.choice(characters) for _ in range(length))
    return password

length = int(input("Enter the desired password length: "))
complexity = int(input("Enter complexity level (1-3): "))
print("Generated password:", generate_password(length, complexity))
```

Implement a simple text-based adventure game with branching scenarios:

```
def adventure_game():
    print("Welcome to the Adventure Game!")
    print("You are in a dark forest. You can go left or right.")
    choice1 = input("Which way do you want to go? (left/right): ").lower()

    if choice1 == "left":
        print("You encounter a wild animal!")
        choice2 = input("Do you want to run or fight? (run/fight): ").lower()
        if choice2 == "run":
            print("You safely escape. You win!")
        else:
            print("You fight bravely but lose. Game over.")
    else:
        print("You find a treasure chest!")
        choice2 = input("Do you want to open it? (yes/no): ").lower()
        if choice2 == "yes":
            print("You find gold and jewels. You win!")
        else:
            print("You walk away. Game over.")

adventure_game()
```

Solve a linear equation for x, considering different cases:

```
def solve_linear_equation(a, b):
    if a == 0:
        if b == 0:
            return "Infinite solutions"
```

```
        else:
            return "No solution"
    else:
        return -b / a

a = float(input("Enter coefficient a: "))
b = float(input("Enter constant b: "))
solution = solve_linear_equation(a, b)
print("The solution is:", solution)
```

Simulate a basic quiz game with multiple-choice questions and scoring:

```
def quiz_game():
    questions = {
        "What is the capital of France?": "a",
        "What is 2 + 2?": "b",
        "What is the color of the sky?": "c"
    }
    options = [
        ["a. Paris", "b. London", "c. Rome"],
        ["a. 3", "b. 4", "c. 5"],
        ["a. Green", "b. Red", "c. Blue"]
    ]
    score = 0

    for i, (question, answer) in enumerate(questions.items()):
        print(question)
        for option in options[i]:
            print(option)
        user_answer = input("Enter your answer (a/b/c): ").lower()
        if user_answer == answer:
            score += 1

    print("Your final score is:", score)

quiz_game()
```

Determine whether a given year is a prime number or not:

```
def is_prime(year):
    if year <= 1:
        return False
    for i in range(2, int(year**0.5) + 1):
        if year % i == 0:
            return False
    return True

year = int(input("Enter a year: "))
if is_prime(year):
    print(year, "is a prime number.")
else:
    print(year, "is not a prime number.")
```

Sort three numbers in ascending order using conditional statements:

```
def sort_three_numbers(a, b, c):
    if a > b:
        a, b = b, a
    if a > c:
        a, c = c, a
    if b > c:
        b, c = c, b
```

```
    return a, b, c

num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
num3 = float(input("Enter third number: "))
sorted_numbers = sort_three_numbers(num1, num2, num3)
print("The numbers in ascending order are:", sorted_numbers)
```

Determine the roots of a quartic equation using numerical methods:

```
import numpy as np

def solve_quartic(a, b, c, d, e):
    coefficients = [a, b, c, d, e]
    roots = np.roots(coefficients)
    return roots

a = float(input("Enter coefficient a: "))
b = float(input("Enter coefficient b: "))
c = float(input("Enter coefficient c: "))
d = float(input("Enter coefficient d: "))
e = float(input("Enter coefficient e: "))
roots = solve_quartic(a, b, c, d, e)
print("The roots are:", roots)
```

Calculate the BMI (Body Mass Index) and provide health recommendations based on the user's input:

```
def calculate_bmi(weight, height):
    bmi = weight / (height ** 2)
    if bmi < 18.5:
        return bmi, "Underweight"
    elif 18.5 <= bmi < 24.9:
        return bmi, "Normal weight"
    elif 25 <= bmi < 29.9:
        return bmi, "Overweight"
    else:
        return bmi, "Obesity"

weight = float(input("Enter your weight in kg: "))
height = float(input("Enter your height in meters: "))
bmi, category = calculate_bmi(weight, height)
print("Your BMI is:", bmi)
print("You are classified as:", category)
```

Challenge Level

Validate a password based on complexity rules (length, characters, etc.):

```
import re

def validate_password(password):
    if len(password) < 8:
        return "Password must be at least 8 characters long."
    if not re.search("[a-z]", password):
        return "Password must contain at least one lowercase letter."
    if not re.search("[A-Z]", password):
        return "Password must contain at least one uppercase letter."
    if not re.search("[0-9]", password):
        return "Password must contain at least one digit."
    if not re.search("[@#$%^&+=]", password):
        return "Password must contain at least one special character (@#$%^&+=)."
    return "Password is valid."
```

```
password = input("Enter a password: ")
print(validate_password(password))
```

Perform matrix addition and subtraction based on user input:

```
def matrix_addition(matrix1, matrix2):
    result = [[matrix1[i][j] + matrix2[i][j] for j in range(len(matrix1[0]))] for i in range(len(matrix1))]
    return result

def matrix_subtraction(matrix1, matrix2):
    result = [[matrix1[i][j] - matrix2[i][j] for j in range(len(matrix1[0]))] for i in range(len(matrix1))]
    return result

def get_matrix(rows, cols):
    matrix = []
    for i in range(rows):
        row = list(map(int, input(f"Enter row {i+1} (space-separated): ").split()))
        matrix.append(row)
    return matrix

rows = int(input("Enter the number of rows: "))
cols = int(input("Enter the number of columns: "))

print("Enter the first matrix:")
matrix1 = get_matrix(rows, cols)

print("Enter the second matrix:")
matrix2 = get_matrix(rows, cols)

print("Matrix Addition Result:")
for row in matrix_addition(matrix1, matrix2):
    print(row)

print("Matrix Subtraction Result:")
for row in matrix_subtraction(matrix1, matrix2):
    print(row)
```

Calculate the greatest common divisor (GCD) of two numbers using the Euclidean algorithm:

```
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
print("The GCD of", num1, "and", num2, "is", gcd(num1, num2))
```

Perform matrix multiplication using nested loops and conditional statements:

```
def matrix_multiplication(matrix1, matrix2):
    result = [[0 for _ in range(len(matrix2[0]))] for _ in range(len(matrix1))]
    for i in range(len(matrix1)):
        for j in range(len(matrix2[0])):
            for k in range(len(matrix2)):
                result[i][j] += matrix1[i][k] * matrix2[k][j]
    return result

def get_matrix(rows, cols):
    matrix = []
```

```
    for i in range(rows):
        row = list(map(int, input(f"Enter row {i+1} (space-separated): ").split()))
        matrix.append(row)
    return matrix

rows1 = int(input("Enter the number of rows for the first matrix: "))
cols1 = int(input("Enter the number of columns for the first matrix: "))
rows2 = int(input("Enter the number of rows for the second matrix: "))
cols2 = int(input("Enter the number of columns for the second matrix: "))

if cols1 != rows2:
    print("Matrix multiplication is not possible with the given dimensions.")
else:
    print("Enter the first matrix:")
    matrix1 = get_matrix(rows1, cols1)

    print("Enter the second matrix:")
    matrix2 = get_matrix(rows2, cols2)

    print("Matrix Multiplication Result:")
    for row in matrix_multiplication(matrix1, matrix2):
        print(row)
```

Simulate a basic text-based tic-tac-toe game against the computer:

```
import random

def print_board(board):
    for row in board:
        print(" | ".join(row))
        print("-" * 5)

def check_winner(board, player):
    for row in board:
        if all(s == player for s in row):
            return True
    for col in range(3):
        if all(row[col] == player for row in board):
            return True
    if all(board[i][i] == player for i in range(3)) or all(board[i][2-i] == player for i in range(3)):
        return True
    return False

def get_computer_move(board):
    empty_cells = [(i, j) for i in range(3) for j in range(3) if board[i][j] == " "]
    return random.choice(empty_cells)

def tic_tac_toe():
    board = [[" " for _ in range(3)] for _ in range(3)]
    current_player = "X"

    for _ in range(9):
        print_board(board)
        if current_player == "X":
            row, col = map(int, input("Enter your move (row and column): ").split())
        else:
            row, col = get_computer_move(board)
            print(f"Computer chose: {row} {col}")
```

```
        if board[row][col] == " ":
            board[row][col] = current_player
            if check_winner(board, current_player):
                print_board(board)
                print(f"Player {current_player} wins!")
                return
            current_player = "O" if current_player == "X" else "X"
        else:
            print("Cell already taken. Try again.")

    print_board(board)
    print("It's a tie!")

tic_tac_toe()
```

Generate Fibonacci numbers up to a specified term using iterative methods:

```
def fibonacci(n):
    fib_sequence = [0, 1]
    for i in range(2, n):
        fib_sequence.append(fib_sequence[-1] + fib_sequence[-2])
    return fib_sequence[:n]

n = int(input("Enter the number of terms: "))
print("Fibonacci sequence:", fibonacci(n))
```

Calculate the nth term of the Fibonacci sequence using memoization:

```
def fibonacci_memo(n, memo={}):
    if n in memo:
        return memo[n]
    if n <= 1:
        return n
    memo[n] = fibonacci_memo(n-1, memo) + fibonacci_memo(n-2, memo)
    return memo[n]

n = int(input("Enter the term number: "))
print(f"The {n}th term of the Fibonacci sequence is:", fibonacci_memo(n))
```

Generate a calendar for a given month and year using conditional statements:

```
import calendar

def generate_calendar(year, month):
    cal = calendar.monthcalendar(year, month)
    print(calendar.month_name[month], year)
    print("Mo Tu We Th Fr Sa Su")
    for week in cal:
        print(" ".join(f"{day:2}" if day != 0 else "  " for day in week))

year = int(input("Enter the year: "))
month = int(input("Enter the month (1-12): "))
generate_calendar(year, month)
```

39. Build a program that simulates a basic text-based blackjack game against the computer.

```
import random

def deal_card():
"""Returns a random card from the deck."""
cards = [2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10, 11]
return random.choice(cards)
```

```
def calculate_score(hand):
"""Calculates the score of a hand."""
if sum(hand) == 21 and len(hand) == 2:
return 0  # Blackjack
if 11 in hand and sum(hand) > 21:
hand.remove(11)
hand.append(1)
return sum(hand)

def compare_scores(user_score, computer_score):
"""Compares the scores of the user and the computer."""
if user_score == computer_score:
return "Draw!"
elif computer_score == 0:
return "Lose, opponent has Blackjack!"
elif user_score == 0:
return "Win with a Blackjack!"
elif user_score > 21:
return "You went over. You lose!"
elif computer_score > 21:
return "Opponent went over. You win!"
elif user_score > computer_score:
return "You win!"
else:
return "You lose!"

def blackjack():
user_hand = [deal_card(), deal_card()]
computer_hand = [deal_card(), deal_card()]

game_over = False

while not game_over:
user_score = calculate_score(user_hand)
computer_score = calculate_score(computer_hand)
print(f"Your cards: {user_hand}, current score: {user_score}")
print(f"Computer's first card: {computer_hand[0]}")

if user_score == 0 or computer_score == 0 or user_score > 21:
game_over = True
else:
user_should_deal = input("Type 'y' to get another card, type 'n' to pass: ").lower()
if user_should_deal == 'y':
        user_hand.append(deal_card())
else:
        game_over = True

while computer_score != 0 and computer_score < 17:
computer_hand.append(deal_card())
computer_score = calculate_score(computer_hand)
```

```
print(f"Your final hand: {user_hand}, final score: {user_score}")
print(f"Computer's final hand: {computer_hand}, final score: {computer_score}")
print(compare_scores(user_score, computer_score))

blackjack()
```

40. Write a program that generates the prime factors of a given number using trial division.

```
def prime_factors(n):
factors = []
# Check for number of 2s that divide n
while n % 2 == 0:
factors.append(2)
n = n // 2

# n must be odd at this point, so a skip of 2 (i.e., i = i + 2) can be used
for i in range(3, int(n**0.5) + 1, 2):
# While i divides n, append i and divide n
while n % i == 0:
factors.append(i)
n = n // i

# This condition is to check if n is a prime number greater than 2
if n > 2:
factors.append(n)

return factors

number = int(input("Enter a number: "))
print("The prime factors of", number, "are:", prime_factors(number))
```