controlflow

September 11, 2024

[ ]:
```
#Control Flow
```

[2]:
```
# Basic If-Else Statement
#1.Write a Python program to check if a given number is positive or negative.

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

```
enter a number: -4
```

```
The number is negative.
```

[3]:
```
#2.Create a program that determines if a person is eligible to vote based on␣
 ↪their age.
def check_voting_eligibility(age):
    if age>=18:
        print("you are eligible to vote.")
    else:
        print("you are not eligible to vote yet.")
age=int(input("enter your age:"))
check_voting_eligibility(age)
```

```
enter your age: 17
```

```
you are not eligible to vote yet.
```

[6]:
```
#3. Develop a program to find the maximum of two numbers using if-else␣
 ↪statements
def find_maximum(num1,num2):
    if num1>num2:
        return num1
    else:
```

```
        return num2
num1=float(input("enter the first number:"))
num2=float(input("enter the second number:"))
maximum=find_maximum(num1,num2)
print("the maximum of the two numbers is:",maximum)
```

```
enter the first number: 2
enter the second number: 3
```

```
the maximum of the two numbers is: 3.0
```

[8]:
```
#4.Write a Python script to classify a given year as a leap year or not.
def is_leap_year(year):
    if(year%4==0and 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 a not a leap year.")
```

```
enter a year 2001
```

```
2001 is a not a leap year.
```

[2]:
```
#5.Create a program that checks whether a character is a vowel or a consonant.
def check_vowel_or_consonant(char):
    vowels = "aeiouAEIOU"
    if char.isalpha():
        if char in vowels:
            print(char, "is a vowel.")
        else:
            print(char, "is a consonant.")
    else:
        print("Invalid input. Please enter a valid alphabet character.")

character = input("Enter a character: ")
if len(character) == 1:
    check_vowel_or_consonant(character)
else:
    print("Please enter only one character.")
```

```
Enter a character:  a
```

```
a is a vowel.
```

[1]:
```
#6. Implement a program to determine whether a given number is even or odd.
def check_even_odd(num):
    if num % 2 == 0:
        return "Even"
    else:
        return "Odd"

num = int(input("Enter a number: "))
result = check_even_odd(num)
print(f"The number {num} is {result}.")
```

```
Enter a number:  6
```

```
The number 6 is Even.
```

[2]:
```
#7. Write a Python function to calculate the absolute value of a number without␣
 ↪using the `abs()` function.
def absolute_value(num):
    if num < 0:
        return -num
    else:
        return num

num = float(input("Enter a number: "))
print(f"The absolute value of {num} is {absolute_value(num)}.")
```

```
Enter a number:  5
```

```
The absolute value of 5.0 is 5.0.
```

[6]:
```
#8. Develop a program that determines the largest of three given numbers using␣
 ↪if-else statements
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

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

largest = find_largest(num1, num2, num3)
print(f"The largest number among {num1}, {num2},{num3} is {largest}.")
```

```
Enter the first number:  5
Enter the second number:  8
```

```
Enter the third number:  9

The largest number among 5.0, 8.0,9.0 is 9.0.
```

[7]:
```python
#9. Create a program that checks if a given string is a palindrome.
def is_palindrome(s):

    s = s.replace(" ", "").lower()

    return s == s[::-1]

string = input("Enter a string: ")
if is_palindrome(string):
    print(f"The string '{string}' is a palindrome.")
else:
    print(f"The string '{string}' is not a palindrome.")
```

```
Enter a string:  poja

The string 'poja' is not a palindrome.
```

[8]:
```python
#10. Write a Python program to calculate the grade based on a student's 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 student's score: "))
grade = calculate_grade(score)
print(f"The student's grade is: {grade}")
```

```
Enter the student's score:  88

The student's grade is: B
```

[11]:
```python
# Nested If-Else Statements
#11.Write a program to find the largest among three numbers using nested␣
 ↪if-else statements
num1=float(input("enter first number:"))
num2=float(input("enter second number:"))
num3=float(input("enter 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)
```

```
enter first number: 4
enter second number: 5
enter third number: 7

the largest number is  7.0
```

[12]:
```
#12.Implement a program to determine if a triangle is equilateral, isosceles,␣
 ↪or scalene.
def triangle_type(side1, side2, side3):
    if side1 == side2 == side3:
        return "Equilateral"
    elif side1 == side2 or side1 == side3 or side2 == side3:
        return "Isosceles"
    else:
        return "Scalene"

def main():
    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: "))

    if side1 + side2 > side3 and side1 + side3 > side2 and side2 + side3 >␣
 ↪side1:
        triangle = triangle_type(side1, side2, side3)
        print("The triangle is", triangle)
    else:
        print("Invalid triangle")

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

```
Enter the length of the first side:  4
Enter the length of the second side:  6
Enter the length of the third side:  8

The triangle is Scalene
```

[14]:
```
#13.Develop a program that checks if a year is a leap year and also if it is a␣
↪century year.
def is_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False

def is_century_year(year):
    if year % 100 == 0:
        return True
    else:
        return False

def main():
    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.")

    if is_century_year(year):
        print(year, "is a century year.")
    else:
        print(year, "is not a century year.")

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

```
Enter a year:  2000

2000 is a leap year.
2000 is a century year.
```

[15]:
```
#14.Write a Python script to determine if a number is positive, negative, or␣
↪zero.
def check_number(number):
    if number > 0:
        return "Positive"
    elif number < 0:
        return "Negative"
    else:
        return "Zero"

def main():
    number = float(input("Enter a number: "))
```

```
    result = check_number(number)
    print("The number is", result)

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

```
Enter a number:  5
```

```
The number is Positive
```

[19]:
```
#15.Create a program to check if a person is a teenager (between 13 and 19 years old).
age= int(input("entr your age:"))
if 13<= age<=19:
    print("you are a teenage")
else:
    print("you are not a teenage")
```

```
entr your age: 12
```

```
you are not a teenage
```

[1]:
```
#16.Develop a program that determines the type of angle based on its measure (acute, obtuse, or right).
def angle_type(angle):
    if angle < 90:
        return "Acute Angle"
    elif angle == 90:
        return "Right Angle"
    elif angle > 90 and angle < 180:
        return "Obtuse Angle"
    elif angle == 180:
        return "Straight Angle"
    else:
        return "Reflex Angle"

def main():
    angle = float(input("Enter the measure of the angle in degrees: "))
    angle_classification = angle_type(angle)
    print("The angle is a", angle_classification)

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

```
Enter the measure of the angle in degrees:  5
```

```
The angle is a Acute Angle
```

[5]:
```python
# 17. Write a Python program to calculate the roots of a quadratic equation.
import math

def quadratic_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:
        real_part = -b / (2*a)
        imaginary_part = math.sqrt(abs(discriminant)) / (2*a)
        root1 = complex(real_part, imaginary_part)
        root2 = complex(real_part, -imaginary_part)
        return root1, root2

def main():
    a = float(input("Enter the coefficient of x^2: "))
    b = float(input("Enter the coefficient of x: "))
    c = float(input("Enter the constant term: "))

    root1, root2 = quadratic_roots(a, b, c)

    print("Root 1:", root1)
    print("Root 2:", root2)

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

```
Enter the coefficient of x^2:  1
Enter the coefficient of x:  -3
Enter the constant term:  2
```

```
Root 1: 2.0
Root 2: 1.0
```

[7]:
```python
# 18. Implement a program to determine the day of the week based on a␣
 ↪user-provided number (1 for Monday, 2 for Tuesday, etc.).

def determine_day_of_week(day_number):
    days_of_week = {
        1: "Monday",
        2: "Tuesday",
        3: "Wednesday",
```

```
        4: "Thursday",
        5: "Friday",
        6: "Saturday",
        7: "Sunday"
    }
    return days_of_week.get(day_number, "Invalid day number")

def main():
    day_number = int(input("Enter a number between 1 and 7: "))
    day_of_week = determine_day_of_week(day_number)
    print("The day of the week is:", day_of_week)

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

```
Enter a number between 1 and 7:  4
The day of the week is: Thursday
```

[8]:
```
# 19. Create a program that determines if a year is a leap year and also if it␣
↪is evenly divisible by 400.
def is_leap_year(year):
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                return True
            else:
                return False
        else:
            return True
    else:
        return False

def main():
    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.")

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

```
Enter a year:  2001
2001 is not a leap year.
```

[9]:
```
# 20. Develop a program that checks if a given number is prime or not using␣
 ↪nested if-else statements.
def is_prime(number):
    if number <= 1:
        return False
    elif number <= 3:
        return True
    elif number % 2 == 0 or number % 3 == 0:
        return False
    else:
        i = 5
        while i * i <= number:
            if number % i == 0 or number % (i + 2) == 0:
                return False
            i += 6
        return True

def main():
    number = int(input("Enter a number to check if it's prime: "))
    if is_prime(number):
        print(number, "is a prime number.")
    else:
        print(number, "is not a prime number.")

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

```
Enter a number to check if it's prime:  3
3 is a prime number.
```

[1]:
```
# Elif Statements:
# 21. Write a Python program to assign grades based on different ranges of␣
 ↪scores using elif statements.
def assign_grade(score):
    if score>=90:
        return "A"
    elif score>=80:
        return "B"
    elif score>=60:
        return "C"
    else:
        return"D"
score=float(input("enter the score:"))
grade=assign_grade(score)
print("Grade:", grade)
```

```
enter the score: 60
```

```
Grade: C
```

[4]:
```python
# 22. Implement a program to determine the type of a triangle based on its angles.
def triangle_type(angle1, angle2, angle3):
    if angle1 + angle2 + angle3 != 180:
        return "Not a valid triangle"
    elif angle1 == angle2 == angle3:
        return "Equilateral triangle"
    elif angle1 == angle2 or angle1 == angle3 or angle2 == angle3:
        return "Isosceles triangle"
    else:
        return "Scalene triangle"

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

print("The triangle is a", triangle_type(angle1, angle2, angle3))
```

```
Enter the first angle of the triangle:  60
Enter the second angle of the triangle:  60
Enter the third angle of the triangle:  60

The triangle is a Equilateral triangle
```

[6]:
```python
#23. Develop a program to categorize a given person's BMI into underweight, normal, overweight, or obese using elif statements.
def calculate_bmi(weight, height):
    return weight / (height ** 2)

def categorize_bmi(bmi):
    if bmi < 18.5:
        return "Underweight"
    elif 18.5 <= bmi < 25:
        return "Normal"
    elif 25 <= bmi < 30:
        return "Overweight"
    else:
        return "Obese"

weight = float(input("Enter your weight in kilograms: "))
height = float(input("Enter your height in meters: "))

bmi = calculate_bmi(weight, height)
category = categorize_bmi(bmi)

print("Your BMI is:", bmi)
```

```
print("You are categorized as:", category)
```

```
Enter your weight in kilograms:  70
Enter your height in meters:  1.55

Your BMI is: 29.1363163371488
You are categorized as: Overweight
```

[7]:
```
#24. Create a program that determines whether a given number is positive,␣
 ↪negative, or zero using elif statements.
def check_number(number):
    if number > 0:
        return "Positive"
    elif number < 0:
        return "Negative"
    else:
        return "Zero"

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

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

```
Enter a number:  5

The number is Positive
```

[9]:
```
# 25. Write a Python script to determine the type of a character (uppercase,␣
 ↪lowercase, or special) using elif statement
def character_type(char):
    if char.isupper():
        return "Uppercase"
    elif char.islower():
        return "Lowercase"
    else:
        return "Special"

character = input("Enter a character: ")

print("Type of character:", character_type(character))
```

```
Enter a character:  pooja

Type of character: Lowercase
```

[1]:
```
#26. Implement a program to calculate the discounted price based on different␣
 ↪purchase amounts using elif statements.
def calculate_discounted_price(amount):
    if amount < 100:
        discount = 0
```

```
    elif amount < 500:
        discount = 0.05  # 5% discount for purchases between 100 and 499
    elif amount < 1000:
        discount = 0.1   # 10% discount for purchases between 500 and 999
    else:
        discount = 0.15  # 15% discount for purchases of 1000 or more

    discounted_price = amount - (amount * discount)
    return discounted_price

purchase_amount = float(input("Enter the purchase amount: "))
discounted_price = calculate_discounted_price(purchase_amount)
print("Discounted Price: $", discounted_price)
```

```
Enter the purchase amount:  60000
```

```
Discounted Price: $ 51000.0
```

[2]:
```
#27. Develop a program to calculate the electricity bill based on different␣
 ↪consumption slabs using elif statements.
def calculate_electricity_bill(units):
    if units <= 50:
        bill = units * 0.50
    elif units <= 150:
        bill = 25 + (units - 50) * 0.75
    elif units <= 250:
        bill = 100 + (units - 150) * 1.20
    else:
        bill = 220 + (units - 250) * 1.50

    return bill

units_consumed = float(input("Enter the units consumed: "))
electricity_bill = calculate_electricity_bill(units_consumed)
print("Electricity Bill: $", electricity_bill)
```

```
Enter the units consumed:  444
```

```
Electricity Bill: $ 511.0
```

[3]:
```
#28. Create a program to determine the type of quadrilateral based on its␣
 ↪angles and sides using elif statements.
def determine_quadrilateral_type(sides, angles):
    if sides.count(sides[0]) == 4 and angles.count(90) == 4:
        return "Square"
    elif sides[0] == sides[1] == sides[2] == sides[3]:
        return "Rhombus"
    elif angles.count(90) == 4:
```

```
        return "Rectangle"
    elif sides[0] == sides[2] and sides[1] == sides[3]:
        return "Parallelogram"
    else:
        return "Quadrilateral"

sides = [4, 4, 4, 4]
angles = [90, 90, 90, 90]
quadrilateral_type = determine_quadrilateral_type(sides, angles)
print("Type of Quadrilateral:", quadrilateral_type)
```

```
Type of Quadrilateral: Square
```

[5]:
```
# 29. Write a Python script to determine the season based on a user-provided␣
 ↪month using elif statements.
def determine_season(month):
    if month in [12, 1, 2]:
        season = "Winter"
    elif month in [3, 4, 5]:
        season = "Spring"
    elif month in [6, 7, 8]:
        season = "Summer"
    elif month in [9, 10, 11]:
        season = "Autumn"
    else:
        season = "Invalid month"

    return season

month = int(input("Enter the month (as a number): "))
result = determine_season(month)
print("The season for month", month, "is", result)
```

```
Enter the month (as a number):  7
```

```
The season for month 7 is Summer
```

[2]:
```
# 30. Implement a program to determine the type of a year (leap or common) and␣
 ↪month (30 or 31 days) using
elif statements.
def is_leap_year(year):
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                return True
            else:
                return False
        else:
```

```
            return True
    else:
        return False

def days_in_month(month, year):
    if month in [1, 3, 5, 7, 8, 10, 12]:
        return 31
    elif month in [4, 6, 9, 11]:
        return 30
    elif month == 2:
        if is_leap_year(year):
            return 29
        else:
            return 28
    else:
        return -1

year = int(input("Enter the year: "))
month = int(input("Enter the month (as a number): "))

if is_leap_year(year):
    print(year, "is a leap year.")
else:
    print(year, "is a common year.")

num_days = days_in_month(month, year)
if num_days == -1:
    print("Invalid month.")
else:
    print("The month", month, "of the year", year, "has", num_days, "days.")
```

```
Enter the year:  2000
Enter the month (as a number):  6
```

```
2000 is a leap year.
The month 6 of the year 2000 has 30 days.
```

[2]:
```
# Basic Level:
#1. Write a Python program that checks if a given number is positive, negative,␣
 ↪or zero.
def check_number(num):
    if num > 0:
        return "The number is positive."
    elif num < 0:
        return "The number is negative."
    else:
        return "The number is zero."
```

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

```
Enter a number:  90
```

```
The number is positive.
```

[3]:
```
#2. Create a program to determine if a person is eligible to vote based on␣
 ↪their age.
def check_voting_eligibility(age):
    if age >= 18:
        return "You are eligible to vote."
    else:
        return "You are not eligible to vote."


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

```
Enter your age:  22
```

```
You are eligible to vote.
```

[4]:
```
# 3. Write a program to find the maximum of two given numbers using conditional␣
 ↪statements.
def find_maximum(num1, num2):
    if num1 > num2:
        return num1
    elif num2 > num1:
        return num2
    else:
        return "Both numbers are equal."


number1 = float(input("Enter the first number: "))
number2 = float(input("Enter the second number: "))
result = find_maximum(number1, number2)
print(f"The maximum of the two numbers is: {result}")
```

```
Enter the first number:  6
Enter the second number:  9
```

```
The maximum of the two numbers is: 9.0
```

[5]:
```
# 4. Develop a program that calculates 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"


try:
    score = float(input("Enter the exam score: "))
    if score < 0 or score > 100:
        print("Please enter a valid score between 0 and 100.")
    else:
        grade = calculate_grade(score)
        print(f"The grade for the score {score} is: {grade}")
except ValueError:
    print("Invalid input. Please enter a numerical value.")
```

```
Enter the exam score:  88
```

```
The grade for the score 88.0 is: B
```

[6]:
```
# 5. Create a program that checks 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


try:
    year = int(input("Enter a year: "))
    if year <= 0:
        print("Please enter a valid positive year.")
    else:
        if is_leap_year(year):
            print(f"The year {year} is a leap year.")
        else:
            print(f"The year {year} is not a leap year.")
except ValueError:
    print("Invalid input. Please enter a numerical value.")
```

```
Enter a year:  2000
```

```
The year 2000 is a leap year.
```

[7]:
```python
# 6. Write a program to classify a triangle based on its sides' lengths.
def classify_triangle(a, b, c):
    if a + b <= c or a + c <= b or b + c <= a:
        return "Not a valid triangle."
    elif a == b == c:
        return "Equilateral triangle."
    elif a == b or b == c or a == c:
        return "Isosceles triangle."
    else:
        return "Scalene triangle."

try:
    a = float(input("Enter the length of the first side: "))
    b = float(input("Enter the length of the second side: "))
    c = float(input("Enter the length of the third side: "))

    if a <= 0 or b <= 0 or c <= 0:
        print("Please enter valid positive lengths for all sides.")
    else:
        result = classify_triangle(a, b, c)
        print(f"The triangle is: {result}")
except ValueError:
    print("Invalid input. Please enter numerical values for the sides.")
```

```
Enter the length of the first side:  6
Enter the length of the second side:  4
Enter the length of the third side:  2

The triangle is: Not a valid triangle.
```

[8]:
```python
# 7. Build a program that determines 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


try:
    number1 = float(input("Enter the first number: "))
    number2 = float(input("Enter the second number: "))
    number3 = float(input("Enter the third number: "))

    largest = find_largest(number1, number2, number3)
```

```
    print(f"The largest of the three numbers is: {largest}")
except ValueError:
    print("Invalid input. Please enter numerical values.")
```

```
Enter the first number:  5
Enter the second number:  7
Enter the third number:  4
```

```
The largest of the three numbers is: 7.0
```

[9]:
```
# 8. Develop a program that checks whether a character is a vowel or a␣
 ↪consonant.
def check_vowel_consonant(char):
    vowels = 'aeiouAEIOU'
    if len(char) != 1 or not char.isalpha():
        return "Invalid input. Please enter a single alphabetic character."
    elif char in vowels:
        return "The character is a vowel."
    else:
        return "The character is a consonant."


char = input("Enter a single character: ")
result = check_vowel_consonant(char)
print(result)
```

```
Enter a single character:  o
```

```
The character is a vowel.
```

[10]:
```
# 9. Create a program to calculate the total cost of a shopping cart based on␣
 ↪discounts.
def apply_discount(price, discount):
    """
    Calculate the price after discount.

    Parameters:
    price (float): Original price of the item.
    discount (float): Discount on the item (in percentage, e.g., 20 for 20%).

    Returns:
    float: Price after applying the discount.
    """
    return price * (1 - discount / 100)

def calculate_total_cost(cart):
    """
    Calculate the total cost of items in the shopping cart after discounts.
```

```
    Parameters:
    cart (list of dict): Shopping cart where each item is represented by a␣
 ↪dictionary with 'name', 'price', and 'discount'.

    Returns:
    float: Total cost after applying discounts.
    """
    total_cost = 0.0
    for item in cart:
        price_after_discount = apply_discount(item['price'], item['discount'])
        total_cost += price_after_discount
    return total_cost

shopping_cart = [
    {'name': 'Laptop', 'price': 1000.0, 'discount': 10},
    {'name': 'Headphones', 'price': 200.0, 'discount': 20},
    {'name': 'Mouse', 'price': 50.0, 'discount': 5},
    {'name': 'Keyboard', 'price': 100.0, 'discount': 15},
]

total_cost = calculate_total_cost(shopping_cart)

print(f'Total cost after discounts: ${total_cost:.2f}')
```

```
Total cost after discounts: $1192.50
```

[11]:
```
# 10. Write a program that checks if a given number is even or odd.
def check_even_odd(number):
    """
    Check if the given number is even or odd.

    Parameters:
    number (int): The number to check.

    Returns:
    str: "even" if the number is even, "odd" if the number is odd.
    """
    if number % 2 == 0:
        return "even"
    else:
        return "odd"

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

result = check_even_odd(number)
```

```
print(f"The number {number} is {result}.")
```

```
Enter a number:  45
```

```
The number 45 is odd.
```

[13]:
```
#Intermediate Level
# 11. Write a program that calculates the roots of a quadratic equation .
import math

def calculate_roots(a, b, c):
    """
    Calculate the roots of a quadratic equation of the form ax^2 + bx + c = 0.

    Parameters:
    a (float): Coefficient of x^2.
    b (float): Coefficient of x.
    c (float): Constant term.

    Returns:
    tuple: A tuple containing roots of the quadratic equation.
    """
    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  # Return a repeated root
    else:
        real_part = -b / (2*a)
        imaginary_part = math.sqrt(abs(discriminant)) / (2*a)
        root1 = complex(real_part, imaginary_part)
        root2 = complex(real_part, -imaginary_part)
        return root1, root2

if __name__ == "__main__":
    a = float(input("Enter the coefficient of x^2 (a): "))
    b = float(input("Enter the coefficient of x (b): "))
    c = float(input("Enter the constant term (c): "))

    roots = calculate_roots(a, b, c)

    if isinstance(roots[0], complex):
        print(f"The roots of the quadratic equation are complex:")
        print(f"Root 1: {roots[0]}")
```

```
        print(f"Root 2: {roots[1]}")
    else:
        print(f"The roots of the quadratic equation are real:")
        print(f"Root 1: {roots[0]}")
        print(f"Root 2: {roots[1]}")
```

```
Enter the coefficient of x^2 (a):  1
Enter the coefficient of x (b):  2
Enter the constant term (c):  3

The roots of the quadratic equation are complex:
Root 1: (-1+1.4142135623730951j)
Root 2: (-1-1.4142135623730951j)
```

[2]:
```
# 12. Create a program that determines the day of the week based on the day␣
 ↪number (1-7).
def get_day_of_week(day_number):
    days_of_week = {
        1: "Monday",
        2: "Tuesday",
        3: "Wednesday",
        4: "Thursday",
        5: "Friday",
        6: "Saturday",
        7: "Sunday"
    }

    if 1 <= day_number <= 7:
        return days_of_week[day_number]
    else:
        return "Invalid day number. Please enter a number between 1 and 7."

day_number = int(input("Enter the day number (1-7): "))
print(get_day_of_week(day_number))
```

```
Enter the day number (1-7):  5

Friday
```

[4]:
```
# 13. Develop a program that calculates the factorial of a given number using␣
 ↪recursion.
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

number = int(input("Enter a number to calculate its factorial: "))
```

```
result = factorial(number)
print(f"The factorial of {number} is {result}")
```

```
Enter a number to calculate its factorial:  4

The factorial of 4 is 24
```

[5]:
```
# 14. Write a program to find the largest among three numbers without using the␣
 ↪`max()` function.
def find_largest(a, b, c):
    if a >= b and a >= c:
        return a
    elif b >= a and b >= c:
        return b
    else:
        return c

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

largest = find_largest(num1, num2, num3)
print(f"The largest number among {num1}, {num2}, and {num3} is {largest}")
```

```
Enter the first number:  4
Enter the second number:  7
Enter the third number:  9

The largest number among 4.0, 7.0, and 9.0 is 9.0
```

[ ]:
```
# 15. Create a program that simulates a basic ATM transaction menu.
def atm_menu():
    balance = 1000

    while True:
        print("\nATM Transaction Menu")
        print("1. Check Balance")
        print("2. Deposit Money")
        print("3. Withdraw Money")
        print("4. Exit")

        choice = input("Enter your choice (1-4): ")

        if choice == '1':
            print(f"Your current balance is: ${balance:.2f}")

        elif choice == '2':
            deposit_amount = float(input("Enter amount to deposit: $"))
```

```
            if deposit_amount > 0:
                balance += deposit_amount
                print(f"${deposit_amount:.2f} has been deposited. New balance␣
 ↪is: ${balance:.2f}")
            else:
                print("Invalid amount. Please enter a positive number.")

        elif choice == '3':
            withdraw_amount = float(input("Enter amount to withdraw: $"))
            if 0 < withdraw_amount <= balance:
                balance -= withdraw_amount
                print(f"${withdraw_amount:.2f} has been withdrawn. New balance␣
 ↪is: ${balance:.2f}")
            elif withdraw_amount > balance:
                print("Insufficient funds. Please enter a lesser amount.")
            else:
                print("Invalid amount. Please enter a positive number.")

        elif choice == '4':
            print("Thank you for using the ATM. Goodbye!")
            break

        else:
            print("Invalid choice. Please enter a number between 1 and 4.")

atm_menu()
```

[ ]:
```
# 16. Build a program that checks if a given string is a palindrome or not.
def is_palindrome(s):
    cleaned_str = ''.join(char.lower() for char in s if char.isalnum())

    return cleaned_str == cleaned_str[::-1]

input_str = input("Enter a string to check if it is a palindrome: ")
if is_palindrome(input_str):
    print(f'"{input_str}" is a palindrome.')
else:
    print(f'"{input_str}" is not a palindrome.')
```

[1]:
```
#17. Write a program that calculates the average of a list of numbers,␣
 ↪excluding the smallest and largest values.
def average_excluding_extremes(numbers):
    if len(numbers) <= 2:
        return "List is too short to exclude extremes and calculate an average."

    sorted_numbers = sorted(numbers)
```

```
    truncated_list = sorted_numbers[1:-1]

    if len(truncated_list) == 0:
        return "No numbers left to average after excluding extremes."

    average = sum(truncated_list) / len(truncated_list)
    return average

numbers = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
result = average_excluding_extremes(numbers)
print(f"Average excluding extremes: {result}")
```

```
Average excluding extremes: 3.7777777777777777
```

[2]:
```
# 18. Develop a program that converts a given temperature from Celsius to␣
 ↪Fahrenheit.
def celsius_to_fahrenheit(celsius):
    """Convert Celsius to Fahrenheit."""
    fahrenheit = (celsius * 9/5) + 32
    return fahrenheit

try:
    celsius = float(input("Enter temperature in Celsius: "))
    fahrenheit = celsius_to_fahrenheit(celsius)
    print(f"{celsius}°C is equal to {fahrenheit}°F")
except ValueError:
    print("Please enter a valid number.")
```

```
Enter temperature in Celsius:  5.7
```

```
5.7°C is equal to 42.260000000000005°F
```

[5]:
```
# 19. Create a program that simulates a basic calculator for addition,␣
 ↪subtraction, multiplication, and division.
def add(x, y):
    return x + y

def subtract(x, y):
    return x - y

def multiply(x, y):
    return x * y

def divide(x, y):
    if y == 0:
        return "Error: Division by zero is not allowed."
    return x / y
```

```
def calculator():
    print("Welcome to the basic calculator!")
    print("Select operation:")
    print("1. Addition")
    print("2. Subtraction")
    print("3. Multiplication")
    print("4. Division")

    choice = input("Enter choice (1/2/3/4): ")

    if choice in ('1', '2', '3', '4'):
        try:
            num1 = float(input("Enter first number: "))
            num2 = float(input("Enter second number: "))
        except ValueError:
            print("Invalid input. Please enter numeric values.")
            return

        if choice == '1':
            print(f"{num1} + {num2} = {add(num1, num2)}")
        elif choice == '2':
            print(f"{num1} - {num2} = {subtract(num1, num2)}")
        elif choice == '3':
            print(f"{num1} * {num2} = {multiply(num1, num2)}")
        elif choice == '4':
            print(f"{num1} / {num2} = {divide(num1, num2)}")
    else:
        print("Invalid input. Please select a valid operation.")

calculator()
```

```
Welcome to the basic calculator!
Select operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division

Enter choice (1/2/3/4):  4
Enter first number:  6
Enter second number:  8

6.0 / 8.0 = 0.75
```

[6]:
```
# 20. Write a program that determines the roots of a cubic equation using the␣
 ↪Cardano formula.
import cmath

def cardano_roots(a, b, c, d):


    p = (3*a*c - b**2) / (3*a**2)
    q = (2*b**3 - 9*a*b*c + 27*a**2*d) / (27*a**3)

    discriminant = (q**2 / 4) + (p**3 / 27)

    if discriminant > 0:
        sqrt_discriminant = cmath.sqrt(discriminant)
        u = cmath.exp(cmath.log(-q / 2 + sqrt_discriminant) / 3)
        v = cmath.exp(cmath.log(-q / 2 - sqrt_discriminant) / 3)
        root1 = u + v
        root2 = -(u + v) / 2 + cmath.sqrt(3) * (u - v) / 2 * 1j
        root3 = -(u + v) / 2 - cmath.sqrt(3) * (u - v) / 2 * 1j
    elif discriminant == 0:
        u = cmath.exp(cmath.log(-q / 2) / 3)
        root1 = 2 * u
        root2 = -u
        root3 = -u
    else:
        theta = cmath.acos(-q / (2 * cmath.sqrt(-(p**3 / 27))))
        root1 = 2 * cmath.sqrt(-p / 3) * cmath.cos(theta / 3)
        root2 = 2 * cmath.sqrt(-p / 3) * cmath.cos((theta + 2 * cmath.pi) / 3)
        root3 = 2 * cmath.sqrt(-p / 3) * cmath.cos((theta + 4 * cmath.pi) / 3)

    root1 -= b / (3 * a)
    root2 -= b / (3 * a)
    root3 -= b / (3 * a)

    return root1, root2, root3

try:
    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 = cardano_roots(a, b, c, d)
    print(f"The roots of the cubic equation are: {roots[0]}, {roots[1]},␣
 ↪{roots[2]}")
except ValueError:
    print("Invalid input. Please enter valid numbers.")
```

```
Enter coefficient a:  6
Enter coefficient b:  8
Enter coefficient c:  9
Enter coefficient d:  3

The roots of the cubic equation are: (0.37809009998519394+0.4805630690803345j),
(-0.4395318907150778-0.00850879265417509j),
(-1.2718915426034494-0.47205427642615944j)
```

[7]:
```python
# Advanced Level
#  21. Create a program that calculates the income tax based on the user's␣
 ↪income and tax brackets.
def calculate_tax(income):
    brackets = [
        (10000, 0.00),
        (30000, 0.10),
        (70000, 0.20),
        (float('inf'), 0.30)
    ]

    tax = 0
    previous_bracket = 0

    for limit, rate in brackets:
        if income > limit:
            taxable_income = limit - previous_bracket
            tax += taxable_income * rate
            previous_bracket = limit
        else:
            taxable_income = income - previous_bracket
            tax += taxable_income * rate
            break

    return tax

try:
    income = float(input("Enter your income: "))

    if income < 0:
        print("Income cannot be negative.")
    else:
        tax_due = calculate_tax(income)
        print(f"The tax due on an income of ${income:.2f} is ${tax_due:.2f}")
except ValueError:
    print("Invalid input. Please enter a valid number.")
```

```
Enter your income:  5

The tax due on an income of $5.00 is $0.00
```

[9]:
```python
#22. Write a program that simulates a rock-paper-scissors game against the computer.
import random

def get_computer_choice():
    """Randomly choose rock, paper, or scissors for the computer."""
    choices = ['rock', 'paper', 'scissors']
    return random.choice(choices)

def determine_winner(user_choice, computer_choice):
    """Determine the winner of the game based on user and computer choices."""
    if user_choice == computer_choice:
        return "It's a tie!"

    if (user_choice == 'rock' and computer_choice == 'scissors') or \
       (user_choice == 'scissors' and computer_choice == 'paper') or \
       (user_choice == 'paper' and computer_choice == 'rock'):
        return "You win!"

    return "You lose!"

def play_game():
    """Play a single game of rock-paper-scissors."""
    print("Welcome to Rock-Paper-Scissors!")

    user_choice = input("Enter your choice (rock, paper, or scissors): ").lower()

    if user_choice not in ['rock', 'paper', 'scissors']:
        print("Invalid choice. Please choose rock, paper, or scissors.")
        return

    computer_choice = get_computer_choice()
    print(f"Computer chose: {computer_choice}")

    result = determine_winner(user_choice, computer_choice)
    print(result)

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

```
Welcome to Rock-Paper-Scissors!

Enter your choice (rock, paper, or scissors):  paper

Computer chose: scissors
You lose!
```

[11]:
```
#23.Develop a program that generates a random password based on user␣
 ↪preferences (length, complexity).
import random
import string

def generate_password(length, use_uppercase, use_digits, use_special):
    """Generate a random password based on user preferences."""

    lower_chars = string.ascii_lowercase
    upper_chars = string.ascii_uppercase if use_uppercase else ''
    digits = string.digits if use_digits else ''
    special_chars = string.punctuation if use_special else ''

    all_chars = lower_chars + upper_chars + digits + special_chars

    if not all_chars:
        raise ValueError("At least one character set must be selected.")

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

    return password

def get_user_preferences():
    """Get user preferences for password generation."""
    try:
        length = int(input("Enter the desired password length: "))
        if length <= 0:
            print("Password length must be a positive integer.")
            return None, None, None, None

        use_uppercase = input("Include uppercase letters? (yes/no): ").strip().
 ↪lower() == 'yes'
        use_digits = input("Include digits? (yes/no): ").strip().lower() ==␣
 ↪'yes'
        use_special = input("Include special characters? (yes/no): ").strip().
 ↪lower() == 'yes'

        return length, use_uppercase, use_digits, use_special
    except ValueError:
        print("Invalid input. Please enter a valid number.")
        return None, None, None, None

def main():
    print("Welcome to the Random Password Generator!")

    length, use_uppercase, use_digits, use_special = get_user_preferences()
```

```
    if length is not None:
        password = generate_password(length, use_uppercase, use_digits,␣
 ↪use_special)
        print(f"Your generated password is: {password}")

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

```
Welcome to the Random Password Generator!

Enter the desired password length:  6
Include uppercase letters? (yes/no):  no
Include digits? (yes/no):  no
Include special characters? (yes/no):  no

Your generated password is: flohns
```

[ ]:
```
# 24. Create a program that implements a simple text-based adventure game with␣
 ↪branching scenarios.
def intro():
    """Introduction to the game."""
    print("Welcome to the Adventure Game!")
    print("You find yourself in a dark forest. There are two paths in front of␣
 ↪you.")
    print("1. Take the left path.")
    print("2. Take the right path.")

    choice = input("> ")
    if choice == "1":
        left_path()
    elif choice == "2":
        right_path()
    else:
        print("Invalid choice. Please select 1 or 2.")
        intro()

def left_path():
    """Scenario for taking the left path."""
    print("You walk down the left path and find a peaceful meadow.")
    print("In the meadow, you see a shimmering lake and a small house.")
    print("1. Approach the lake.")
    print("2. Go to the house.")

    choice = input("> ")
    if choice == "1":
        lake()
    elif choice == "2":
        house()
```

```
    else:
        print("Invalid choice. Please select 1 or 2.")
        left_path()

def right_path():
    """Scenario for taking the right path."""
    print("You walk down the right path and encounter a large, mysterious cave.
 ↪")
    print("There is a faint light coming from inside the cave.")
    print("1. Enter the cave.")
    print("2. Walk away from the cave.")

    choice = input("> ")
    if choice == "1":
        cave()
    elif choice == "2":
        print("You decide to walk away from the cave. After some time, you find␣
 ↪your way back to the forest entrance.")
        intro()
    else:
        print("Invalid choice. Please select 1 or 2.")
        right_path()

def lake():
    """Scenario for approaching the lake."""
    print("At the lake, you see a boat floating gently on the water.")
    print("1. Take the boat across the lake.")
    print("2. Return to the forest path.")

    choice = input("> ")
    if choice == "1":
        print("You row across the lake and find a treasure chest.␣
 ↪Congratulations, you found a hidden treasure!")
    elif choice == "2":
        intro()
    else:
        print("Invalid choice. Please select 1 or 2.")
        lake()

def house():
    """Scenario for going to the house."""
    print("You approach the small house and find it surprisingly well-kept.")
    print("An old woman opens the door and invites you in.")
    print("1. Enter the house.")
    print("2. Politely decline and return to the forest path.")

    choice = input("> ")
```

```
    if choice == "1":
        print("Inside, the old woman offers you a magical potion. You drink it␣
 ↪and suddenly find yourself with incredible powers! You've won the game!")
    elif choice == "2":
        intro()
    else:
        print("Invalid choice. Please select 1 or 2.")
        house()

def cave():
    """Scenario for entering the cave."""
    print("You enter the cave and follow the faint light.")
    print("Deep inside, you discover an ancient library filled with mysterious␣
 ↪books.")
    print("1. Examine the books.")
    print("2. Leave the cave.")

    choice = input("> ")
    if choice == "1":
        print("You read a magical book and gain knowledge of ancient secrets.␣
 ↪You have unlocked the wisdom of the ages! You've won the game!")
    elif choice == "2":
        print("You exit the cave and find yourself back at the forest entrance.
 ↪")
        intro()
    else:
        print("Invalid choice. Please select 1 or 2.")
        cave()

intro()
```

```
Welcome to the Adventure Game!
You find yourself in a dark forest. There are two paths in front of you.
1. Take the left path.
2. Take the right path.

>  1

You walk down the left path and find a peaceful meadow.
In the meadow, you see a shimmering lake and a small house.
1. Approach the lake.
2. Go to the house.

>  4

Invalid choice. Please select 1 or 2.
You walk down the left path and find a peaceful meadow.
In the meadow, you see a shimmering lake and a small house.
1. Approach the lake.
```

2. Go to the house.

[1]:
```
#25. Write a Python script to determine the type of a character (uppercase,␣
 ↪lowercase, or special) using elif statements.
def character_type(char):
    if len(char) != 1:
        return "Input must be a single character."

    if char.isupper():
        return "Uppercase"
    elif char.islower():
        return "Lowercase"
    elif char.isalnum():
        return "Special character"
    else:
        return "Special character"

if __name__ == "__main__":
    char = input("Enter a single character: ")
    result = character_type(char)
    print(result)
```

```
Enter a single character:  g
```

```
Lowercase
```

[2]:
```
#26. Implement a program to calculate the discounted price based on different␣
 ↪purchase amounts using elif statements.
def calculate_discounted_price(purchase_amount):
    if purchase_amount >= 1000:
        discount_rate = 0.20
    elif purchase_amount >= 500:
        discount_rate = 0.15
    elif purchase_amount >= 200:
        discount_rate = 0.10
    elif purchase_amount >= 100:
        discount_rate = 0.05
    else:
        discount_rate = 0.00

    discount_amount = purchase_amount * discount_rate
    discounted_price = purchase_amount - discount_amount

    return discounted_price, discount_rate * 100

if __name__ == "__main__":
    try:
        purchase_amount = float(input("Enter the purchase amount: "))
```

```
        if purchase_amount < 0:
            print("Purchase amount cannot be negative.")
        else:
            discounted_price, discount_percentage =␣
 ↪calculate_discounted_price(purchase_amount)
            print(f"Original Price: ${purchase_amount:.2f}")
            print(f"Discount Percentage: {discount_percentage:.2f}%")
            print(f"Discounted Price: ${discounted_price:.2f}")
    except ValueError:
        print("Invalid input. Please enter a numeric value.")
```

```
Enter the purchase amount:  9000
```

```
Original Price: $9000.00
Discount Percentage: 20.00%
Discounted Price: $7200.00
```

[3]:
```
#27. Develop a program to calculate the electricity bill based on different␣
 ↪consumption slabs using elif statements.
def calculate_electricity_bill(consumption):
    if consumption <= 100:
        rate_per_unit = 0.50
    elif consumption <= 300:
        rate_per_unit = 0.75
    elif consumption <= 500:
        rate_per_unit = 1.00
    else:
        rate_per_unit = 1.50

    bill_amount = consumption * rate_per_unit

    return bill_amount

if __name__ == "__main__":
    try:
        consumption = float(input("Enter the electricity consumption in units:␣
 ↪"))
        if consumption < 0:
            print("Consumption cannot be negative.")
        else:
            bill_amount = calculate_electricity_bill(consumption)
            print(f"Electricity Consumption: {consumption:.2f} units")
            print(f"Total Bill Amount: ${bill_amount:.2f}")
    except ValueError:
        print("Invalid input. Please enter a numeric value.")
```

```
Enter the electricity consumption in units:  50
```

```
Electricity Consumption: 50.00 units
Total Bill Amount: $25.00
```

[5]:
```python
#28. Create a program to determine the type of quadrilateral based on its␣
 ↪angles and sides using elif statements.
def determine_quadrilateral_type(sides, angles):
    if len(sides) != 4 or len(angles) != 4:
        return "Invalid input. Please provide exactly 4 sides and 4 angles."

    if sum(angles) != 360:
        return "Invalid angles. The sum of the angles in a quadrilateral must␣
 ↪be 360 degrees."

    all_right_angles = all(angle == 90 for angle in angles)

    all_sides_equal = len(set(sides)) == 1

    opposite_sides_equal = (sides[0] == sides[2] and sides[1] == sides[3])

    if all_sides_equal and all_right_angles:
        return "Square"
    elif all_right_angles and opposite_sides_equal:
        return "Rectangle"
    elif all_sides_equal and not all_right_angles:
        return "Rhombus"
    elif opposite_sides_equal and (angles[0] == angles[2] and angles[1] ==␣
 ↪angles[3]):
        return "Parallelogram"
    elif (sides[0] == sides[1] and sides[2] == sides[3]) or (sides[0] ==␣
 ↪sides[2] and sides[1] == sides[3]):
        return "Trapezium"
    else:
        return "Other Quadrilateral"

if __name__ == "__main__":
    try:
        sides = list(map(float, input("Enter the four sides of the␣
 ↪quadrilateral separated by spaces: ").split()))
        angles = list(map(float, input("Enter the four angles of the␣
 ↪quadrilateral separated by spaces: ").split()))
        result = determine_quadrilateral_type(sides, angles)
        print(result)
    except ValueError:
        print("Invalid input. Please enter numeric values.")
```

```
Enter the four sides of the quadrilateral separated by spaces:  4
Enter the four angles of the quadrilateral separated by spaces:  4
```

```
Invalid input. Please provide exactly 4 sides and 4 angles.
```

[6]:
```python
# 29. Write a Python script to determine the season based on a user-provided␣
 ↪month using elif statements.
def determine_season(month):
    month = month.lower()

    if month in ['december', 'january', 'february']:
        return "Winter"
    elif month in ['march', 'april', 'may']:
        return "Spring"
    elif month in ['june', 'july', 'august']:
        return "Summer"
    elif month in ['september', 'october', 'november']:
        return "Fall (Autumn)"
    else:
        return "Invalid month. Please enter a valid month name."

if __name__ == "__main__":
    month = input("Enter the month: ")
    season = determine_season(month)
    print(season)
```

```
Enter the month:  december

Winter
```

[7]:
```python
#30. Implement a program to determine the type of a year (leap or common) and␣
 ↪month (30 or 31 days) using elif statements.
def is_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False

def days_in_month(month, year):
    month = month.lower()

    if month in ['january', 'march', 'may', 'july', 'august', 'october',␣
 ↪'december']:
        return 31
    elif month in ['april', 'june', 'september', 'november']:
        return 30
    elif month == 'february':
        if is_leap_year(year):
            return 29
        else:
            return 28
```

```
    else:
        return "Invalid month. Please enter a valid month name."

if __name__ == "__main__":
    try:
        year = int(input("Enter the year: "))
        month = input("Enter the month: ")

        if is_leap_year(year):
            print(f"{year} is a Leap Year.")
        else:
            print(f"{year} is a Common Year.")

        days = days_in_month(month, year)
        if isinstance(days, int):
            print(f"{month.capitalize()} has {days} days.")
        else:
            print(days)
    except ValueError:
        print("Invalid input. Please enter a numeric value for the year.")
```

```
Enter the year:  2001
Enter the month:  january
```

```
2001 is a Common Year.
January has 31 days.
```

[8]:
```
# Basic Level:
#  1. Write a Python program that checks if a given number is positive,␣
 ↪negative, or zero.
def check_number(number):
    if number > 0:
        return "The number is positive."
    elif number < 0:
        return "The number is negative."
    else:
        return "The number is zero."

if __name__ == "__main__":
    try:
        number = float(input("Enter a number: "))
        result = check_number(number)
        print(result)
    except ValueError:
        print("Invalid input. Please enter a numeric value.")
```

```
Enter a number:  4
```

```
The number is positive.
```

[9]:
```python
#2.Create a program to determine if a person is eligible to vote based on their␣
 ↪age.
def check_voting_eligibility(age):
    if age >= 18:
        return "You are eligible to vote."
    else:
        return "You are not eligible to vote."

if __name__ == "__main__":
    try:
        age = int(input("Enter your age: "))
        result = check_voting_eligibility(age)
        print(result)
    except ValueError:
        print("Invalid input. Please enter a numeric value.")
```

```
Enter your age:  21

You are eligible to vote.
```

[10]:
```python
# 3. Write a program to find the maximum of two given numbers using conditional␣
 ↪statements.
def find_maximum(num1, num2):
    if num1 > num2:
        return num1
    elif num2 > num1:
        return num2
    else:
        return "Both numbers are equal."

if __name__ == "__main__":
    try:
        num1 = float(input("Enter the first number: "))
        num2 = float(input("Enter the second number: "))
        result = find_maximum(num1, num2)
        print(f"The maximum of the two numbers is: {result}")
    except ValueError:
        print("Invalid input. Please enter numeric values.")
```

```
Enter the first number:  2
Enter the second number:  5

The maximum of the two numbers is: 5.0
```

[11]:
```python
#  4. Develop a program that calculates the grade of a student based on their␣
 ↪exam score.
```

```
def calculate_grade(score):
    if 90 <= score <= 100:
        return "A"
    elif 80 <= score < 90:
        return "B"
    elif 70 <= score < 80:
        return "C"
    elif 60 <= score < 70:
        return "D"
    elif 0 <= score < 60:
        return "F"
    else:
        return "Invalid score. Please enter a score between 0 and 100."

if __name__ == "__main__":
    try:
        score = float(input("Enter the exam score: "))
        grade = calculate_grade(score)
        print(f"The grade for a score of {score} is: {grade}")
    except ValueError:
        print("Invalid input. Please enter a numeric value.")
```

```
Enter the exam score:  80
```

```
The grade for a score of 80.0 is: B
```

[12]:
```
#  5. Create a program that checks 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

if __name__ == "__main__":
    try:
        year = int(input("Enter a year: "))
        if is_leap_year(year):
            print(f"{year} is a Leap Year.")
        else:
            print(f"{year} is not a Leap Year.")
    except ValueError:
        print("Invalid input. Please enter a numeric value.")
```

```
Enter a year:  2002
```

```
2002 is not a Leap Year.
```

[13]:
```python
#  6. Write a program to classify a triangle based on its sides' lengths.
def classify_triangle(a, b, c):
    if a + b > c and a + c > b and b + c > a:
        if a == b == c:
            return "Equilateral Triangle"
        elif a == b or b == c or a == c:
            return "Isosceles Triangle"
        else:
            return "Scalene Triangle"
    else:
        return "The provided sides do not form a valid triangle."

if __name__ == "__main__":
    try:
        a = float(input("Enter the length of the first side: "))
        b = float(input("Enter the length of the second side: "))
        c = float(input("Enter the length of the third side: "))
        result = classify_triangle(a, b, c)
        print(result)
    except ValueError:
        print("Invalid input. Please enter numeric values.")
```

```
Enter the length of the first side:  4
Enter the length of the second side:  4
Enter the length of the third side:  5
```

```
Isosceles Triangle
```

[14]:
```python
#  7. Build a program that determines the largest of three given numbers.
def find_largest_of_three(num1, num2, num3):
    if num1 >= num2 and num1 >= num3:
        return num1
    elif num2 >= num1 and num2 >= num3:
        return num2
    else:
        return num3

if __name__ == "__main__":
    try:
        num1 = float(input("Enter the first number: "))
        num2 = float(input("Enter the second number: "))
        num3 = float(input("Enter the third number: "))
        largest = find_largest_of_three(num1, num2, num3)
        print(f"The largest of the three numbers is: {largest}")
    except ValueError:
        print("Invalid input. Please enter numeric values.")
```

```
Enter the first number:  3
```

```
Enter the second number:  4
Enter the third number:  5

The largest of the three numbers is: 5.0
```

[15]:
```python
#  8. Develop a program that checks whether a character is a vowel or a␣
 ↪consonant.
def check_character(character):
    vowels = 'aeiouAEIOU'

    if character in vowels:
        return "The character is a vowel."
    elif character.isalpha():
        return "The character is a consonant."
    else:
        return "The character is not a letter."

if __name__ == "__main__":
    character = input("Enter a single character: ")

    if len(character) == 1:
        result = check_character(character)
        print(result)
    else:
        print("Invalid input. Please enter a single character.")
```

```
Enter a single character:  g

The character is a consonant.
```

[16]:
```python
#  9. Create a program to calculate the total cost of a shopping cart based on␣
 ↪discounts.
def calculate_total_cost(initial_cost):
    if initial_cost >= 500:
        discount_rate = 0.20
    elif initial_cost >= 200:
        discount_rate = 0.10
    elif initial_cost >= 100:
        discount_rate = 0.05
    else:
        discount_rate = 0.00

    discount_amount = initial_cost * discount_rate
    total_cost = initial_cost - discount_amount
    return total_cost, discount_amount

if __name__ == "__main__":
    try:
```

```
        initial_cost = float(input("Enter the initial cost of the shopping cart:
 ↪ "))
        total_cost, discount_amount = calculate_total_cost(initial_cost)
        print(f"Initial Cost: ${initial_cost:.2f}")
        print(f"Discount Amount: ${discount_amount:.2f}")
        print(f"Total Cost after Discount: ${total_cost:.2f}")
    except ValueError:
        print("Invalid input. Please enter a numeric value.")
```

```
Enter the initial cost of the shopping cart:  500
```

```
Initial Cost: $500.00
Discount Amount: $100.00
Total Cost after Discount: $400.00
```

[17]:
```
#  10. Write a program that checks if a given number is even or odd.
def check_even_or_odd(number):
    if number % 2 == 0:
        return "The number is even."
    else:
        return "The number is odd."

if __name__ == "__main__":
    try:
        number = int(input("Enter a number: "))
        result = check_even_or_odd(number)
        print(result)
    except ValueError:
        print("Invalid input. Please enter an integer value.")
```

```
Enter a number:  5
```

```
The number is odd.
```

[18]:
```
#  Intermediate Level:
#  11. Write a program that calculates the roots of a quadratic equation .
import cmath

def calculate_roots(a, b, c):
    discriminant = b**2 - 4*a*c

    root1 = (-b + cmath.sqrt(discriminant)) / (2 * a)
    root2 = (-b - cmath.sqrt(discriminant)) / (2 * a)

    return root1, root2

if __name__ == "__main__":
    try:
```

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

        if a == 0:
            print("Coefficient 'a' cannot be zero for a quadratic equation.")
        else:
            root1, root2 = calculate_roots(a, b, c)
            print(f"The roots of the quadratic equation are: {root1} and␣
 ↪{root2}")
    except ValueError:
        print("Invalid input. Please enter numeric values.")
```

```
Enter the coefficient a:  5
Enter the coefficient b:  6
Enter the coefficient c:  7

The roots of the quadratic equation are: (-0.6+1.0198039027185568j) and
(-0.6-1.0198039027185568j)
```

[19]:
```
# 12. Create a program that determines the day of the week based on the day␣
 ↪number (1-7).
def day_of_week(day_number):
    if day_number == 1:
        return "Monday"
    elif day_number == 2:
        return "Tuesday"
    elif day_number == 3:
        return "Wednesday"
    elif day_number == 4:
        return "Thursday"
    elif day_number == 5:
        return "Friday"
    elif day_number == 6:
        return "Saturday"
    elif day_number == 7:
        return "Sunday"
    else:
        return "Invalid day number. Please enter a number between 1 and 7."

if __name__ == "__main__":
    try:
        day_number = int(input("Enter a day number (1-7): "))
        result = day_of_week(day_number)
        print(result)
    except ValueError:
        print("Invalid input. Please enter an integer value.")
```

```
Enter a day number (1-7):  4

Thursday
```

[21]:
```python
#  13. Develop a program that calculates the factorial of a given number using␣
 ↪recursion.
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

if __name__ == "__main__":
    try:
        number = int(input("Enter a non-negative integer: "))

        if number < 0:
            print("Invalid input. Please enter a non-negative integer.")
        else:
            result = factorial(number)
            print(f"The factorial of {number} is: {result}")
    except ValueError:
        print("Invalid input. Please enter an integer value.")
```

```
Enter a non-negative integer:  4

The factorial of 4 is: 24
```

[22]:
```python
#  14. Write a program to find the largest among three numbers without using␣
 ↪the `max()` function.
def find_largest_of_three(num1, num2, num3):
    if num1 >= num2 and num1 >= num3:
        return num1
    elif num2 >= num1 and num2 >= num3:
        return num2
    else:
        return num3

if __name__ == "__main__":
    try:
        num1 = float(input("Enter the first number: "))
        num2 = float(input("Enter the second number: "))
        num3 = float(input("Enter the third number: "))

        largest = find_largest_of_three(num1, num2, num3)
        print(f"The largest number among {num1}, {num2}, and {num3} is:␣
 ↪{largest}")
    except ValueError:
```

```
        print("Invalid input. Please enter numeric values.")
```

```
Enter the first number:  5
Enter the second number:  7
Enter the third number:  8

The largest number among 5.0, 7.0, and 8.0 is: 8.0
```

[24]:
```
# 15. Create a program that simulates a basic ATM transaction menu.
def atm_menu():
    balance = 1000
    while True:
        print("\nATM Menu")
        print("1. Check Balance")
        print("2. Deposit Money")
        print("3. Withdraw Money")
        print("4. Exit")

        choice = input("Enter your choice (1-4): ")

        if choice == '1':
            print(f"Your current balance is: ${balance:.2f}")
        elif choice == '2':
            try:
                deposit_amount = float(input("Enter amount to deposit: "))
                if deposit_amount > 0:
                    balance += deposit_amount
                    print(f"Deposited ${deposit_amount:.2f}. New balance is:␣
↪${balance:.2f}")
                else:
                    print("Deposit amount must be positive.")
            except ValueError:
                print("Invalid input. Please enter a numeric value.")
        elif choice == '3':
            try:
                withdraw_amount = float(input("Enter amount to withdraw: "))
                if withdraw_amount > 0:
                    if withdraw_amount <= balance:
                        balance -= withdraw_amount
                        print(f"Withdrew ${withdraw_amount:.2f}. New balance is:
↪ ${balance:.2f}")
                    else:
                        print("Insufficient funds.")
                else:
                    print("Withdrawal amount must be positive.")
            except ValueError:
                print("Invalid input. Please enter a numeric value.")
```

```
        elif choice == '4':
            print("Exiting... Thank you for using the ATM.")
            break
        else:
            print("Invalid choice. Please select a valid option (1-4).")


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

```
ATM Menu
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit

Enter your choice (1-4):  2
Enter amount to deposit:  50000

Deposited $50000.00. New balance is: $51000.00

ATM Menu
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit

Enter your choice (1-4):  4

Exiting… Thank you for using the ATM.
```

[25]:
```
# 16. Build a program that checks if a given string is a palindrome or not.
import re

def is_palindrome(s):
    normalized_str = re.sub(r'[^a-zA-Z0-9]', '', s).lower()

    return normalized_str == normalized_str[::-1]

if __name__ == "__main__":
    input_string = input("Enter a string to check if it's a palindrome: ")

    if is_palindrome(input_string):
        print("The string is a palindrome.")
    else:
        print("The string is not a palindrome.")
```

```
Enter a string to check if it's a palindrome:  4
```

```
The string is a palindrome.
```

[27]:
```python
# 17. Write a program that calculates the average of a list of numbers, excluding the smallest and largest values.
def calculate_average_excluding_extremes(numbers):
    if len(numbers) <= 2:
        return "Not enough numbers to calculate average after excluding extremes."

    sorted_numbers = sorted(numbers)

    trimmed_numbers = sorted_numbers[1:-1]

    average = sum(trimmed_numbers) / len(trimmed_numbers)

    return average

if __name__ == "__main__":
    try:
        input_list = input("Enter a list of numbers separated by spaces: ")
        numbers = list(map(float, input_list.split()))

        result = calculate_average_excluding_extremes(numbers)

        if isinstance(result, str):
            print(result)
        else:
            print(f"The average of the numbers excluding the smallest and largest values is: {result:.2f}")

    except ValueError:
        print("Invalid input. Please enter numeric values separated by spaces.")
```

```
Enter a list of numbers separated by spaces:  3 5 7  7
The average of the numbers excluding the smallest and largest values is: 6.00
```

[28]:
```python
# 18. Develop a program that converts a given temperature from Celsius to Fahrenheit.
def celsius_to_fahrenheit(celsius):
    fahrenheit = (9/5) * celsius + 32
    return fahrenheit

if __name__ == "__main__":
    try:
        celsius_temp = float(input("Enter temperature in Celsius: "))

        fahrenheit_temp = celsius_to_fahrenheit(celsius_temp)
```

```
        print(f"{celsius_temp:.2f} Celsius is equal to {fahrenheit_temp:.2f}␣
↪Fahrenheit.")

    except ValueError:
        print("Invalid input. Please enter a numeric value.")
```

```
Enter temperature in Celsius:  45

45.00 Celsius is equal to 113.00 Fahrenheit.
```

[29]:
```
# 19. Create a program that simulates a basic calculator for addition,␣
↪subtraction, multiplication, and division.
def basic_calculator():
    print("Basic Calculator")
    print("Select operation:")
    print("1. Addition")
    print("2. Subtraction")
    print("3. Multiplication")
    print("4. Division")

    try:
        choice = input("Enter choice (1/2/3/4): ")

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

        if choice == '1':
            result = num1 + num2
            print(f"{num1} + {num2} = {result:.2f}")
        elif choice == '2':
            result = num1 - num2
            print(f"{num1} - {num2} = {result:.2f}")
        elif choice == '3':
            result = num1 * num2
            print(f"{num1} * {num2} = {result:.2f}")
        elif choice == '4':
            if num2 == 0:
                print("Error: Division by zero is not allowed.")
            else:
                result = num1 / num2
                print(f"{num1} / {num2} = {result:.2f}")
        else:
            print("Invalid input. Please select a valid operation (1/2/3/4).")

    except ValueError:
        print("Invalid input. Please enter numeric values.")
```

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

```
Basic Calculator
Select operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division

Enter choice (1/2/3/4):  2
Enter the first number:  4
Enter the second number:  5

4.0 - 5.0 = -1.00
```

[30]:
```
#  20. Write a program that determines the roots of a cubic equation using the␣
 ↪Cardano formula.
import cmath

def cardano_cubic_solver(a, b, c, d):
    if a == 0:
        raise ValueError("Coefficient 'a' cannot be zero for a cubic equation.")

    b /= a
    c /= a
    d /= a

    delta_0 = b**2 - 3*c
    delta_1 = 2*b**3 - 9*b*c + 27*d


    discriminant = (delta_1**2 - 4*delta_0**3) / 27

    C = cmath.sqrt((delta_1**2 - 4*delta_0**3) / 27) / 2
    theta = cmath.acos(delta_1 / (2 * C)) / 3

    roots = []
    for k in range(3):
        root = -b / 3 + 2 * cmath.sqrt(delta_0) * cmath.cos(theta + 2 * cmath.
 ↪pi * k / 3) / 3
        roots.append(root)

    return roots

if __name__ == "__main__":
```

```
    try:
        a = float(input("Enter coefficient a (non-zero): "))
        b = float(input("Enter coefficient b: "))
        c = float(input("Enter coefficient c: "))
        d = float(input("Enter coefficient d: "))

        roots = cardano_cubic_solver(a, b, c, d)

        print("The roots of the cubic equation are:")
        for i, root in enumerate(roots, start=1):
            print(f"Root {i}: {root:.4f}")

    except ValueError as e:
        print(f"Error: {e}")
```

```
Enter coefficient a (non-zero):  4
Enter coefficient b:  6
Enter coefficient c:  7
Enter coefficient d:  3

The roots of the cubic equation are:
Root 1: -0.5000+1.2263j
Root 2: -0.8575-0.6131j
Root 3: -0.1425-0.6131j
```

[31]:
```
#  Advanced Level:
#  21. Create a program that calculates the income tax based on the user's␣
 ↪income and tax brackets.
def calculate_income_tax(income):
    if income <= 0:
        return 0

    tax = 0

    if income <= 10000:
        tax = income * 0.10
    elif income <= 40000:
        tax = 10000 * 0.10 + (income - 10000) * 0.20
    elif income <= 100000:
        tax = 10000 * 0.10 + 30000 * 0.20 + (income - 40000) * 0.30
    else:
        tax = 10000 * 0.10 + 30000 * 0.20 + 60000 * 0.30 + (income - 100000) *␣
 ↪0.40

    return tax

if __name__ == "__main__":
```

```
    try:
        income = float(input("Enter your income: "))

        tax = calculate_income_tax(income)

        print(f"Your income tax is: ${tax:.2f}")

    except ValueError:
        print("Invalid input. Please enter a numeric value.")
```

```
Enter your income:  40000
```

```
Your income tax is: $7000.00
```

[32]:
```
# 22. Write a program that simulates a rock-paper-scissors game against the␣
↪computer.
import random

def get_computer_choice():
    choices = ['rock', 'paper', 'scissors']
    return random.choice(choices)

def determine_winner(user_choice, 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!"

def rock_paper_scissors_game():
    print("Rock, Paper, Scissors Game")
    print("Choose your option: rock, paper, or scissors")

    user_choice = input("Enter your choice: ").lower()

    if user_choice not in ['rock', 'paper', 'scissors']:
        print("Invalid choice. Please choose rock, paper, or scissors.")
        return

    computer_choice = get_computer_choice()

    result = determine_winner(user_choice, computer_choice)

    print(f"You chose: {user_choice}")
    print(f"The computer chose: {computer_choice}")
```

```
    print(result)

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

```
Rock, Paper, Scissors Game
Choose your option: rock, paper, or scissors

Enter your choice:  rock

You chose: rock
The computer chose: scissors
You win!
```

[34]:
```
#  23. Develop a program that generates a random password based on user␣
 ↪preferences (length, complexity).
import random
import string

def generate_password(length, use_uppercase, use_digits, use_special_chars):
    # Define character sets
    lower = string.ascii_lowercase
    upper = string.ascii_uppercase if use_uppercase else ''
    digits = string.digits if use_digits else ''
    special_chars = string.punctuation if use_special_chars else ''

    all_characters = lower + upper + digits + special_chars

    if not all_characters:
        raise ValueError("At least one character type must be selected.")

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

    return password

def password_generator():
    print("Password Generator")

    try:
        length = int(input("Enter desired password length: "))
        use_uppercase = input("Include uppercase letters? (yes/no): ").strip().
 ↪lower() == 'yes'
        use_digits = input("Include digits? (yes/no): ").strip().lower() ==␣
 ↪'yes'
        use_special_chars = input("Include special characters? (yes/no): ").
 ↪strip().lower() == 'yes'

        if length <= 0:
```

```
            print("Password length must be a positive integer.")
            return

        password = generate_password(length, use_uppercase, use_digits,␣
 ↪use_special_chars)

        print(f"Generated password: {password}")

    except ValueError as e:
        print(f"Invalid input: {e}")

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

```
Password Generator

Enter desired password length:  5
Include uppercase letters? (yes/no):   k d t u
Include digits? (yes/no):  k
Include special characters? (yes/no):  jjjj

Generated password: qiqbz
```

[ ]:
```
# 24. Create a program that implements a simple text-based adventure game with␣
 ↪branching scenarios.
def start_game():
    print("Welcome to 'Escape the Dungeon'!")
    print("You find yourself in a dark dungeon with two paths ahead of you.")
    print("Do you want to go 'left' or 'right'?")

    choice = input("> ").strip().lower()

    if choice == 'left':
        left_path()
    elif choice == 'right':
        right_path()
    else:
        print("Invalid choice. Please type 'left' or 'right'.")
        start_game()

def left_path():
    print("You head down the left path and find a treasure chest.")
    print("Do you want to 'open' the chest or 'leave' it alone?")

    choice = input("> ").strip().lower()

    if choice == 'open':
        print("You open the chest and find a key inside!")
```

```
        print("You can now 'go back' to the starting point or 'continue' to␣
↪explore further.")
        choice = input("> ").strip().lower()

        if choice == 'go back':
            start_game()
        elif choice == 'continue':
            print("You continue down the path and find a way out of the dungeon!
↪")
            print("Congratulations, you have escaped the dungeon!")
        else:
            print("Invalid choice. Please type 'go back' or 'continue'.")
            left_path()
    elif choice == 'leave':
        print("You decide to leave the chest and continue exploring.")
        print("You encounter a dead end and have to go back.")
        start_game()
    else:
        print("Invalid choice. Please type 'open' or 'leave'.")
        left_path()

def right_path():
    print("You take the right path and encounter a fierce dragon!")
    print("Do you want to 'fight' the dragon or 'run' away?")

    choice = input("> ").strip().lower()

    if choice == 'fight':
        print("You bravely fight the dragon but it is too strong.")
        print("Unfortunately, you are defeated and the game is over.")
    elif choice == 'run':
        print("You run away from the dragon and find yourself back at the␣
↪starting point.")
        start_game()
    else:
        print("Invalid choice. Please type 'fight' or 'run'.")
        right_path()

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

```
Welcome to 'Escape the Dungeon'!
You find yourself in a dark dungeon with two paths ahead of you.
Do you want to go 'left' or 'right'?

>  right

You take the right path and encounter a fierce dragon!
```

```
Do you want to 'fight' the dragon or 'run' away?

>  open

Invalid choice. Please type 'fight' or 'run'.
You take the right path and encounter a fierce dragon!
Do you want to 'fight' the dragon or 'run' away?

>  rum

Invalid choice. Please type 'fight' or 'run'.
You take the right path and encounter a fierce dragon!
Do you want to 'fight' the dragon or 'run' away?
```

[ ]:
```python
# 25.Build a program that solves a linear equation  for x, considering␣
 ↪different cases.
def solve_linear_equation(a, b, c):
    """Solves the linear equation ax + b = c for x.

    Args:
        a (float): Coefficient of x
        b (float): Constant term on the left side
        c (float): Constant term on the right side

    Returns:
        str: The solution to the equation"""
    if a == 0:
        if b == c:
            return "The equation has infinitely many solutions."
        else:
            return "The equation has no solution."
    else:
        # Solve for x
        x = (c - b) / a
        return f"The solution is x = {x}"

# Example usage:
a = float(input("Enter the coefficient a: "))
b = float(input("Enter the constant b: "))
c = float(input("Enter the constant c: "))

solution = solve_linear_equation(a, b, c)
print(solution)
```

[ ]:
```python
#  26. Write a program that simulates a basic quiz game with multiple-choice␣
 ↪questions and scoring.
def run_quiz():
    """
    Runs a basic multiple-choice quiz game.
```

```
    """
    questions = [
        {
            "question": "What is the capital of France?",
            "options": ["A. Paris", "B. London", "C. Rome", "D. Berlin"],
            "answer": "A"
        },
        {
            "question": "Which planet is known as the Red Planet?",
            "options": ["A. Earth", "B. Mars", "C. Jupiter", "D. Saturn"],
            "answer": "B"
        },
        {
            "question": "What is the largest ocean on Earth?",
            "options": ["A. Atlantic Ocean", "B. Indian Ocean", "C. Arctic␣
 ↪Ocean", "D. Pacific Ocean"],
            "answer": "D"
        },
        {
            "question": "Who wrote 'To Kill a Mockingbird'?",
            "options": ["A. Harper Lee", "B. J.K. Rowling", "C. Ernest␣
 ↪Hemingway", "D. Mark Twain"],
            "answer": "A"
        }
    ]

    score = 0

    print("Welcome to the Quiz Game!")
    print("Please answer the following questions:")

    for i, question in enumerate(questions):
        print(f"\nQuestion {i + 1}: {question['question']}")
        for option in question['options']:
            print(option)

        user_answer = input("Enter your answer (A, B, C, D): ").strip().upper()

        if user_answer == question['answer']:
            print("Correct!")
            score += 1
        else:
            print("Incorrect!")

    print(f"\nQuiz Over! Your final score is {score} out of {len(questions)}.")

if __name__ == "__main__":
```

```
    run_quiz()
```

```
Welcome to the Quiz Game!
Please answer the following questions:

Question 1: What is the capital of France?
A. Paris
B. London
C. Rome
D. Berlin

Enter your answer (A, B, C, D):  A

Correct!

Question 2: Which planet is known as the Red Planet?
A. Earth
B. Mars
C. Jupiter
D. Saturn
```

[1]:
```
#  27. Develop a program that determines whether a given year is a prime number␣
 ↪or not.
def is_prime(year):
    """Check if a given year is a prime number."""
    if year <= 1:
        return False
    if year <= 3:
        return True
    if year % 2 == 0 or year % 3 == 0:
        return False
    i = 5
    while i * i <= year:
        if year % i == 0 or year % (i + 2) == 0:
            return False
        i += 6
    return True

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

```
Enter a year:  2001

2001 is not a prime number.
```

[2]:
```
# 28. Create a program that sorts three numbers in ascending order using␣
 ↪conditional statements.
def sort_three_numbers(a, b, c):
    """Sort three numbers in ascending order and return them."""
    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

try:
    a = float(input("Enter the first number: "))
    b = float(input("Enter the second number: "))
    c = float(input("Enter the third number: "))

    sorted_numbers = sort_three_numbers(a, b, c)
    print(f"The numbers in ascending order are: {sorted_numbers[0]},␣
 ↪{sorted_numbers[1]}, {sorted_numbers[2]}")
except ValueError:
    print("Invalid input. Please enter numeric values.")
```

```
Enter the first number:  4
Enter the second number:  5
Enter the third number:  6
```

```
The numbers in ascending order are: 4.0, 5.0, 6.0
```

[3]:
```
#  29. Build a program that determines the roots of a quartic equation using␣
 ↪numerical methods.
import numpy as np

def find_quartic_roots(a, b, c, d, e):
    """Find the roots of the quartic equation ax^4 + bx^3 + cx^2 + dx + e = 0.
 ↪"""
    coefficients = [a, b, c, d, e]

    roots = np.roots(coefficients)

    return roots

try:
    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 = find_quartic_roots(a, b, c, d, e)
    print("The roots of the quartic equation are:")
    for root in roots:
        print(root)
except ValueError:
    print("Invalid input. Please enter numeric values.")
```

```
Enter coefficient a:  5
Enter coefficient b:  6
Enter coefficient c:  7
Enter coefficient d:  8
Enter coefficient e:  3

The roots of the quartic equation are:
(0.10736184830484519+1.0931238484241725j)
(0.10736184830484519-1.0931238484241725j)
(-0.7624290534726604+0j)
(-0.6522946431370306+0j)
```

[4]:
```
# 30. Write a program that calculates the BMI (Body Mass Index) and provides␣
 ↪health recommendations based on the user's input.
def calculate_bmi(weight, height):
    """Calculate BMI given weight (kg) and height (m)."""
    return weight / (height ** 2)

def health_recommendation(bmi):
    """Provide health recommendations based on BMI."""
    if bmi < 18.5:
        return "Underweight: You may need to gain weight. Consider consulting a␣
 ↪healthcare provider for advice."
    elif 18.5 <= bmi < 24.9:
        return "Normal weight: Keep up the good work! Maintain a balanced diet␣
 ↪and regular exercise."
    elif 25 <= bmi < 29.9:
        return "Overweight: You may want to consider losing weight. A healthy␣
 ↪diet and regular exercise can help."
    else:
        return "Obesity: It is recommended to consult a healthcare provider for␣
 ↪personalized advice on weight management."

try:
    weight = float(input("Enter your weight (kg): "))
    height = float(input("Enter your height (m): "))

    if weight <= 0 or height <= 0:
```

```
        raise ValueError("Weight and height must be positive numbers.")

    bmi = calculate_bmi(weight, height)
    print(f"Your BMI is: {bmi:.2f}")
    print(health_recommendation(bmi))

except ValueError as e:
    print(f"Invalid input: {e}. Please enter valid numeric values for weight␣
 ↪and height.")
```

```
Enter your weight (kg):  23
Enter your height (m):  4

Your BMI is: 1.44
Underweight: You may need to gain weight. Consider consulting a healthcare
provider for advice.
```

[5]:
```
# Challenge Level:
# 31. Create a program that validates a password based on complexity rules␣
 ↪(length, characters, etc.).
import re

def validate_password(password):
    """Validate the password based on complexity rules."""
    min_length = 8
    special_characters = r'[!@#$%^&*(),.?":{}|<>]'

    if len(password) < min_length:
        return "Password must be at least 8 characters long."

    if not re.search(r'[A-Z]', password):
        return "Password must include at least one uppercase letter."

    if not re.search(r'[a-z]', password):
        return "Password must include at least one lowercase letter."

    if not re.search(r'\d', password):
        return "Password must include at least one digit."

    if not re.search(special_characters, password):
        return "Password must include at least one special character."

    return "Password is valid."

password = input("Enter your password: ")
validation_result = validate_password(password)
print(validation_result)
```

```
Enter your password:  pass@123
Password must include at least one uppercase letter.
```

[7]:
```python
#  32. Develop a program that performs matrix addition and subtraction based on user input.
def get_matrix_input(rows, cols, matrix_name):
    """Get matrix elements from the user."""
    matrix = []
    print(f"Enter the elements for {matrix_name} matrix:")
    for i in range(rows):
        row = input(f"Row {i + 1} (space-separated values): ").split()
        if len(row) != cols:
            raise ValueError(f"Each row must have exactly {cols} values.")
        matrix.append([int(x) for x in row])
    return matrix

def print_matrix(matrix):
    """Print the matrix."""
    for row in matrix:
        print(" ".join(map(str, row)))

def add_matrices(matrix1, matrix2):
    """Add two matrices."""
    return [[matrix1[i][j] + matrix2[i][j] for j in range(len(matrix1[0]))] for i in range(len(matrix1))]

def subtract_matrices(matrix1, matrix2):
    """Subtract matrix2 from matrix1."""
    return [[matrix1[i][j] - matrix2[i][j] for j in range(len(matrix1[0]))] for i in range(len(matrix1))]

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

    print("\nMatrix A:")
    matrix_a = get_matrix_input(rows, cols, "Matrix A")

    print("\nMatrix B:")
    matrix_b = get_matrix_input(rows, cols, "Matrix B")

    print("\nMatrix A + Matrix B:")
    added_matrix = add_matrices(matrix_a, matrix_b)
    print_matrix(added_matrix)

    print("\nMatrix A - Matrix B:")
    subtracted_matrix = subtract_matrices(matrix_a, matrix_b)
```

```
    print_matrix(subtracted_matrix)

except ValueError as e:
    print(f"Invalid input: {e}. Please enter numeric values and ensure matrices␣
 ↪have the correct dimensions.")
```

```
Enter the number of rows for the matrices:  3
Enter the number of columns for the matrices:  3


Matrix A:
Enter the elements for Matrix A matrix:

Row 1 (space-separated values):  2 3 4
Row 2 (space-separated values):  3 4 5
Row 3 (space-separated values):  1 2 3


Matrix B:
Enter the elements for Matrix B matrix:

Row 1 (space-separated values):  1 2 3
Row 2 (space-separated values):  4 5 6
Row 3 (space-separated values):  6 7 8


Matrix A + Matrix B:
3 5 7
7 9 11
7 9 11

Matrix A - Matrix B:
1 1 1
-1 -1 -1
-5 -5 -5
```

[8]:
```
# 33. Write a program that calculates the greatest common divisor (GCD) of two␣
 ↪numbers using the Euclidean algorithm.
def gcd(a, b):
    """Calculate the Greatest Common Divisor (GCD) using the Euclidean␣
 ↪algorithm."""
    while b != 0:
        a, b = b, a % b
    return a

try:
    num1 = int(input("Enter the first number: "))
    num2 = int(input("Enter the second number: "))
```

```
    if num1 < 0 or num2 < 0:
        raise ValueError("Both numbers must be non-negative.")

    result = gcd(num1, num2)
    print(f"The Greatest Common Divisor (GCD) of {num1} and {num2} is:␣
 ↪{result}")

except ValueError as e:
    print(f"Invalid input: {e}. Please enter non-negative integers.")
```

```
Enter the first number:  3
Enter the second number:  5

The Greatest Common Divisor (GCD) of 3 and 5 is: 1
```

[9]:
```
# 34. Build a program that performs matrix multiplication using nested loops␣
 ↪and conditional statements.
def get_matrix_input(rows, cols, matrix_name):
    """Get matrix elements from the user."""
    matrix = []
    print(f"Enter the elements for {matrix_name} matrix:")
    for i in range(rows):
        row = input(f"Row {i + 1} (space-separated values): ").split()
        if len(row) != cols:
            raise ValueError(f"Each row must have exactly {cols} values.")
        matrix.append([int(x) for x in row])
    return matrix

def print_matrix(matrix):
    """Print the matrix."""
    for row in matrix:
        print(" ".join(map(str, row)))

def multiply_matrices(matrix1, matrix2):
    """Multiply two matrices."""
    rows1 = len(matrix1)
    cols1 = len(matrix1[0])
    rows2 = len(matrix2)
    cols2 = len(matrix2[0])

    if cols1 != rows2:
        raise ValueError("Number of columns in the first matrix must be equal␣
 ↪to the number of rows in the second matrix.")

    result = [[0 for _ in range(cols2)] for _ in range(rows1)]

    for i in range(rows1):
```

```
        for j in range(cols2):
            for k in range(cols1):
                result[i][j] += matrix1[i][k] * matrix2[k][j]

    return result

try:
    rows1 = int(input("Enter the number of rows for Matrix A: "))
    cols1 = int(input("Enter the number of columns for Matrix A: "))
    matrix_a = get_matrix_input(rows1, cols1, "Matrix A")

    rows2 = int(input("Enter the number of rows for Matrix B: "))
    cols2 = int(input("Enter the number of columns for Matrix B: "))
    matrix_b = get_matrix_input(rows2, cols2, "Matrix B")

    result = multiply_matrices(matrix_a, matrix_b)

    print("\nMatrix A:")
    print_matrix(matrix_a)

    print("\nMatrix B:")
    print_matrix(matrix_b)

    print("\nResult of Matrix A * Matrix B:")
    print_matrix(result)

except ValueError as e:
    print(f"Invalid input: {e}. Please ensure that matrices have the correct␣
 ↪dimensions and contain numeric values.")
```

```
Enter the number of rows for Matrix A:  3
Enter the number of columns for Matrix A:  3

Enter the elements for Matrix A matrix:

Row 1 (space-separated values):  3 4 5
Row 2 (space-separated values):  4 6 7
Row 3 (space-separated values):  1 2 3
Enter the number of rows for Matrix B:  3
Enter the number of columns for Matrix B:  3

Enter the elements for Matrix B matrix:

Row 1 (space-separated values):   6 7 8
Row 2 (space-separated values):   1 3 5
Row 3 (space-separated values):  6 3 6


Matrix A:
3 4 5
```

```
4 6 7
1 2 3

Matrix B:
6 7 8
1 3 5
6 3 6

Result of Matrix A * Matrix B:
52 48 74
72 67 104
26 22 36
```

[ ]:
```python
#  35. Create a program that simulates a basic text-based tic-tac-toe game␣
 ↪against the computer.
import random

def print_board(board):
    """Print the game board."""
    print("\n".join([" | ".join(row) for row in board]))
    print()

def check_win(board, player):
    """Check if the current player has won."""
    for i in range(3):
        if all([cell == player for cell in board[i]]) or all([board[j][i] ==␣
 ↪player for j in range(3)]):
            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 check_draw(board):
    """Check if the board is full and it's a draw."""
    return all([cell in ['X', 'O'] for row in board for cell in row])

def get_empty_positions(board):
    """Return a list of empty positions on the board."""
    return [(r, c) for r in range(3) for c in range(3) if board[r][c] == ' ']

def player_move(board):
    """Get and validate the player's move."""
    while True:
        try:
```

```
            row, col = map(int, input("Enter your move (row and column numbers,␣
 ↪0-2) separated by space: ").split())
            if board[row][col] == ' ':
                board[row][col] = 'X'
                break
            else:
                print("The cell is already occupied. Try again.")
        except (ValueError, IndexError):
            print("Invalid move. Please enter row and column numbers between 0␣
 ↪and 2.")

def computer_move(board):
    """Make a random move for the computer."""
    empty_positions = get_empty_positions(board)
    row, col = random.choice(empty_positions)
    board[row][col] = 'O'

def play_game():
    """Main function to play the Tic-Tac-Toe game."""
    board = [[' ' for _ in range(3)] for _ in range(3)]
    print("Welcome to Tic-Tac-Toe!")
    print_board(board)

    while True:
        player_move(board)
        print_board(board)
        if check_win(board, 'X'):
            print("Congratulations! You win!")
            break
        if check_draw(board):
            print("It's a draw!")
            break

        computer_move(board)
        print_board(board)
        if check_win(board, 'O'):
            print("Computer wins! Better luck next time.")
            break
        if check_draw(board):
            print("It's a draw!")
            break

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

```
Welcome to Tic-Tac-Toe!
  |   |
```

```
  |   |
  |   |


Enter your move (row and column numbers, 0-2) separated by space:  1
Invalid move. Please enter row and column numbers between 0 and 2.
Enter your move (row and column numbers, 0-2) separated by space:  1 2
  |   |
  |   | X
  |   |

  | O |
  |   | X
  |   |

Enter your move (row and column numbers, 0-2) separated by space:  1 2 3
Invalid move. Please enter row and column numbers between 0 and 2.
```

[3]:
```python
# 36. Write a program that generates Fibonacci numbers up to a specified term␣
 ↪using iterative methods.
def generate_fibonacci(n):
    """Generate Fibonacci numbers up to the nth term using iterative methods."""
    if n <= 0:
        return []
    elif n == 1:
        return [0]

    fibonacci_sequence = [0, 1]
    while len(fibonacci_sequence) < n:
        next_term = fibonacci_sequence[-1] + fibonacci_sequence[-2]
        fibonacci_sequence.append(next_term)

    return fibonacci_sequence

try:
    num_terms = int(input("Enter the number of terms to generate in the␣
 ↪Fibonacci sequence: "))
    if num_terms < 0:
        raise ValueError("Number of terms must be a non-negative integer.")

    fibonacci_numbers = generate_fibonacci(num_terms)
    print("Fibonacci sequence:")
    print(fibonacci_numbers)

except ValueError as e:
```

```
    print(f"Invalid input: {e}. Please enter a non-negative integer.")
```

```
Enter the number of terms to generate in the Fibonacci sequence:  3

Fibonacci sequence:
[0, 1, 1]
```

[1]:
```
# 37. Develop a program that calculates the nth term of the Fibonacci sequence␣
↪using memoization.
def fibonacci_memo(n, memo={}):
    """Calculate the nth Fibonacci number using memoization."""
    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]

try:
    n = int(input("Enter the term number to find in the Fibonacci sequence: "))
    if n < 0:
        raise ValueError("Term number must be a non-negative integer.")

    result = fibonacci_memo(n)
    print(f"The {n}th term of the Fibonacci sequence is: {result}")

except ValueError as e:
    print(f"Invalid input: {e}. Please enter a non-negative integer.")
```

```
Enter the term number to find in the Fibonacci sequence:  3

The 3th term of the Fibonacci sequence is: 2
```

[1]:
```
# 38. Create a program that generates a calendar for a given month and year␣
↪using conditional statements.
import calendar

def is_leap_year(year):
    """Check if the given year is a leap year."""
    return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)

def get_number_of_days(month, year):
    """Return the number of days in a given month of a given year."""
    if month in [1, 3, 5, 7, 8, 10, 12]:
        return 31
    elif month in [4, 6, 9, 11]:
```

```
        return 30
    elif month == 2:
        return 29 if is_leap_year(year) else 28
    else:
        raise ValueError("Invalid month")

def print_calendar(month, year):
    """Print the calendar for the given month and year."""
    try:
        print(f"\n{calendar.month_name[month]} {year}")
        print("Su Mo Tu We Th Fr Sa")

        first_day = calendar.weekday(year, month, 1)
        num_days = get_number_of_days(month, year)

        print("   " * first_day, end="")

        for day in range(1, num_days + 1):
            print(f"{day:2} ", end="")
            if (first_day + day) % 7 == 0:
                print()

        print()

    except ValueError as e:
        print(e)

def main():
    try:
        year = int(input("Enter year (e.g., 2024): "))
        month = int(input("Enter month (1-12): "))

        if month < 1 or month > 12:
            raise ValueError("Month must be between 1 and 12.")

        print_calendar(month, year)

    except ValueError as e:
        print(f"Invalid input: {e}")

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

```
Enter year (e.g., 2024):  2024
Enter month (1-12):  5


May 2024
```

```
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

[ ]:
```python
# 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."""
    score = sum(hand)
    while 11 in hand and score > 21:
        hand.remove(11)
        hand.append(1)
        score = sum(hand)
    return score

def print_hand(name, hand):
    """Prints the name and hand of the player or dealer."""
    print(f"{name}'s hand: {hand}, score: {calculate_score(hand)}")

def blackjack():
    """Simulates a basic text-based Blackjack game."""
    print("Welcome to Blackjack!")

    player_hand = [deal_card(), deal_card()]
    dealer_hand = [deal_card(), deal_card()]

    game_over = False
    while not game_over:
        print_hand("Player", player_hand)
        if calculate_score(player_hand) == 21:
            print("Blackjack! You win!")
            return
        elif calculate_score(player_hand) > 21:
            print("You went over 21. You lose!")
            return
```

```
        action = input("Type 'hit' to get another card or 'stand' to pass: ").
↪lower()
        if action == 'hit':
            player_hand.append(deal_card())
        elif action == 'stand':
            game_over = True
        else:
            print("Invalid input. Please type 'hit' or 'stand'.")

    while calculate_score(dealer_hand) < 17:
        dealer_hand.append(deal_card())

    print_hand("Dealer", dealer_hand)

    player_score = calculate_score(player_hand)
    dealer_score = calculate_score(dealer_hand)

    if dealer_score > 21:
        print("Dealer went over 21. You win!")
    elif player_score > dealer_score:
        print("You win!")
    elif player_score < dealer_score:
        print("Dealer wins!")
    else:
        print("It's a tie!")

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

```
Welcome to Blackjack!
Player's hand: [8, 4], score: 12

Type 'hit' to get another card or 'stand' to pass:  hit

Player's hand: [8, 4, 11], score: 13

Type 'hit' to get another card or 'stand' to pass:  4

Invalid input. Please type 'hit' or 'stand'.
Player's hand: [8, 4, 1], score: 13

Type 'hit' to get another card or 'stand' to pass:  12

Invalid input. Please type 'hit' or 'stand'.
Player's hand: [8, 4, 1], score: 13
```

[1]:
```
# 40. Write a program that generates the prime factors of a given number using␣
↪trial division.
def prime_factors(n):
    """Returns a list of the prime factors of the given number n."""
```

```
    factors = []

    while n % 2 == 0:
        factors.append(2)
        n //= 2

    while n % 3 == 0:
        factors.append(3)
        n //= 3

    i = 5
    while i * i <= n:
        while n % i == 0:
            factors.append(i)
            n //= i
        while n % (i + 2) == 0:
            factors.append(i + 2)
            n //= (i + 2)
        i += 6

    if n > 3:
        factors.append(n)

    return factors

def main():
    try:
        number = int(input("Enter a number to find its prime factors: "))
        if number < 1:
            print("Please enter a positive integer.")
            return

        factors = prime_factors(number)
        print(f"Prime factors of {number} are: {factors}")

    except ValueError:
        print("Invalid input. Please enter a valid integer.")

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

```
Enter a number to find its prime factors:  2
Prime factors of 2 are: [2]
```

[ ]:
```
#
```