03_September_Assignment_Python_Control_Flow

September 8, 2023

[ ]:
```
#Basic If-Else Statements:

#1. Write a Python program to check if a given number is positive or negative.

# Input: Get a number from the user
number = float(input("Enter a number: "))

# Check if the number is positive, negative, or zero
if number > 0:
    print("The number is positive.")
elif number < 0:
    print("The number is negative.")
else:
    print("The number is zero.")
```

[2]:
```
#2. Create a program that determines if a person is eligible to vote based on␣
 ↪their age.

# Input: Get the person's age from the user
age = int(input("Enter your age: "))

# Check if the person is eligible to vote
voting_age = 18

if age >= voting_age:
    print("You are eligible to vote.")
else:
    print("You are not eligible to vote yet.")
```

```
Enter your age:  45
```

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

[3]:
```
#3. Develop a program to find the maximum of two numbers using if-else␣
 ↪statements.

 # Input: Get two numbers from the user
num1 = float(input("Enter the first number: "))
```

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

# Find the maximum of the two numbers using if-else statements
if num1 > num2:
    maximum = num1
else:
    maximum = num2

# Print the maximum number
print("The maximum number is:", maximum)
```

```
Enter the first number:  4
Enter the second number:  3
```

```
The maximum number is: 4.0
```

[4]:
```
#4. Write a Python script to classify a given year as a leap year or not.

# Input: Get the year from the user
year = int(input("Enter a year: "))

# Check if the year is a leap year using if-else statements
if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    print(year, "is a leap year.")
else:
    print(year, "is not a leap year.")
```

```
Enter a year:  33
```

```
33 is not a leap year.
```

[5]:
```
#5. Create a program that checks whether a character is a vowel or a consonant.

# Input: Get a character from the user
char = input("Enter a character: ")

# Convert the character to lowercase to handle both uppercase and lowercase␣
 ↪input
char = char.lower()

# Check if the character is a vowel or a consonant using if-else statements
if char.isalpha() and len(char) == 1:
    if char in "aeiou":
        print(char, "is a vowel.")
    else:
        print(char, "is a consonant.")
else:
    print("Invalid input. Please enter a single alphabet character.")
```

```
Enter a character:  4

Invalid input. Please enter a single alphabet character.
```

[6]:
```python
#6. Implement a program to determine whether a given number is even or odd.

# Input: Get a number from the user
number = int(input("Enter a number: "))

# Check if the number is even or odd using an if-else statement
if number % 2 == 0:
    print(number, "is even.")
else:
    print(number, "is odd.")
```

```
Enter a number:  4

4 is even.
```

[7]:
```python
#7. Write a Python function to calculate the absolute value of a number without␣
 ↪using the `abs()` function.

def absolute_value(number):
    if number < 0:
        return -number
    else:
        return number

# Test the function
num = float(input("Enter a number: "))
result = absolute_value(num)
print("Absolute value:", result)
```

```
Enter a number:  4

Absolute value: 4.0
```

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

# Input: Get three numbers from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))

# Determine the largest number using if-else statements
if num1 >= num2 and num1 >= num3:
    largest = num1
elif num2 >= num1 and num2 >= num3:
```

```
    largest = num2
else:
    largest = num3

# Print the largest number
print("The largest number is:", largest)
```

```
Enter the first number:  4
Enter the second number:  4
Enter the third number:  3

The largest number is: 4.0
```

[9]:
```
#9. Create a program that checks if a given string is a palindrome.

# Input: Get a string from the user
string = input("Enter a string: ")

# Remove spaces and convert the string to lowercase for case-insensitive␣
 ↪comparison
cleaned_string = string.replace(" ", "").lower()

# Check if the cleaned string is a palindrome using if-else statements
if cleaned_string == cleaned_string[::-1]:
    print(string, "is a palindrome.")
else:
    print(string, "is not a palindrome.")
```

```
Enter a string:  My name is ganesh

My name is ganesh is not a palindrome.
```

[10]:
```
#10. Write a Python program to calculate the grade based on a student's score.

# Input: Get the student's score from the user
score = float(input("Enter the student's score (0-100): "))

# Check the score and calculate the grade using if-elif-else statements
if 0 <= score <= 100:
    if score >= 90:
        grade = 'A'
    elif score >= 80:
        grade = 'B'
    elif score >= 70:
        grade = 'C'
    elif score >= 60:
        grade = 'D'
```

```
    else:
        grade = 'F'

    print("The student's grade is:", grade)
else:
    print("Invalid score. Please enter a score between 0 and 100.")
```

```
Enter the student's score (0-100):  45

The student's grade is: F
```

[11]:
```
#Nested If-Else Statements:
#11. Write a program to find the largest among three numbers using nested␣
 ↪if-else statements.

 # Input: Get three numbers from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))

# Find the largest number using nested if-else statements
if num1 >= num2:
    if num1 >= num3:
        largest = num1
    else:
        largest = num3
else:
    if num2 >= num3:
        largest = num2
    else:
        largest = num3

# Print the largest number
print("The largest number is:", largest)
```

```
Enter the first number:  3
Enter the second number:  5
Enter the third number:  2

The largest number is: 5.0
```

[12]:
```
#12. Implement a program to determine if a triangle is equilateral, isosceles,␣
 ↪or scalene.

# Input: Get the lengths of the three sides of the triangle from the user
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: "))
```

```
# Check the type of triangle using if-elif-else statements
if side1 == side2 == side3:
    triangle_type = "Equilateral"
elif side1 == side2 or side1 == side3 or side2 == side3:
    triangle_type = "Isosceles"
else:
    triangle_type = "Scalene"

# Print the type of triangle
print("The triangle is:", triangle_type)
```

```
Enter the length of the first side:  12
Enter the length of the second side:  34
Enter the length of the third side:  24

The triangle is: Scalene
```

[13]:
```
#13. Develop a program that checks if a year is a leap year and also if it is a␣
 ↪century year.

# Input: Get the year from the user
year = int(input("Enter a year: "))

# Check if the year is a leap year and if it's a century year using if-else␣
 ↪statements
if year % 4 == 0:
    if year % 100 == 0:
        if year % 400 == 0:
            leap_year = True
            century_year = True
        else:
            leap_year = False
            century_year = True
    else:
        leap_year = True
        century_year = False
else:
    leap_year = False
    century_year = False

# Print the results
if leap_year:
    print(year, "is a leap year.")
else:
    print(year, "is not a leap year.")
```

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

```
Enter a year:  202
```

```
202 is not a leap year.
202 is not a century year.
```

[14]:
```
#14. Write a Python script to determine if a number is positive, negative, or␣
 ↪zero.

# Input: Get a number from the user
number = float(input("Enter a number: "))

# Determine if the number is positive, negative, or zero using if-elif-else␣
 ↪statements
if number > 0:
    print("The number is positive.")
elif number < 0:
    print("The number is negative.")
else:
    print("The number is zero.")
```

```
Enter a number:  45
```

```
The number is positive.
```

[15]:
```
#15. Create a program to check if a person is a teenager (between 13 and 19␣
 ↪years old).

# Input: Get the person's age from the user
age = int(input("Enter your age: "))

# Check if the person is a teenager using if-else statements
if 13 <= age <= 19:
    print("You are a teenager.")
else:
    print("You are not a teenager.")
```

```
Enter your age:  45
```

```
You are not a teenager.
```

[16]:
```
#16. Develop a program that determines the type of angle based on its measure␣
 ↪(acute, obtuse, or right).

# Input: Get the measure of the angle from the user
```

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

# Determine the type of angle using if-elif-else statements
if angle_measure > 0 and angle_measure < 90:
    angle_type = "Acute"
elif angle_measure == 90:
    angle_type = "Right"
elif angle_measure > 90 and angle_measure < 180:
    angle_type = "Obtuse"
else:
    angle_type = "Invalid"

# Print the type of angle
if angle_type != "Invalid":
    print(f"The angle is {angle_type}.")
else:
    print("Invalid angle measure. Please enter a valid angle measure.")
```

```
Enter the measure of the angle in degrees:  34
```

```
The angle is Acute.
```

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

import math

# Input: Get the coefficients a, b, and c from the user
a = float(input("Enter the coefficient 'a': "))
b = float(input("Enter the coefficient 'b': "))
c = float(input("Enter the coefficient 'c': "))

# Calculate the discriminant
discriminant = b**2 - 4*a*c

# Check the discriminant to determine the number and type of roots
if discriminant > 0:
    root1 = (-b + math.sqrt(discriminant)) / (2*a)
    root2 = (-b - math.sqrt(discriminant)) / (2*a)
    root_type = "Two real and distinct roots"
elif discriminant == 0:
    root1 = root2 = -b / (2*a)
    root_type = "One real and 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)
    root_type = "Two complex roots"
```

```
# Print the roots and their type
print("Root 1:", root1)
print("Root 2:", root2)
print("Root Type:", root_type)
```

```
Enter the coefficient 'a':  3
Enter the coefficient 'b':  4
Enter the coefficient 'c':  6

Root 1: (-0.6666666666666666+1.247219128924647j)
Root 2: (-0.6666666666666666-1.247219128924647j)
Root Type: Two complex roots
```

[18]:
```
#19. Create a program that determines if a year is a leap year and also if it␣
 ↪is evenly divisible by 400.

# Input: Get the year from the user
year = int(input("Enter a year: "))

# Check if the year is a leap year and if it is evenly divisible by 400 using␣
 ↪if-else statements
if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    leap_year = True
else:
    leap_year = False

# Print the results
if leap_year:
    print(year, "is a leap year.")
else:
    print(year, "is not a leap year.")
```

```
Enter a year:  2024

2024 is a leap year.
```

[19]:
```
#20. Develop a program that checks if a given number is prime or not using␣
 ↪nested if-else statements.

# Input: Get a number from the user
num = int(input("Enter a number: "))

# Check if the number is prime using nested if-else statements
if num > 1:
    for i in range(2, num):
        if (num % i) == 0:
            is_prime = False
```

```
            break
    else:
        is_prime = True
else:
    is_prime = False

# Print the result
if is_prime:
    print(num, "is a prime number.")
else:
    print(num, "is not a prime number.")
```

```
Enter a number:  12
12 is not a prime number.
```

[21]:
```
 #Elif Statements:
#21. Write a Python program to assign grades based on different ranges of␣
 ↪scores using elif statements.

# Input: Get the student's score from the user
score = float(input("Enter the student's score (0-100): "))

# Assign grades using elif statements based on score ranges
if 0 <= score < 60:
    grade = 'F'
elif 60 <= score < 70:
    grade = 'D'
elif 70 <= score < 80:
    grade = 'C'
elif 80 <= score < 90:
    grade = 'B'
elif 90 <= score <= 100:
    grade = 'A'
else:
    grade = 'Invalid'

# Print the assigned grade
if grade != 'Invalid':
    print(f"The student's grade is: {grade}")
else:
    print("Invalid score. Please enter a score between 0 and 100.")
```

```
Enter the student's score (0-100):  43
The student's grade is: F
```

[22]:
```
#22. Implement a program to determine the type of a triangle based on its␣
 ↪angles.

# Input: Get the measures of the three angles of the triangle from the user
angle1 = float(input("Enter the measure of the first angle: "))
angle2 = float(input("Enter the measure of the second angle: "))
angle3 = float(input("Enter the measure of the third angle: "))

# Determine the type of triangle using if-elif-else statements
if angle1 + angle2 + angle3 == 180:
    if angle1 < 90 and angle2 < 90 and angle3 < 90:
        triangle_type = "Acute"
    elif angle1 == 90 or angle2 == 90 or angle3 == 90:
        triangle_type = "Right"
    else:
        triangle_type = "Obtuse"
else:
    triangle_type = "Invalid"

# Print the type of triangle
if triangle_type != "Invalid":
    print("The triangle is:", triangle_type)
else:
    print("Invalid input. Please enter valid angle measures.")
```

```
Enter the measure of the first angle:  34
Enter the measure of the second angle:  32
Enter the measure of the third angle:  123

Invalid input. Please enter valid angle measures.
```

[23]:
```
#23. Develop a program to categorize a given person's BMI into underweight,␣
 ↪normal, overweight, or obese using elif statements.

# Input: Get the person's weight (in kilograms) and height (in meters) from the␣
 ↪user
weight = float(input("Enter your weight (kg): "))
height = float(input("Enter your height (m): "))

# Calculate BMI
bmi = weight / (height ** 2)

# Categorize the BMI using if-elif-else statements
if bmi < 18.5:
    category = "Underweight"
elif 18.5 <= bmi < 24.9:
    category = "Normal Weight"
```

```
elif 25 <= bmi < 29.9:
    category = "Overweight"
else:
    category = "Obese"

# Print the BMI category
print(f"Your BMI is {bmi:.2f}, which falls into the category of {category}.")
```

```
Enter your weight (kg):  78
Enter your height (m):  1.82

Your BMI is 23.55, which falls into the category of Normal Weight.
```

[24]:
```
#24. Create a program that determines whether a given number is positive,␣
 ↪negative, or zero using elif statements.

# Input: Get a number from the user
number = float(input("Enter a number: "))

# Determine if the number is positive, negative, or zero using elif statements
if number > 0:
    print("The number is positive.")
elif number < 0:
    print("The number is negative.")
else:
    print("The number is zero.")
```

```
Enter a number:  43

The number is positive.
```

[25]:
```
#25. Write a Python script to determine the type of a character (uppercase,␣
 ↪lowercase, or special) using elif statements.

# Input: Get a character from the user (assuming a single character input)
char = input("Enter a character: ")

# Determine the type of character using elif statements
if char.isalpha():
    if char.islower():
        char_type = "Lowercase"
    else:
        char_type = "Uppercase"
elif char.isdigit():
    char_type = "Digit"
else:
    char_type = "Special Character"
```

```
# Print the type of character
print(f"The character '{char}' is of type: {char_type}")
```

```
Enter a character:  r

The character 'r' is of type: Lowercase
```

[26]:
```
#26. Implement a program to calculate the discounted price based on different␣
 ↪purchase amounts using elif statements.

# Input: Get the purchase amount from the user
purchase_amount = float(input("Enter the purchase amount: "))

# Calculate the discounted price using elif statements
if purchase_amount >= 1000:
    discount_percentage = 10
elif purchase_amount >= 500:
    discount_percentage = 5
elif purchase_amount >= 200:
    discount_percentage = 2
else:
    discount_percentage = 0

# Calculate the discount amount
discount_amount = (discount_percentage / 100) * purchase_amount

# Calculate the final price after applying the discount
final_price = purchase_amount - discount_amount

# Print the results
print(f"Original Price: ${purchase_amount:.2f}")
print(f"Discount Percentage: {discount_percentage}%")
print(f"Discount Amount: ${discount_amount:.2f}")
print(f"Final Price: ${final_price:.2f}")
```

```
Enter the purchase amount:  100

Original Price: $100.00
Discount Percentage: 0%
Discount Amount: $0.00
Final Price: $100.00
```

[27]:
```
#27. Develop a program to calculate the electricity bill based on different␣
 ↪consumption slabs using elif statements.

# Input: Get the electricity consumption in kilowatt-hours (kWh) from the user
consumption = float(input("Enter the electricity consumption in kWh: "))
```

```
# Initialize variables for the bill and rate
bill_amount = 0.0
rate_per_unit = 0.0

# Determine the rate per unit and calculate the bill using elif statements
if consumption <= 50:
    rate_per_unit = 2.60
    bill_amount = consumption * rate_per_unit
elif consumption <= 150:
    rate_per_unit = 3.25
    bill_amount = 50 * 2.60 + (consumption - 50) * rate_per_unit
elif consumption <= 250:
    rate_per_unit = 4.00
    bill_amount = 50 * 2.60 + 100 * 3.25 + (consumption - 150) * rate_per_unit
else:
    rate_per_unit = 5.25
    bill_amount = 50 * 2.60 + 100 * 3.25 + 100 * 4.00 + (consumption - 250) *␣
 ↪rate_per_unit

# Print the results
print(f"Electricity Consumption: {consumption} kWh")
print(f"Rate per Unit: ${rate_per_unit:.2f} per kWh")
print(f"Total Bill Amount: ${bill_amount:.2f}")
```

```
Enter the electricity consumption in kWh:  12
```

```
Electricity Consumption: 12.0 kWh
Rate per Unit: $2.60 per kWh
Total Bill Amount: $31.20
```

[28]:
```
#28. Create a program to determine the type of quadrilateral based on its␣
 ↪angles and sides using elif statements.

# Input: Get the angle measures of the quadrilateral from the user
angle1 = float(input("Enter the measure of angle 1 (degrees): "))
angle2 = float(input("Enter the measure of angle 2 (degrees): "))
angle3 = float(input("Enter the measure of angle 3 (degrees): "))
angle4 = float(input("Enter the measure of angle 4 (degrees): "))

# Check if it's a square, rectangle, or other based on angle measures using␣
 ↪elif statements
if angle1 == angle2 == angle3 == angle4 == 90:
    quadrilateral_type = "Square"
elif angle1 == angle2 == angle3 == angle4:
    quadrilateral_type = "Rectangle"
else:
    quadrilateral_type = "Other"
```

```
# Print the type of quadrilateral
print(f"The quadrilateral is a {quadrilateral_type}.")
```

```
Enter the measure of angle 1 (degrees):  120
Enter the measure of angle 2 (degrees):  60
Enter the measure of angle 3 (degrees):  40
Enter the measure of angle 4 (degrees):  140

The quadrilateral is a Other.
```

[29]:
```
#29. Write a Python script to determine the season based on a user-provided␣
 ↪month using elif statements.

# Input: Get the month from the user
month = input("Enter a month (e.g., January, February, etc.): ")

# Convert the input month to lowercase for case-insensitive comparison
month = month.lower()

# Determine the season based on the month using elif statements
if month in ("december", "january", "february"):
    season = "Winter"
elif month in ("march", "april", "may"):
    season = "Spring"
elif month in ("june", "july", "august"):
    season = "Summer"
elif month in ("september", "october", "november"):
    season = "Fall"
else:
    season = "Invalid"

# Print the season or an error message
if season != "Invalid":
    print(f"The season for {month.capitalize()} is {season}.")
else:
    print("Invalid month. Please enter a valid month name.")
```

```
Enter a month (e.g., January, February, etc.):  March

The season for March is Spring.
```

[30]:
```
#30. Implement a program to determine the type of a year (leap or common) and␣
 ↪month (30 or 31 days) using elif statements.

# Input: Get the year and month from the user
year = int(input("Enter a year: "))
month = input("Enter a month (e.g., January, February, etc.): ")
```

```
# Convert the input month to lowercase for case-insensitive comparison
month = month.lower()

# Determine the type of year (leap or common) using elif statements
if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    year_type = "Leap"
else:
    year_type = "Common"

# Determine the number of days in the month using elif statements
if month in ("january", "march", "may", "july", "august", "october",␣
 ↪"december"):
    days_in_month = 31
elif month in ("april", "june", "september", "november"):
    days_in_month = 30
elif month == "february":
    if year_type == "Leap":
        days_in_month = 29
    else:
        days_in_month = 28
else:
    days_in_month = "Invalid"

# Print the results
if days_in_month != "Invalid":
    print(f"The year {year} is a {year_type} year.")
    print(f"{month.capitalize()} has {days_in_month} days.")
else:
    print("Invalid month. Please enter a valid month name.")
```

```
Enter a year:  2023
Enter a month (e.g., January, February, etc.):  April

The year 2023 is a Common year.
April has 30 days.
```

[1]:
```
#Assignment Questions
#Basic Level:
#1. Write a Python program that checks if a given number is positive, negative,␣
 ↪or zero.

# Input a number from the user
number = float(input("Enter a number: "))

# Check if the number is positive, negative, or zero
if number > 0:
```

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

```
Enter a number:  13

The number is positive.
```

[2]:
```
#2. Create a program to determine if a person is eligible to vote based on␣
 ↪their age.

# Input the person's age
age = int(input("Enter your age: "))

# Check if the person is eligible to vote
if age >= 18:
    print("You are eligible to vote.")
else:
    print("You are not eligible to vote.")
```

```
Enter your age:  45

You are eligible to vote.
```

[3]:
```
#3. Write a program to find the maximum of two given numbers using conditional␣
 ↪statements.

# Input two numbers from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))

# Compare the numbers to find the maximum
if num1 > num2:
    max_num = num1
else:
    max_num = num2

# Print the maximum number
print("The maximum number is:", max_num)
```

```
Enter the first number:  65
Enter the second number:  78

The maximum number is: 78.0
```

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

```
# Input the exam score from the user
score = float(input("Enter the exam score: "))

# Define the grading scale (you can adjust this as needed)
A_score = 90
B_score = 80
C_score = 70
D_score = 60

# Determine the grade based on the score
if score >= A_score:
    grade = "A"
elif score >= B_score:
    grade = "B"
elif score >= C_score:
    grade = "C"
elif score >= D_score:
    grade = "D"
else:
    grade = "F"

# Print the grade
print("The grade is:", grade)
```

```
Enter the exam score:  78
```

```
The grade is: C
```

[6]:
```
#5. Create a program that checks if a year is a leap year or not.

# Input a year from the user
year = int(input("Enter a year: "))

# Check if it's a leap year
if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    print(year, "is a leap year.")
else:
    print(year, "is not a leap year.")
```

```
Enter a year:  2018
```

```
2018 is not a leap year.
```

[7]:
```
#6. Write a program to classify a triangle based on its sides' lengths.

# Input the lengths of the three sides of the triangle
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: "))

# Check the type of triangle
if side1 == side2 == side3:
    triangle_type = "equilateral"
elif side1 == side2 or side1 == side3 or side2 == side3:
    triangle_type = "isosceles"
else:
    triangle_type = "scalene"

# Print the classification
print("The triangle is a", triangle_type, "triangle.")
```

```
Enter the length of the first side:  23
Enter the length of the second side:  54
Enter the length of the third side:  19

The triangle is a scalene triangle.
```

[9]:
```
#7. Build a program that determines the largest of three given numbers.


# Input three numbers from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))

# Find the largest number using conditional statements
if num1 >= num2 and num1 >= num3:
    largest = num1
elif num2 >= num1 and num2 >= num3:
    largest = num2
else:
    largest = num3

# Print the largest number
print("The largest number is:", largest)
```

```
Enter the first number:  34
Enter the second number:  45
Enter the third number:  23

The largest number is: 45.0
```

[13]:
```
#8. Program to Check if a Character is a Vowel or a Consonant:
```

```
# Input a character from the user (assuming a single character is entered)
char = input("Enter a character: ")

# Convert the character to lowercase to handle uppercase input
char = char.lower()

# Check if it's a vowel or a consonant
if char.isalpha() and len(char) == 1:
    if char in 'aeiou':
        print("It's a vowel.")
    else:
        print("It's a consonant.")
else:
    print("Please enter a valid single character.")
```

```
Enter a character:  t
```

```
It's a consonant.
```

[1]:
```
#9. Program to Calculate Total Cost of a Shopping Cart with Discounts:


# Input the item prices and quantities from the user
num_items = int(input("Enter the number of items: "))
total_cost = 0

# Calculate the total cost
for i in range(num_items):
    price = float(input(f"Enter the price of item {i + 1}: "))
    quantity = int(input(f"Enter the quantity of item {i + 1}: "))
    total_cost += price * quantity

# Apply a discount if the total cost exceeds a certain amount
if total_cost >= 100:
    total_cost *= 0.9  # 10% discount for total cost >= $100

# Print the total cost
print(f"The total cost is: ${total_cost:.2f}")
```

```
Enter the number of items:  3
Enter the price of item 1:  1
Enter the quantity of item 1:  2
Enter the price of item 2:  2
Enter the quantity of item 2:  4
Enter the price of item 3:  5
```

```
Enter the quantity of item 3:  6

The total cost is: $40.00
```

[2]:
```python
# 10. Program to Check if a Number is Even or Odd:

# Input a number from the user
number = int(input("Enter a number: "))

# Check if it's even or odd
if number % 2 == 0:
    print("The number is even.")
else:
    print("The number is odd.")
```

```
Enter a number:  14

The number is even.
```

[4]:
```python
#Intermediate Level:

#11. Write a program that calculates the roots of a quadratic equation .


import math

# Input coefficients of the quadratic equation
a = float(input("Enter the coefficient 'a': "))
b = float(input("Enter the coefficient 'b': "))
c = float(input("Enter the coefficient 'c': "))

# Calculate the discriminant
discriminant = b**2 - 4*a*c

# Check if the discriminant is positive, negative, or zero
if discriminant > 0:
    root1 = (-b + math.sqrt(discriminant)) / (2*a)
    root2 = (-b - math.sqrt(discriminant)) / (2*a)
    print(f"Roots are real and different: {root1} and {root2}")
elif discriminant == 0:
    root1 = -b / (2*a)
    print(f"Roots are real and equal: {root1}")
else:
    real_part = -b / (2*a)
    imaginary_part = math.sqrt(abs(discriminant)) / (2*a)
    print(f"Roots are complex: {real_part} + {imaginary_part}i and {real_part}␣
 ↪- {imaginary_part}i")
```

```
Enter the coefficient 'a':  4
```

```
Enter the coefficient 'b':  5
Enter the coefficient 'c':  3

Roots are complex: -0.625 + 0.5994789404140899i and -0.625 - 0.5994789404140899i
```

[5]:
```python
#12. Program to Determine the Day of the Week based on the Day Number (1-7):


# Input the day number (1-7)
day_number = int(input("Enter a day number (1-7): "))

# Define a list of days of the week
days_of_week = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday",␣
 ↪"Saturday", "Sunday"]

# Check if the input is within the valid range
if 1 <= day_number <= 7:
    day_name = days_of_week[day_number - 1]
    print(f"The day of the week is {day_name}.")
else:
    print("Invalid day number. Please enter a number between 1 and 7.")
```

```
Enter a day number (1-7):  6

The day of the week is Saturday.
```

[7]:
```python
#13. Program to Calculate the Factorial of a Given Number using Recursion:


# Recursive function to calculate factorial
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

# Input a number from the user
num = int(input("Enter a non-negative integer: "))

# Check if the number is non-negative
if num < 0:
    print("Factorial is undefined for negative numbers.")
else:
    result = factorial(num)
    print(f"The factorial of {num} is {result}.")
```

```
Enter a non-negative integer:  8

The factorial of 8 is 40320.
```

[8]:
```
#14. Program to Find the Largest Among Three Numbers without Using the max()␣
↪Function:


# Input three numbers from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))

# Find the largest number without using max()
largest = num1

if num2 > largest:
    largest = num2
if num3 > largest:
    largest = num3

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

```
Enter the first number:  6
Enter the second number:  4
Enter the third number:  3
```

```
The largest number is: 6.0
```

[9]:
```
#15. Program Simulating a Basic ATM Transaction Menu:


# Initialize the account balance
balance = 1000

# Display the ATM menu
print("Welcome to the ATM")
print("1. Check Balance")
print("2. Deposit")
print("3. Withdraw")
print("4. Exit")

# Get user choice
choice = int(input("Enter your choice: "))

# Process the choice
if choice == 1:
    print(f"Your account balance is ${balance:.2f}")
elif choice == 2:
    deposit_amount = float(input("Enter the deposit amount: "))
    balance += deposit_amount
```

```
    print(f"Deposited ${deposit_amount:.2f} successfully. Your new balance is␣
 ↪${balance:.2f}")
elif choice == 3:
    withdrawal_amount = float(input("Enter the withdrawal amount: "))
    if withdrawal_amount > balance:
        print("Insufficient funds.")
    else:
        balance -= withdrawal_amount
        print(f"Withdrew ${withdrawal_amount:.2f} successfully. Your new␣
 ↪balance is ${balance:.2f}")
elif choice == 4:
    print("Thank you for using the ATM. Have a nice day!")
else:
    print("Invalid choice. Please select a valid option.")
```

```
Welcome to the ATM
1. Check Balance
2. Deposit
3. Withdraw
4. Exit

Enter your choice:  3
Enter the withdrawal amount:  1000

Withdrew $1000.00 successfully. Your new balance is $0.00
```

[10]:
```
#16. Program to Check if a String is a Palindrome:


# Input a string from the user
string = input("Enter a string: ")

# Remove spaces and convert to lowercase for case-insensitive comparison
cleaned_string = string.replace(" ", "").lower()

# Check if it's a palindrome
if cleaned_string == cleaned_string[::-1]:
    print("It's a palindrome.")
else:
    print("It's not a palindrome.")
```

```
Enter a string:  Indians are all around the globe

It's not a palindrome.
```

[11]:
```
#17. Program to Calculate the Average of a List of Numbers, Excluding Smallest␣
 ↪and Largest Values:
```

```python
# Input a list of numbers from the user
numbers = input("Enter a list of numbers separated by spaces: ").split()
numbers = [float(num) for num in numbers]

# Check if there are at least 3 numbers
if len(numbers) < 3:
    print("Please enter at least three numbers.")
else:
    # Remove the smallest and largest values
    numbers.remove(min(numbers))
    numbers.remove(max(numbers))

    # Calculate the average
    average = sum(numbers) / len(numbers)
    print("The average (excluding smallest and largest) is:", average)
```

```
Enter a list of numbers separated by spaces:  1  2  5  6  7  8
```

```
The average (excluding smallest and largest) is: 5.0
```

[12]:
```python
#18. Program to Convert Temperature from Celsius to Fahrenheit:

# Input temperature in Celsius
celsius = float(input("Enter temperature in Celsius: "))

# Convert to Fahrenheit
fahrenheit = (celsius * 9/5) + 32

print(f"{celsius}°C is equal to {fahrenheit}°F")
```

```
Enter temperature in Celsius:  54
```

```
54.0°C is equal to 129.2°F
```

[13]:
```python
#19. Program Simulating a Basic Calculator:


# Display the calculator menu
print("Calculator Menu:")
print("1. Addition")
print("2. Subtraction")
print("3. Multiplication")
print("4. Division")

# Get user choice
choice = int(input("Enter your choice (1/2/3/4): "))

# Get input numbers
```

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

# Perform the chosen operation
if choice == 1:
    result = num1 + num2
elif choice == 2:
    result = num1 - num2
elif choice == 3:
    result = num1 * num2
elif choice == 4:
    if num2 == 0:
        print("Division by zero is not allowed.")
    else:
        result = num1 / num2
else:
    print("Invalid choice. Please select a valid operation.")
    result = None

if result is not None:
    print("Result:", result)
```

```
Calculator Menu:
1. Addition
2. Subtraction
3. Multiplication
4. Division

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

Invalid choice. Please select a valid operation.
```

[1]:
```
#20. Write a program that determines the roots of a cubic equation using the␣
 ↪Cardano formula.

import math

# Input coefficients of the cubic equation
a = float(input("Enter coefficient 'a': "))
b = float(input("Enter coefficient 'b': "))
c = float(input("Enter coefficient 'c': "))
d = float(input("Enter coefficient 'd': "))

# Calculate intermediate values
p = c / a - (b ** 2) / (3 * (a ** 2))
q = (2 * (b ** 3)) / (27 * (a ** 3)) - (b * c) / (3 * (a ** 2)) + d / a
```

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

if discriminant > 0:
    # One real root and two complex roots
    u = (-q / 2 + math.sqrt(discriminant)) ** (1 / 3)
    v = (-q / 2 - math.sqrt(discriminant)) ** (1 / 3)

    real_root = u + v - (b / (3 * a))
    print(f"The real root is: {real_root:.4f}")
else:
    # Three real roots (one real and two complex conjugates)
    r = math.sqrt((-p ** 3) / 27)
    theta = math.acos(-q / (2 * r))

    root1 = 2 * math.sqrt(-p / 3) * math.cos(theta / 3) - (b / (3 * a))
    root2 = 2 * math.sqrt(-p / 3) * math.cos((theta + 2 * math.pi) / 3) - (b /␣
 ↪(3 * a))
    root3 = 2 * math.sqrt(-p / 3) * math.cos((theta - 2 * math.pi) / 3) - (b /␣
 ↪(3 * a))

    print(f"The real roots are: {root1:.4f}, {root2:.4f}, {root3:.4f}")
```

```
Enter coefficient 'a':  1
Enter coefficient 'b':  -6
Enter coefficient 'c':  11
Enter coefficient 'd':  -6
```

```
The real roots are: 3.0000, 1.0000, 2.0000
```

[2]:
```
#21. Program to Calculate Income Tax based on Tax Brackets:


# Input the user's income
income = float(input("Enter your annual income: "))

# Define tax brackets and their corresponding tax rates
tax_brackets = [(10000, 0.10), (30000, 0.20), (70000, 0.30), (100000, 0.40)]
tax_due = 0

# Calculate income tax based on tax brackets
for bracket, rate in tax_brackets:
    if income > bracket:
        taxable_amount = bracket
    else:
        taxable_amount = income
```

```
    tax_due += taxable_amount * rate
    income -= taxable_amount

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

```
Enter your annual income:  1000000
Your income tax due is: $68000.00
```

[3]:
```
#22. Program to Simulate Rock-Paper-Scissors Game:


import random

# Input user's choice
user_choice = input("Choose rock, paper, or scissors: ").lower()

# Generate a random choice for the computer
choices = ["rock", "paper", "scissors"]
computer_choice = random.choice(choices)

# Determine the winner
if user_choice == computer_choice:
    result = "It's a tie!"
elif (
    (user_choice == "rock" and computer_choice == "scissors") or
    (user_choice == "scissors" and computer_choice == "paper") or
    (user_choice == "paper" and computer_choice == "rock")
):
    result = f"You win! Computer chose {computer_choice}."
else:
    result = f"Computer wins! Computer chose {computer_choice}."

print(result)
```

```
Choose rock, paper, or scissors:  paper
Computer wins! Computer chose scissors.
```

[5]:
```
 #23. Program to Generate a Random Password:


import random
import string

# Input password length
length = int(input("Enter the password length: "))
```

```
# Define character sets for different complexity levels
lowercase_letters = string.ascii_lowercase
uppercase_letters = string.ascii_uppercase
digits = string.digits
special_characters = "!@#$%^&*()_+[]{}|;:,.<>?-="

# Combine character sets based on user preferences
characters = lowercase_letters + uppercase_letters + digits + special_characters

# Generate a random password
password = ''.join(random.choice(characters) for _ in range(length))

print(f"Your random password is: {password}")
```

```
Enter the password length:  14
Your random password is: ^{9!X@#00)#rH]
```

[6]:
```
#24. Text-Based Adventure Game:


print("Welcome to the Text-Based Adventure Game!")

# Introduction
print("You find yourself in a dark forest. You can go left or right.")
choice1 = input("Enter 'left' or 'right': ").lower()

# Branch 1
if choice1 == "left":
    print("You stumble upon a treasure chest!")
    choice2 = input("Do you open it? (yes/no): ").lower()

    if choice2 == "yes":
        print("Congratulations! You found a bag of gold!")
    else:
        print("You decide not to open it and continue your adventure.")

# Branch 2
elif choice1 == "right":
    print("You encounter a ferocious dragon!")
    choice3 = input("Do you try to fight it or run away? (fight/run): ").lower()

    if choice3 == "fight":
        print("You bravely fight the dragon, but unfortunately, you lose the␣
 ↪battle.")
    elif choice3 == "run":
        print("You wisely choose to run away and escape the dragon's wrath.")
    else:
```

```
        print("Invalid choice. You hesitate and the dragon attacks!")

# Invalid Choice
else:
    print("Invalid choice. You are lost in the forest.")

print("Game Over!")
```

```
Welcome to the Text-Based Adventure Game!
You find yourself in a dark forest. You can go left or right.

Enter 'left' or 'right':  left

You stumble upon a treasure chest!

Do you open it? (yes/no):  yes

Congratulations! You found a bag of gold!
Game Over!
```

[7]:
```
#25. Linear Equation Solver:


# Input coefficients a and b from the user
a = float(input("Enter coefficient 'a': "))
b = float(input("Enter coefficient 'b': "))

# Solve the equation ax + b = 0
if a == 0:
    if b == 0:
        print("Infinite solutions (identity equation).")
    else:
        print("No solution (contradiction).")
else:
    x = -b / a
    print(f"The solution for x is: {x}")
```

```
Enter coefficient 'a':  3
Enter coefficient 'b':  4

The solution for x is: -1.3333333333333333
```

[8]:
```
#26. Quiz Game with Multiple-Choice Questions:


#Define a list of questions and their choices
questions = [
    {
        "question": "What is the capital of France?",
        "choices": ["Paris", "London", "Berlin"],
```

```
        "correct_answer": "Paris",
    },
    {
        "question": "Which planet is known as the 'Red Planet'?",
        "choices": ["Mars", "Venus", "Jupiter"],
        "correct_answer": "Mars",
    },
]

# Initialize the score
score = 0

# Loop through the questions
for i, question_data in enumerate(questions, start=1):
    print(f"Question {i}: {question_data['question']}")
    for j, choice in enumerate(question_data['choices'], start=1):
        print(f"{j}. {choice}")

    user_answer = input("Enter the number of your answer: ")
    if user_answer.isdigit():
        user_answer = int(user_answer)
        if 1 <= user_answer <= len(question_data['choices']):
            if question_data['choices'][user_answer - 1] ==␣
 ↪question_data['correct_answer']:
                print("Correct!")
                score += 1
            else:
                print("Wrong answer. The correct answer is:",␣
 ↪question_data['correct_answer'])
        else:
            print("Invalid choice.")
    else:
        print("Invalid input.")

# Print the final score
print(f"Your score is: {score}/{len(questions)}")
```

```
Question 1: What is the capital of France?
1. Paris
2. London
3. Berlin

Enter the number of your answer:  3

Wrong answer. The correct answer is: Paris
Question 2: Which planet is known as the 'Red Planet'?
1. Mars
2. Venus
```

```
3. Jupiter

Enter the number of your answer:  2

Wrong answer. The correct answer is: Mars
Your score is: 0/2
```

[9]:
```python
#27. Program to Determine if a Year is Prime or Not:


# Input a year from the user
year = int(input("Enter a year: "))

# Check if it's a prime year
is_prime = True

if year <= 1:
    is_prime = False
else:
    for i in range(2, int(year**0.5) + 1):
        if year % i == 0:
            is_prime = False
            break

if is_prime:
    print(f"{year} is a prime year.")
else:
    print(f"{year} is not a prime year.")
```

```
Enter a year:  2014

2014 is not a prime year.
```

[10]:
```python
#28. Program to Sort Three Numbers in Ascending Order using Conditional␣
 ↪Statements:


# Input three numbers from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))

# Sort the numbers in ascending order using conditional statements
if num1 <= num2 <= num3:
    sorted_nums = (num1, num2, num3)
elif num1 <= num3 <= num2:
    sorted_nums = (num1, num3, num2)
elif num2 <= num1 <= num3:
    sorted_nums = (num2, num1, num3)
```

```
elif num2 <= num3 <= num1:
    sorted_nums = (num2, num3, num1)
elif num3 <= num1 <= num2:
    sorted_nums = (num3, num1, num2)
else:
    sorted_nums = (num3, num2, num1)

print("The numbers in ascending order are:", sorted_nums)
```

```
Enter the first number:  4
Enter the second number:  5
Enter the third number:  3
```

```
The numbers in ascending order are: (3.0, 4.0, 5.0)
```

[11]:
```
#29. Program to Determine the Roots of a Quartic Equation using Numerical␣
 ↪Methods:


import sympy as sp

# Define the variables
x = sp.symbols('x')

# Input coefficients of the quartic equation
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': "))

# Define the quartic equation
quartic_eq = a * x**4 + b * x**3 + c * x**2 + d * x + e

# Find the roots of the quartic equation
roots = sp.solve(quartic_eq, x)

print("The roots of the quartic equation are:", roots)
```

```
Enter coefficient 'a':  22
Enter coefficient 'b':  3
Enter coefficient 'c':  4
Enter coefficient 'd':  5
Enter coefficient 'e':  -3
```

```
The roots of the quartic equation are: [-0.710605646850089, 0.372778747005538,
0.100731631740457 - 0.710372939073024*I, 0.100731631740457 +
0.710372939073024*I]
```

[12]:
```
#30. Program to Calculate BMI and Provide Health Recommendations:


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

# Calculate BMI
bmi = weight / (height ** 2)

# Determine BMI category and provide recommendations
if bmi < 18.5:
    category = "Underweight"
    recommendation = "You may need to gain some weight. Consult a healthcare␣
 ↪professional."
elif 18.5 <= bmi < 24.9:
    category = "Normal Weight"
    recommendation = "Your weight is within the healthy range. Maintain a␣
 ↪balanced diet and exercise regularly."
elif 25 <= bmi < 29.9:
    category = "Overweight"
    recommendation = "You may need to lose some weight. Consult a healthcare␣
 ↪professional."
else:
    category = "Obese"
    recommendation = "You may be at risk of health issues due to excess weight.␣
 ↪Consult a healthcare professional."

print(f"Your BMI is: {bmi:.2f}")
print(f"BMI Category: {category}")
print("Recommendation:", recommendation)
```

```
Enter your height (in meters):  1.8
Enter your weight (in kilograms):  80
```

```
Your BMI is: 24.69
BMI Category: Normal Weight
Recommendation: Your weight is within the healthy range. Maintain a balanced
diet and exercise regularly.
```

[13]:
```
#31. Password Validation Program:


import re

# Input a password from the user
password = input("Enter a password: ")
```

```
# Define password complexity rules
# Rule 1: At least 8 characters
# Rule 2: Contains at least one uppercase letter
# Rule 3: Contains at least one lowercase letter
# Rule 4: Contains at least one digit
# Rule 5: Contains at least one special character (e.g., !, @, #, $, %, etc.)
rules = [
    lambda p: len(p) >= 8,
    lambda p: any(c.isupper() for c in p),
    lambda p: any(c.islower() for c in p),
    lambda p: any(c.isdigit() for c in p),
    lambda p: re.search(r'[!@#$%^&*(),.?":{}|<>]', p),
]

# Check if the password meets all rules
if all(rule(password) for rule in rules):
    print("Password is valid.")
else:
    print("Password is not valid. Please follow the password rules.")
```

```
Enter a password:  1234@abcd
```

```
Password is not valid. Please follow the password rules.
```

[4]:
```
#32. Matrix Addition and Subtraction:


import numpy as np

# Input matrix dimensions
rows = int(input("Enter the number of rows: "))
cols = int(input("Enter the number of columns: "))

# Input matrices
matrix1 = np.zeros((rows, cols))
matrix2 = np.zeros((rows, cols))

print("Enter elements for matrix 1:")
for i in range(rows):
    for j in range(cols):
        matrix1[i][j] = float(input(f"Enter element at row {i+1}, column {j+1}:␣
 ↪"))

print("Enter elements for matrix 2:")
for i in range(rows):
    for j in range(cols):
```

```
        matrix2[i][j] = float(input(f"Enter element at row {i+1}, column {j+1}:␣
 ↪"))

# Perform matrix addition and subtraction
matrix_sum = matrix1 + matrix2
matrix_diff = matrix1 - matrix2

print("Matrix 1:")
print(matrix1)
print("Matrix 2:")
print(matrix2)
print("Matrix Sum:")
print(matrix_sum)
print("Matrix Difference:")
print(matrix_diff)
```

```
Enter the number of rows:  2
Enter the number of columns:  2

Enter elements for matrix 1:

Enter element at row 1, column 1:  1
Enter element at row 1, column 2:  2
Enter element at row 2, column 1:  3
Enter element at row 2, column 2:  4

Enter elements for matrix 2:

Enter element at row 1, column 1:  1
Enter element at row 1, column 2:  2
Enter element at row 2, column 1:  3
Enter element at row 2, column 2:  3

Matrix 1:
[[1. 2.]
 [3. 4.]]
Matrix 2:
[[1. 2.]
 [3. 3.]]
Matrix Sum:
[[2. 4.]
 [6. 7.]]
Matrix Difference:
[[0. 0.]
 [0. 1.]]
```

[1]:
```
#33. GCD Calculation using the Euclidean Algorithm:


# Function to calculate GCD using the Euclidean Algorithm
```

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

# Input two numbers from the user
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))

# Calculate and print the GCD
result = gcd(num1, num2)
print(f"The GCD of {num1} and {num2} is {result}.")
```

```
Enter the first number:  23
Enter the second number:  32
```

```
The GCD of 23 and 32 is 1.
```

[2]:
```
#34. Matrix Multiplication:


import numpy as np

# Input matrix dimensions
rows1 = int(input("Enter the number of rows for matrix 1: "))
cols1 = int(input("Enter the number of columns for matrix 1: "))
rows2 = int(input("Enter the number of rows for matrix 2: "))
cols2 = int(input("Enter the number of columns for matrix 2: "))

# Check if matrix multiplication is possible
if cols1 != rows2:
    print("Matrix multiplication is not possible. The number of columns in␣
 ↪matrix 1 must be equal to the number of rows in matrix 2.")
else:
    # Input matrices
    matrix1 = np.zeros((rows1, cols1))
    matrix2 = np.zeros((rows2, cols2))

    print("Enter elements for matrix 1:")
    for i in range(rows1):
        for j in range(cols1):
            matrix1[i][j] = float(input(f"Enter element at row {i+1}, column␣
 ↪{j+1}: "))

    print("Enter elements for matrix 2:")
    for i in range(rows2):
        for j in range(cols2):
```

```
            matrix2[i][j] = float(input(f"Enter element at row {i+1}, column␣
 ↪{j+1}: "))

    # Perform matrix multiplication
    result_matrix = np.dot(matrix1, matrix2)

    print("Matrix 1:")
    print(matrix1)
    print("Matrix 2:")
    print(matrix2)
    print("Matrix Multiplication Result:")
    print(result_matrix)
```

```
Enter the number of rows for matrix 1:  2
Enter the number of columns for matrix 1:  3
Enter the number of rows for matrix 2:  2
Enter the number of columns for matrix 2:  2
```

```
Matrix multiplication is not possible. The number of columns in matrix 1 must be
equal to the number of rows in matrix 2.
```

[3]:
```
#35. Text-Based Tic-Tac-Toe Game Against the Computer:


import random

# Initialize the Tic-Tac-Toe board
board = [' ' for _ in range(9)]

# Function to print the Tic-Tac-Toe board
def print_board(board):
    for i in range(0, 9, 3):
        print(f"{board[i]} | {board[i + 1]} | {board[i + 2]}")
        if i < 6:
            print("---------")

# Function to check if a player has won
def check_win(board, player):
    win_combinations = [(0, 1, 2), (3, 4, 5), (6, 7, 8),
                        (0, 3, 6), (1, 4, 7), (2, 5, 8),
                        (0, 4, 8), (2, 4, 6)]

    for combo in win_combinations:
        if board[combo[0]] == board[combo[1]] == board[combo[2]] == player:
            return True
    return False

# Function for computer's move
```

```python
def computer_move(board):
    # Find available positions
    available_moves = [i for i, mark in enumerate(board) if mark == ' ']

    # Choose a random available position
    if available_moves:
        return random.choice(available_moves)
    else:
        return None

# Main game loop
current_player = 'X'
while True:
    print_board(board)

    if current_player == 'X':
        move = int(input("Enter your move (1-9): ")) - 1
    else:
        move = computer_move(board)

    if move is not None and board[move] == ' ':
        board[move] = current_player

        if check_win(board, current_player):
            print_board(board)
            print(f"{current_player} wins!")
            break
        elif ' ' not in board:
            print_board(board)
            print("It's a tie!")
            break

        current_player = 'O' if current_player == 'X' else 'X'
    else:
        print("Invalid move. Try again.")
```

```
  |   |
---------
  |   |
---------
  |   |

Enter your move (1-9):  7

  |   |
---------
  |   |
---------
```

```
X |   |
  |   |
---------
  |   |
---------
X |   | O

Enter your move (1-9):  2

  | X |
---------
  |   |
---------
X |   | O
  | X |
---------
  |   | O
---------
X |   | O

Enter your move (1-9):  3

  | X | X
---------
  |   | O
---------
X |   | O
  | X | X
---------
O |   | O
---------
X |   | O

Enter your move (1-9):  3

Invalid move. Try again.
  | X | X
---------
O |   | O
---------
X |   | O

Enter your move (1-9):  4

Invalid move. Try again.
  | X | X
---------
O |   | O
---------
X |   | O

Enter your move (1-9):  5
```

```
  | X | X
---------
O | X | O
---------
X |   | O
X wins!
```

[5]:
```
#36. Program to Generate Fibonacci Numbers (Iterative):


# Input the number of terms in the Fibonacci sequence
n = int(input("Enter the number of Fibonacci terms to generate: "))

# Initialize the first two Fibonacci numbers
fibonacci = [0, 1]

# Generate Fibonacci numbers iteratively
for i in range(2, n):
    next_term = fibonacci[i - 1] + fibonacci[i - 2]
    fibonacci.append(next_term)

# Print the generated Fibonacci sequence
print("Fibonacci sequence:")
print(fibonacci)
```

```
Enter the number of Fibonacci terms to generate:  34

Fibonacci sequence:
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584,
4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229,
832040, 1346269, 2178309, 3524578]
```

[6]:
```
#37. Fibonacci Sequence with Memoization:

# Dictionary to store Fibonacci numbers with memoization
fibonacci_cache = {}

# Function to calculate the nth Fibonacci term with memoization
def fibonacci(n):
    if n in fibonacci_cache:
        return fibonacci_cache[n]

    if n <= 1:
        result = n
    else:
        result = fibonacci(n - 1) + fibonacci(n - 2)

    fibonacci_cache[n] = result
```

```
    return result

# Input the term of the Fibonacci sequence to calculate
n = int(input("Enter the term of the Fibonacci sequence to calculate: "))

# Calculate and print the nth Fibonacci term
result = fibonacci(n)
print(f"The {n}th Fibonacci term is: {result}")
```

```
Enter the term of the Fibonacci sequence to calculate:  23
```

```
The 23th Fibonacci term is: 28657
```

[7]:
```
#38. Program to Generate a Calendar:


import calendar

# Input month and year
year = int(input("Enter the year (e.g., 2023): "))
month = int(input("Enter the month (1-12): "))

# Create a calendar for the specified month and year
cal = calendar.month(year, month)

# Print the calendar
print(f"Calendar for {calendar.month_name[month]} {year}:\n")
print(cal)
```

```
Enter the year (e.g., 2023):  2003
Enter the month (1-12):  11
```

```
Calendar for November 2003:

   November 2003
Mo Tu We Th Fr Sa Su
                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
```

[11]:
```
#40. Prime Factors using Trial Division:


# Function to find prime factors using trial division
def prime_factors(n):
```

```
    factors = []
    divisor = 2

    while divisor <= n:
        if n % divisor == 0:
            factors.append(divisor)
            n = n // divisor
        else:
            divisor += 1

    return factors

# Input a number from the user
num = int(input("Enter a number to find its prime factors: "))

# Find and print the prime factors
factors = prime_factors(num)
print(f"The prime factors of {num} are:", factors)
```

```
Enter a number to find its prime factors:  1296
The prime factors of 1296 are: [2, 2, 2, 2, 3, 3, 3, 3]
```

[ ]:
```
#39. Build a program that simulates a basic text-based blackjack game against␣
 ↪the computer.

import random

# Function to calculate the hand value
def hand_value(hand):
    value = 0
    num_aces = 0

    for card in hand:
        if card in "KQJ":
            value += 10
        elif card == "A":
            value += 11
            num_aces += 1
        else:
            value += int(card)

    while num_aces > 0 and value > 21:
        value -= 10
        num_aces -= 1

    return value
```

```
# Initialize the deck and player/computer hands
suits = ["Hearts", "Diamonds", "Clubs", "Spades"]
ranks = ["2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A"]
deck = [rank + " of " + suit for suit in suits for rank in ranks]
random.shuffle(deck)

player_hand = [deck.pop(), deck.pop()]
computer_hand = [deck.pop(), deck.pop()]

# Display initial hands
print("\nPlayer's Hand:", player_hand)
print("Player's Hand Value:", hand_value(player_hand))
print("\nComputer's Hand:", computer_hand[0] + " and an unknown card")

# Main game loop
while True:
    # Check for player blackjack
    if hand_value(player_hand) == 21:
        print("Player has a blackjack! Player wins.")
        break

    # Player's turn
    choice = input("\nDo you want to 'hit' or 'stand'? ").lower()

    if choice == "hit":
        player_hand.append(deck.pop())
        print("\nPlayer's Hand:", player_hand)
        print("Player's Hand Value:", hand_value(player_hand))

        if hand_value(player_hand) > 21:
            print("Player busts. Computer wins.")
            break
    elif choice == "stand":
        # Computer's turn
        while hand_value(computer_hand) < 17:
            computer_hand.append(deck.pop())

        print("\nComputer's Hand:", computer_hand)
        print("Computer's Hand Value:", hand_value(computer_hand))

        if hand_value(computer_hand) > 21:
            print("Computer busts. Player wins.")
        elif hand_value(computer_hand) >= hand_value(player_hand):
            print("Computer wins.")
        else:
            print("Player wins.")
```

```
        break
    else:
        print("Invalid choice. Please enter 'hit' or 'stand'.")

    # Check for player bust
    if hand_value(player_hand) > 21:
        print("Player busts. Computer wins.")
        break
```