```
# Program using different bitwise operators
a = 10  # 1010 in binary
b = 4   # 0100 in binary

print("Bitwise AND:", a & b)   # 0000 = 0
print("Bitwise OR:", a | b)    # 1110 = 14
print("Bitwise XOR:", a ^ b)   # 1110 = 14
print("Bitwise NOT:", ~a)      # -11 (2's complement)
print("Left shift:", a << 2)   # 101000 = 40
print("Right shift:", a >> 2)  # 0010 = 2

# Program using different logical operators
x = True
y = False

print("x and y:", x and y)  # False
print("x or y:", x or y)    # True
print("not x:", not x)      # False
print("not y:", not y)      # True

# Program to swap two variables
a = 5
b = 10

print("Before swap: a =", a, "b =", b)

# Method 1: Using temporary variable
temp = a
a = b
b = temp
```

```
# Method 2: Without temporary variable
# a, b = b, a

print("After swap: a =", a, "b =", b)



# Program to calculate surface volume and area of cylinder
import math

radius = float(input("Enter radius of cylinder: "))
height = float(input("Enter height of cylinder: "))

volume = math.pi * radius**2 * height
surface_area = 2 * math.pi * radius * (radius + height)

print("Volume of cylinder:", round(volume, 2))
print("Surface area of cylinder:", round(surface_area, 2))

# Program to find largest among three numbers
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
num3 = float(input("Enter third number: "))

if (num1 >= num2) and (num1 >= num3):
   largest = num1
elif (num2 >= num1) and (num2 >= num3):
   largest = num2
else:
   largest = num3

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

```
# Program to check leap year
year = int(input("Enter a year: "))

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



# Program to check if number is positive, negative or zero
num = float(input("Enter a number: "))

if num > 0:
   print("Positive number")
elif num == 0:
   print("Zero")
else:
   print("Negative number")



# Program to calculate grade based on marks of 5 subjects
marks = []
total = 0

for i in range(5):
   mark = float(input(f"Enter marks for subject {i+1}: "))
   marks.append(mark)
   total += mark

percentage = total / 5
```

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

print(f"Total Marks: {total}")
print(f"Percentage: {percentage}%")
print(f"Grade: {grade}")
```

```
# Program to find prime numbers from 2 to 100
print("Prime numbers between 2 and 100:")

for num in range(2, 101):
    is_prime = True
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            is_prime = False
            break
    if is_prime:
        print(num, end=' ')
```

```
# Program to check if a number is palindrome
num = int(input("Enter a number: "))
original = num
reverse = 0

while num > 0:
    digit = num % 10
    reverse = reverse * 10 + digit
    num = num // 10

if original == reverse:
    print("The number is a palindrome")
else:
    print("The number is not a palindrome")



# Program to print even numbers between 1 to 100 using while loop
num = 2

print("Even numbers between 1 to 100:")
while num <= 100:
    print(num, end=' ')
    num += 2




# Program to print pattern:
# 1010101
# 101
# 1
```

```
rows = 3
n = 7

for i in range(rows, 0, -1):
    for j in range(0, n):
        print(1 if j % 2 == 0 else 0, end='')
    print()
    n -= 4



# Program to print pattern:
# *
# ***
# *****
# ***
# *
rows = 5
n = 1

for i in range(1, rows + 1):
    print('*' * n)
    if i < (rows + 1) // 2:
        n += 2
    else:
        n -= 2



# Program to reverse a number
num = int(input("Enter a number: "))
```

```
reverse = 0

while num > 0:
    digit = num % 10
    reverse = reverse * 10 + digit
    num = num // 10

print("Reversed number:", reverse)
```

```
# Program to find factorial of a number
num = int(input("Enter a number: "))
factorial = 1

if num < 0:
    print("Factorial doesn't exist for negative numbers")
elif num == 0:
    print("Factorial of 0 is 1")
else:
    for i in range(1, num + 1):
        factorial *= i
    print(f"Factorial of {num} is {factorial}")
```

```
# Program to find Fibonacci series
n = int(input("Enter number of terms: "))
a, b = 0, 1

print("Fibonacci sequence:")
for i in range(n):
```

```
    print(a, end=' ')
    a, b = b, a + b



# Program to find sum of digits of a 4-digit number
num = input("Enter a 4-digit number: ")

if len(num) == 4 and num.isdigit():
    sum_digits = 0
    for digit in num:
        sum_digits += int(digit)
    print("Sum of digits:", sum_digits)
else:
    print("Please enter a valid 4-digit number")



# Program to demonstrate nested if-else
age = int(input("Enter your age: "))
if age >= 18:
    license = input("Do you have a driving license? (yes/no): ")
    if license.lower() == 'yes':
        print("You can drive")
    else:
        print("You need a license to drive")
else:
    print("You are too young to drive")
```

```
# Program to demonstrate list functions
my_list = [3, 1, 4, 1, 5, 9, 2]

print("Original list:", my_list)
print("Length:", len(my_list))
print("Sum:", sum(my_list))
print("Maximum:", max(my_list))
print("Minimum:", min(my_list))
print("Count of 1:", my_list.count(1))
print("Index of 5:", my_list.index(5))

my_list.append(6)
print("After append:", my_list)

my_list.remove(1)
print("After remove:", my_list)

my_list.sort()
print("After sort:", my_list)

my_list.reverse()
print("After reverse:", my_list)



# Program to reverse a list
def reverse_list(lst):
   return lst[::-1]

original_list = [1, 2, 3, 4, 5]
reversed_list = reverse_list(original_list)
```

```
print("Original list:", original_list)
print("Reversed list:", reversed_list)


# Program to find common items in two lists
list1 = [1, 2, 3, 4, 5]
list2 = [4, 5, 6, 7, 8]

common = list(set(list1) & set(list2))
print("Common elements:", common)


# Program to sort a list
my_list = [5, 2, 8, 1, 9, 3]

# Ascending order
my_list.sort()
print("Ascending order:", my_list)

# Descending order
my_list.sort(reverse=True)
print("Descending order:", my_list)


# Program to copy specific elements from tuple
original_tuple = (10, 20, 30, 40, 50, 60)
new_tuple = original_tuple[1:4]  # Elements at index 1, 2, 3

print("Original tuple:", original_tuple)
print("New tuple:", new_tuple)
```

```
# Program to find min and max in tuple
numbers = (45, 23, 67, 12, 89, 34)

print("Tuple:", numbers)
print("Minimum:", min(numbers))
print("Maximum:", max(numbers))



# Program to find repeated items in tuple
my_tuple = (1, 2, 3, 2, 4, 5, 4, 6)
repeated = []

for item in my_tuple:
   if my_tuple.count(item) > 1 and item not in repeated:
      repeated.append(item)

print("Repeated items:", repeated)




# Program to demonstrate set operations
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

print("Union:", A | B)
print("Intersection:", A & B)
print("Difference (A-B):", A - B)
print("Difference (B-A):", B - A)
print("Symmetric Difference:", A ^ B)
```

```
# Program to create set, add and remove items
my_set = {1, 2, 3}
print("Original set:", my_set)

my_set.add(4)
print("After adding 4:", my_set)

my_set.update([5, 6])
print("After updating with [5,6]:", my_set)

my_set.remove(3)
print("After removing 3:", my_set)

my_set.discard(10)  # No error if not found
print("After discarding 10:", my_set)



# Program to find min, max and length of set
num_set = {45, 23, 67, 12, 89, 34}

print("Set:", num_set)
print("Length:", len(num_set))
print("Minimum:", min(num_set))
print("Maximum:", max(num_set))

# Program to sort dictionary by value
my_dict = {'apple': 5, 'banana': 2, 'cherry': 8, 'date': 3}

# Ascending order
sorted_asc = dict(sorted(my_dict.items(), key=lambda item: item[1]))
```

```
print("Ascending order:", sorted_asc)

# Descending order
sorted_desc = dict(sorted(my_dict.items(), key=lambda item: item[1], reverse=True))
print("Descending order:", sorted_desc)



# Program to find min, max and length of set
num_set = {45, 23, 67, 12, 89, 34}

print("Set:", num_set)
print("Length:", len(num_set))
print("Minimum:", min(num_set))
print("Maximum:", max(num_set))



# Program to sort dictionary by value
my_dict = {'apple': 5, 'banana': 2, 'cherry': 8, 'date': 3}

# Ascending order
sorted_asc = dict(sorted(my_dict.items(), key=lambda item: item[1]))
print("Ascending order:", sorted_asc)

# Descending order
sorted_desc = dict(sorted(my_dict.items(), key=lambda item: item[1], reverse=True))
print("Descending order:", sorted_desc)



# Program to concatenate dictionaries
```

```
dic1 = {1: 10, 2: 20}
dic2 = {3: 30, 4: 40}
dic3 = {5: 50, 6: 60}

result = {}
for d in (dic1, dic2, dic3):
    result.update(d)

print("Concatenated dictionary:", result)



# Program to combine dictionaries adding values for common keys
d1 = {'a': 100, 'b': 200, 'c': 300}
d2 = {'a': 300, 'b': 200, 'd': 400}

result = d1.copy()
for key, value in d2.items():
    if key in result:
        result[key] += value
    else:
        result[key] = value

print("Combined dictionary:", result)



# Program to find highest 3 values in dictionary
my_dict = {'a': 500, 'b': 200, 'c': 1500, 'd': 750, 'e': 100}

sorted_items = sorted(my_dict.items(), key=lambda x: x[1], reverse=True)
top_three = dict(sorted_items[:3])
```

```
print("Top three values:", top_three)



# Function to check prime number
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

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



# Function to calculate factorial
def factorial(n):
    if n < 0:
        return "Factorial doesn't exist for negative numbers"
    elif n == 0:
        return 1
    else:
        result = 1
        for i in range(1, n + 1):
            result *= i
        return result
```

```
num = int(input("Enter a number: "))
print(f"Factorial of {num} is {factorial(num)}")
```

```
# Function to count upper and lower case letters
def count_case_letters(string):
    upper = 0
    lower = 0
    for char in string:
        if char.isupper():
            upper += 1
        elif char.islower():
            lower += 1
    return upper, lower

text = input("Enter a string: ")
upper, lower = count_case_letters(text)
print("Upper case letters:", upper)
print("Lower case letters:", lower)
```

```
# Program to demonstrate function arguments
def greet(name, message="Hello"):
    print(f"{message}, {name}!")

# Positional argument
greet("Alice")
```

```
# Keyword argument
greet(message="Hi", name="Bob")

# Default argument
greet("Charlie")

# Variable length arguments
def sum_all(*args):
    return sum(args)

print("Sum:", sum_all(1, 2, 3, 4, 5))
```

```
# Program to demonstrate keyword arguments
def student_info(name, age, grade):
    print(f"Name: {name}, Age: {age}, Grade: {grade}")

# Calling with keyword arguments (order doesn't matter)
student_info(age=20, name="John", grade="A")
student_info(grade="B", age=22, name="Emma")
```

```
# Save this as college.py (module file)
def get_college_name():
    name = input("Enter your college name: ")
    display_college_name(name)

def display_college_name(name):
    print(f"Your college is: {name}")
```

```
# In another file (main program):
# import college
# college.get_college_name()



# Save this as fib_module.py
def fibonacci(n):
    a, b = 0, 1
    series = []
    for _ in range(n):
        series.append(a)
        a, b = b, a + b
    return series

# In another file:
# from fib_module import fibonacci
# print(fibonacci(10))



# Program to display calendar
import calendar

year = int(input("Enter year: "))
month = int(input("Enter month: "))

print(calendar.month(year, month))
```

```
# Program to calculate area of circle using math module
import math

radius = float(input("Enter radius of circle: "))
area = math.pi * radius**2

print(f"Area of circle: {area:.2f}")




"""
To create your own package:
1. Create a directory (package name)
2. Create __init__.py file inside it
3. Add module files to the directory
4. Import using package.module
"""



# Program to demonstrate constructor overloading
class Student:
    def __init__(self, *args):
        if len(args) == 1:
            self.name = args[0]
            self.age = None
        elif len(args) == 2:
            self.name = args[0]
            self.age = args[1]

    def display(self):
        print(f"Name: {self.name}, Age: {self.age}")
```

```
# Different ways to create object
s1 = Student("Alice")
s2 = Student("Bob", 20)

s1.display()
s2.display()



# Program to implement constructor
class Person:
    def __init__(self):
        print("Constructor called")
        self.name = "Unknown"

    def display(self):
        print(f"Name: {self.name}")

p = Person()
p.display()




# Program to implement parameterized constructor
class Employee:
    def __init__(self, name, id):
        self.name = name
        self.id = id

    def display(self):
        print(f"Employee: {self.name}, ID: {self.id}")
```

```
emp = Employee("John Doe", 1001)
emp.display()



# Program to implement constructor overriding
class Parent:
    def __init__(self):
        print("Parent constructor")

class Child(Parent):
    def __init__(self):
        super().__init__()  # Call parent constructor
        print("Child constructor")

c = Child()




# Program with method overloading for area calculation
class Shape:
    def area(self, *args):
        if len(args) == 1:
            # Square
            return args[0] ** 2
        elif len(args) == 2:
            # Rectangle
            return args[0] * args[1]

s = Shape()
print("Area of square (side=5):", s.area(5))
```

```
print("Area of rectangle (4x6):", s.area(4, 6))
```

```
# Program with Degree class hierarchy
class Degree:
    def getDegree(self):
        print("I got a degree")

class Undergraduate(Degree):
    def getDegree(self):
        print("I am an Undergraduate")

class Postgraduate(Degree):
    def getDegree(self):
        print("I am a Postgraduate")

# Create objects
d = Degree()
u = Undergraduate()
p = Postgraduate()

# Call methods
d.getDegree()
u.getDegree()
p.getDegree()
```

```
# Program for student information using single inheritance
class Person:
```

```
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def display(self):
        print(f"Name: {self.name}")
        print(f"Age: {self.age}")

class Student(Person):
    def __init__(self, name, age, rollno):
        super().__init__(name, age)
        self.rollno = rollno

    def display(self):
        super().display()
        print(f"Roll No: {self.rollno}")

s = Student("Alice", 20, "S001")
s.display()



# Program to implement multiple inheritance
class Father:
    def father_quality(self):
        print("Father is strict")

class Mother:
    def mother_quality(self):
        print("Mother is kind")
```

```
class Child(Father, Mother):
    def child_quality(self):
        print("Child is intelligent")

c = Child()
c.father_quality()
c.mother_quality()
c.child_quality()
```

```
# Program to implement multilevel inheritance
class Grandfather:
    def grand_method(self):
        print("Grandfather's method")

class Father(Grandfather):
    def father_method(self):
        print("Father's method")

class Child(Father):
    def child_method(self):
        print("Child's method")

c = Child()
c.grand_method()
c.father_method()
c.child_method()
```

```
# Program to create series from array using pandas
```

```
import pandas as pd
import numpy as np

arr = np.array([10, 20, 30, 40, 50])
series = pd.Series(arr)

print("Pandas Series from array:")
print(series)



# Program to create series from list using pandas
import pandas as pd

data = [100, 200, 300, 400, 500]
series = pd.Series(data, index=['a', 'b', 'c', 'd', 'e'])

print("Pandas Series from list:")
print(series)



# Program to access elements of series
import pandas as pd

series = pd.Series([10, 20, 30, 40, 50], index=['a', 'b', 'c', 'd', 'e'])

print("First element:", series[0])
print("Element at index 'c':", series['c'])
print("First 3 elements:")
print(series.head(3))
print("Last 2 elements:")
print(series.tail(2))
```

```
# Program to create DataFrame using list or dictionary
import pandas as pd

# From list of lists
data = [['Alice', 90], ['Bob', 85], ['Charlie', 78]]
df1 = pd.DataFrame(data, columns=['Name', 'Marks'])

# From dictionary
data = {'Name': ['Alice', 'Bob', 'Charlie'], 'Marks': [90, 85, 78]}
df2 = pd.DataFrame(data)

print("DataFrame from list:")
print(df1)
print("\nDataFrame from dictionary:")
print(df2)



# GUI program with Tkinter
import tkinter as tk

root = tk.Tk()
root.title("My Application")
root.geometry("400x300")

label = tk.Label(root, text="Welcome to Tkinter!")
label.pack(pady=20)

button = tk.Button(root, text="Click Me", command=lambda: print("Button clicked"))
button.pack()
```

```
root.mainloop()
```