2. Python Operators

Program :

```
def arithmetic_operators(a, b):
    print("Arithmetic Operators:")
    print(f"Addition: {a} + {b} = {a + b}")
    print(f"Subtraction: {a} - {b} = {a - b}")
    print(f"Multiplication: {a} * {b} = {a * b}")
    print(f"Division: {a} / {b} = {a / b}")
    print(f"Floor Division: {a} // {b} = {a // b}")
    print(f"Modulus: {a} % {b} = {a % b}")
    print(f"Exponent: {a} ** {b} = {a ** b}\n")

def comparison_operators(a, b):
    print("Comparison Operators:")
    print(f"{a} > {b}: {a > b}")
    print(f"{a} < {b}: {a < b}")
    print(f"{a} == {b}: {a == b}")
    print(f"{a} != {b}: {a != b}")
    print(f"{a} >= {b}: {a >= b}")
    print(f"{a} <= {b}: {a <= b}\n")

def logical_operators(a, b):
    print("Logical Operators:")
    print(f"Both True: {a and b}")
    print(f"Either True: {a or b}")
    print(f"Not a: {not a}\n")
arithmetic_operators(10, 3)
```

```
comparison_operators(10, 3)
logical_operators(True, False)
```

3. Python If-Else Statements

Program :

```
def check_even_odd(num):
    if num % 2 == 0:
        print(f"{num} is Even.")
    else:
        print(f"{num} is Odd.")

def check_positive_negative(num):
    if num > 0:
        print(f"{num} is Positive.")
    elif num < 0:
        print(f"{num} is Negative.")
    else:
        print(f"{num} is Zero.")

def nested_if_example(num):
    if num >= 0:
        if num == 0:
            print("Number is Zero.")
        else:
            print("Number is Positive.")
    else:
        print("Number is Negative.")
check_even_odd(7)
check_positive_negative(-5)
nested_if_example(0)
```

4. Loops in Python

Program :

```
def while_loop_example(n):
    print("While Loop Example:")
    count = 0
    while count < n:
        print("Count:", count)
        count += 1
    print()

def for_loop_example(lst):
    print("For Loop Example:")
    for item in lst:
        print(item)
    print()

def nested_loop_example(n):
    print("Nested Loop Example:")
    for i in range(1, n+1):
        for j in range(1, i+1):
            print(j, end=" ")
        print()
    print()

while_loop_example(3)
for_loop_example(["apple", "banana", "cherry"])
nested_loop_example(4)
```

5. Python Functions

Program :

```
def greet(name):
    """Simple function with parameter"""
    print(f"Hello, {name}!")

def add_numbers(a, b):
    """Function with return value"""
    return a + b

def variable_length_args(*args):
    """Function with variable length arguments"""
    for item in args:
        print(item)

def keyword_args_example(**kwargs):
    """Function with keyword arguments"""
    for key, value in kwargs.items():
        print(f"{key}: {value}")

greet("Alice")
print("Addition:", add_numbers(5, 7))
variable_length_args(1, 2, 3, "Python")
keyword_args_example(name="Alice", age=25)
```

6. Python Modules

Program :

First, create a module:

File 1: calc_module.py

```
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 x / y
    else:
        return "Cannot divide by zero!"
```

File 2: use_calc_module.py

```
import calc_module as calc
def perform_calculations():
    print("Addition:", calc.add(10, 5))
    print("Subtraction:", calc.subtract(10, 5))
    print("Multiplication:", calc.multiply(10, 5))
    print("Division:", calc.divide(10, 0))
perform_calculations()
```

7. Exception Handling

Program :

```
def divide_numbers(a, b):
    try:
        result = a / b
    except ZeroDivisionError:
        print("Error: Cannot divide by zero!")
        return None
    except TypeError:
        print("Error: Invalid input type. Please enter numbers only.")
        return None
    else:
        print(f"The result of {a} / {b} is {result}")
        return result
    finally:
        print("Execution of divide_numbers completed.\n")

def access_list_element(lst, index):
    try:
        print(f"Element at index {index} is {lst[index]}")
    except IndexError:
        print("Error: List index out of range!")
    finally:
        print("Execution of access_list_element completed.\n")

def file_read_demo(filename):
    try:
```

```
        with open(filename, 'r') as file:
            data = file.read()
            print("File contents:\n", data)
    except FileNotFoundError:
        print(f"Error: The file '{filename}' was not found.")
    finally:
        print("Execution of file_read_demo completed.\n")

divide_numbers(10, 2)
divide_numbers(5, 0)
divide_numbers(5, 'a')

sample_list = [1, 2, 3, 4, 5]
access_list_element(sample_list, 3)
access_list_element(sample_list, 10)

file_read_demo('sample.txt')
```

8. CSV File Handling

Program :

```
import csv

def write_csv(filename, data):
    """Writes data to a CSV file."""
    try:
        with open(filename, mode='w', newline='') as file:
            writer = csv.writer(file)
            writer.writerow(["Name", "Age", "City"])  # Header
            writer.writerows(data)
        print(f"Data written successfully to {filename}")
    except Exception as e:
        print("Error writing to CSV:", e)

def read_csv(filename):
    """Reads data from a CSV file."""
    try:
        with open(filename, mode='r') as file:
            reader = csv.reader(file)
            for row in reader:
                print(row)
    except FileNotFoundError:
        print(f"Error: The file '{filename}' was not found.")

def append_csv(filename, new_data):
    """Appends new data to an existing CSV file."""
```

```
    try:
        with open(filename, mode='a', newline='') as file:
            writer = csv.writer(file)
            writer.writerows(new_data)
        print(f"Data appended successfully to {filename}")
    except Exception as e:
        print("Error appending to CSV:", e)

people = [
    ["Alice", 28, "New York"],
    ["Bob", 24, "Los Angeles"],
    ["Charlie", 30, "Chicago"]
]

new_people = [
    ["David", 22, "Houston"],
    ["Eve", 29, "Seattle"]
]

filename = "people.csv"

write_csv(filename, people)
read_csv(filename)
append_csv(filename, new_people)
read_csv(filename)
```