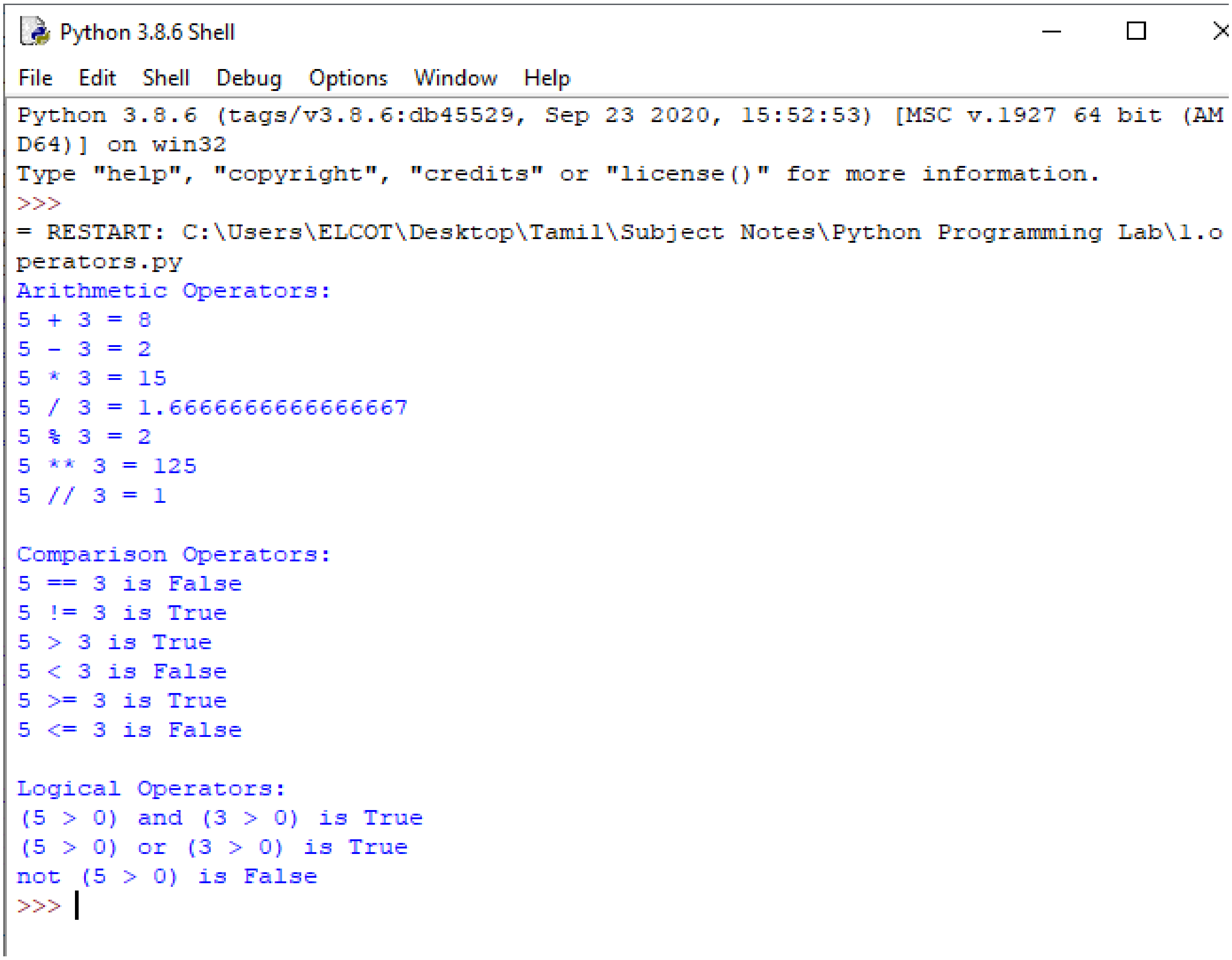
HBCS22I02 – PYTHON PROGRAMMING LAB

Program 1: **Write a Python program to implement the different operators.**

```
a = 5
b = 3
# Arithmetic Operators
print("Arithmetic Operators:")
print(f"{a} + {b} = {a + b}") # Addition
print(f"{a} - {b} = {a - b}") # Subtraction
print(f"{a} * {b} = {a * b}") # Multiplication
print(f"{a} / {b} = {a / b}") # Division
print(f"{a} % {b} = {a % b}") # Modulus
print(f"{a} ** {b} = {a ** b}") # Exponentiation
print(f"{a} // {b} = {a // b}") # Floor Division
# Comparison Operators
print("\nComparison Operators:")
print(f"{a} == {b} is {a == b}") # Equal
print(f"{a} != {b} is {a != b}") # Not equal
print(f"{a} > {b} is {a > b}") # Greater than
print(f"{a} < {b} is {a < b}") # Less than
print(f"{a} >= {b} is {a >= b}") # Greater than or equal to
print(f"{a} <= {b} is {a <= b}") # Less than or equal to
# Logical Operators
print("\nLogical Operators:")
print(f"({a} > 0) and ({b} > 0) is {(a > 0) and (b > 0)}") # Logical AND
print(f"({a} > 0) or ({b} > 0) is {(a > 0) or (b > 0)}") # Logical OR
print(f"not ({a} > 0) is {not (a > 0)}") # Logical NOT
```

OUTPUT:

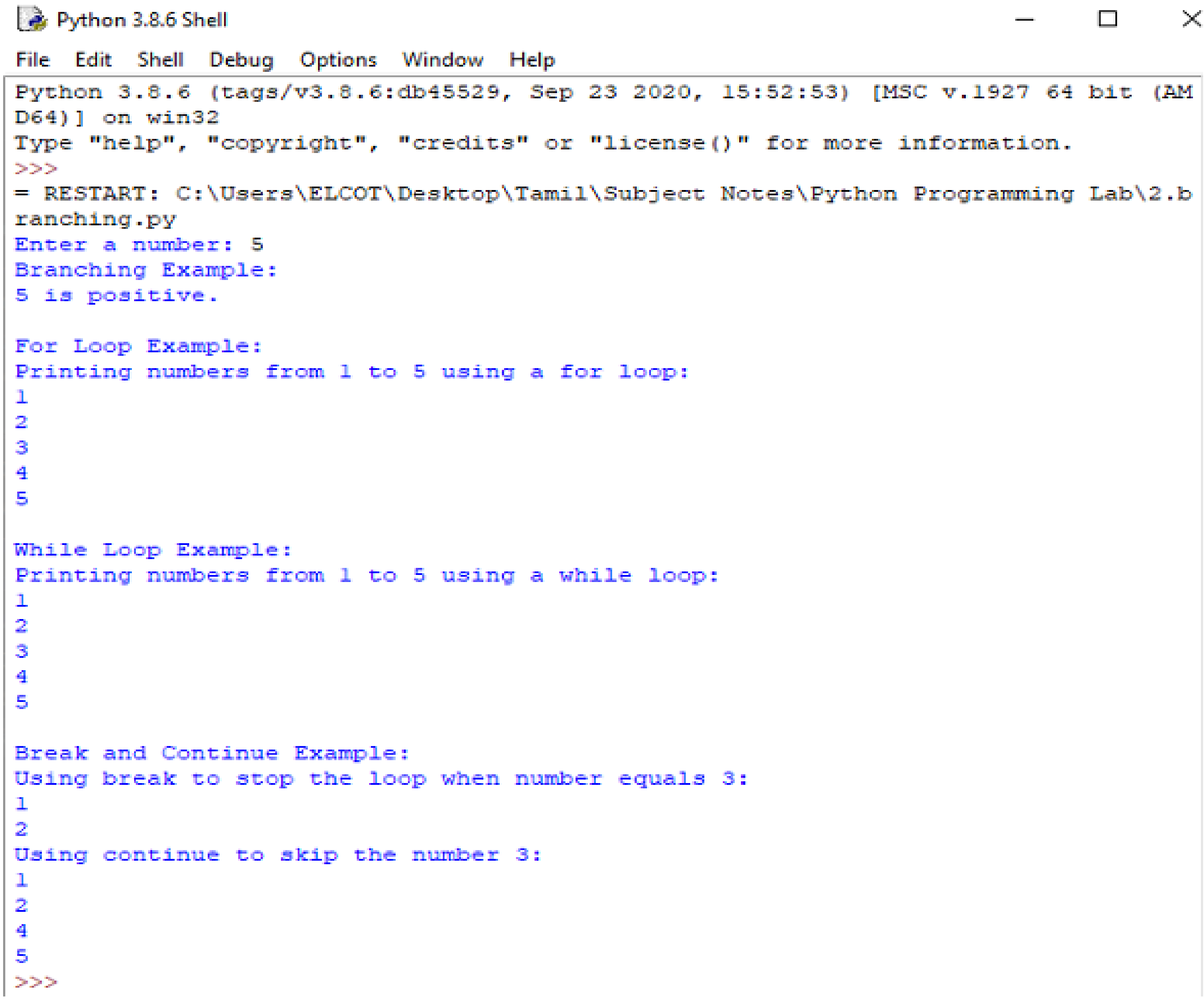


```
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\ELCOT\Desktop\Tamil\Subject Notes\Python Programming Lab\1.o
perators.py
Arithmetic Operators:
5 + 3 = 8
5 - 3 = 2
5 * 3 = 15
5 / 3 = 1.6666666666666667
5 % 3 = 2
5 ** 3 = 125
5 // 3 = 1

Comparison Operators:
5 == 3 is False
5 != 3 is True
5 > 3 is True
5 < 3 is False
5 >= 3 is True
5 <= 3 is False

Logical Operators:
(5 > 0) and (3 > 0) is True
(5 > 0) or (3 > 0) is True
not (5 > 0) is False
>>> 
```

Program 2: **Write a Python program to implement branching and looping constructs**

```
number = int(input("Enter a number: "))
print("Branching Example:")
if number > 0:
    print(f"{number} is positive.")
elif number < 0:
    print(f"{number} is negative.")
else:
    print(f"{number} is zero.")
# Looping with for loop
print("\nFor Loop Example:")
print("Printing numbers from 1 to 5 using a for loop:")
for i in range(1, 6):
    print(i)
# Looping with while loop
print("\nWhile Loop Example:")
print("Printing numbers from 1 to 5 using a while loop:")
i = 1
while i <= 5:
    print(i)
    i += 1
# Using break and continue
print("\nBreak and Continue Example:")
print("Using break to stop the loop when number equals 3:")
for i in range(1, 6):
    if i == 3:
        break
    print(i)

print("Using continue to skip the number 3:")
for i in range(1, 6):
    if i == 3:
        continue
    print(i)
```

OUTPUT:



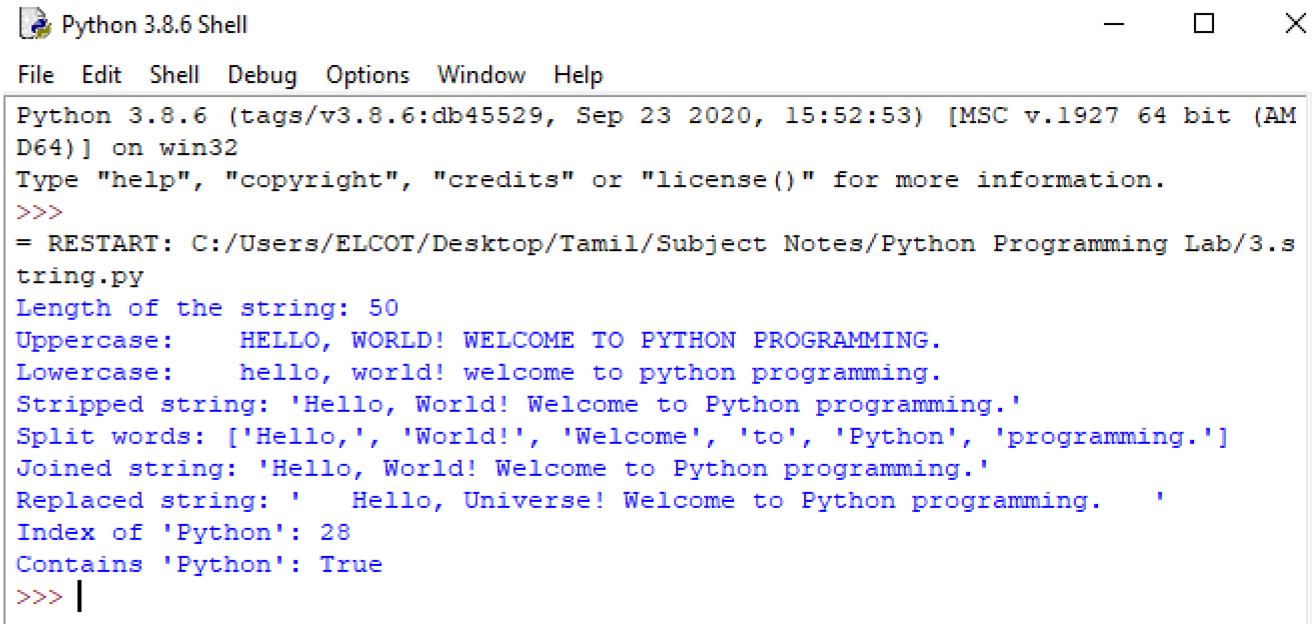
```
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\ELCOT\Desktop\Tamil\Subject Notes\Python Programming Lab\2.b
ranching.py
Enter a number: 5
Branching Example:
5 is positive.

For Loop Example:
Printing numbers from 1 to 5 using a for loop:
1
2
3
4
5

While Loop Example:
Printing numbers from 1 to 5 using a while loop:
1
2
3
4
5

Break and Continue Example:
Using break to stop the loop when number equals 3:
1
2
Using continue to skip the number 3:
1
2
4
5
>>>
```

Program 3: Python program to implement string operations and functions

```
my_string = "   Hello, World! Welcome to Python programming.   "
   # 1. String Length
print(f"Length of the string: {len(my_string)}")
   # 2. String Uppercase and Lowercase
print (f"Uppercase: {my_string.upper()}")
print (f"Lowercase: {my_string.lower()}")
   # 3. Strip Whitespace
print (f"Stripped string: '{my_string.strip()}'")
   # 4. Split String into a List
```

```
words = my_string.split()
print (f"Split words: {words}")
   # 5. Join List into a String
joined_string = ' '.join(words)
print(f"Joined string: '{joined_string}'")
   # 6. Replace Substring
replaced_string = my_string.replace('World', 'Universe')
print(f"Replaced string: '{replaced_string}'")
   # 7. Find Substring
index = my_string.find('Python')
print(f"Index of 'Python': {index}")
   # 8. Check if String Contains Substring
contains_python = 'Python' in my_string
print(f"Contains 'Python': {contains_python}")
```

OUTPUT:

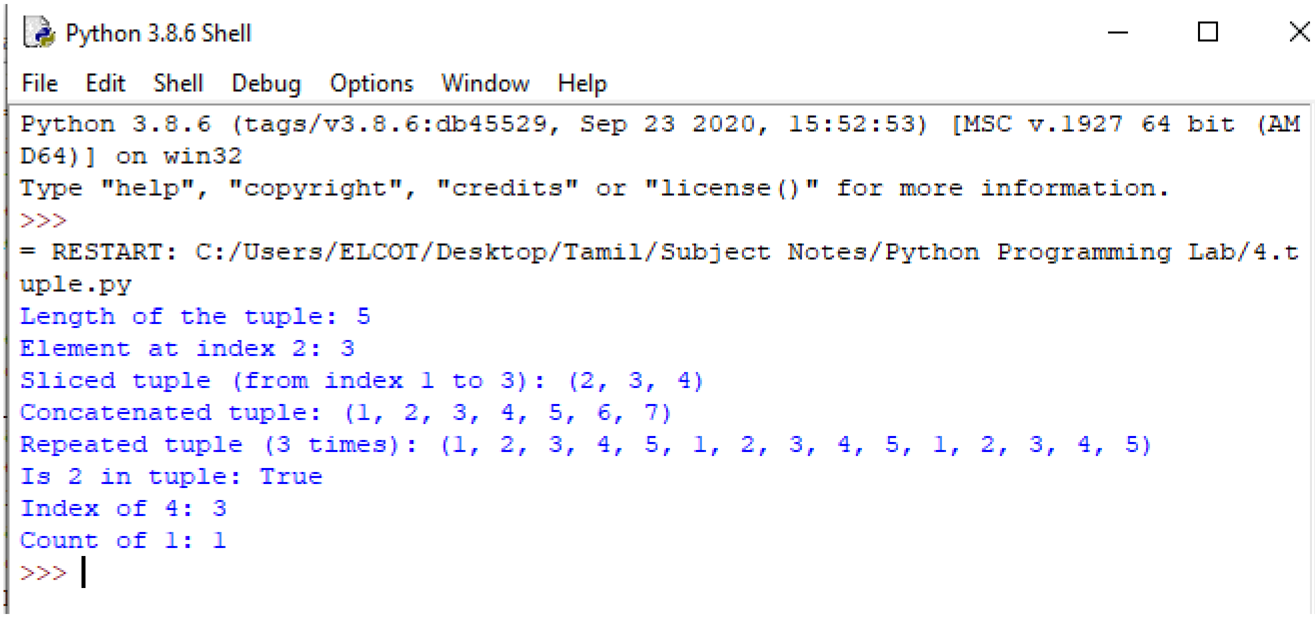


```
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
= RESTART: C:/Users/ELCOT/Desktop/Tamil/Subject Notes/Python Programming Lab/3.s
tring.py
Length of the string: 50
Uppercase:    HELLO, WORLD! WELCOME TO PYTHON PROGRAMMING.
Lowercase:    hello, world! welcome to python programming.
Stripped string: 'Hello, World! Welcome to Python programming.'
Split words: ['Hello,', 'World!', 'Welcome', 'to', 'Python', 'programming.']
Joined string: 'Hello, World! Welcome to Python programming.'
Replaced string: '   Hello, Universe! Welcome to Python programming.   '
Index of 'Python': 28
Contains 'Python': True
>>> 
```

Program 4: Create a Python program to implement various operations on tuple

```
my_tuple = (1, 2, 3, 4, 5)
mixed_tuple = (1, 'hello', 3.14, True)
   # 1. Tuple Length
print(f"Length of the tuple: {len(my_tuple)}")
   # 2. Access Elements by Index
print(f"Element at index 2: {my_tuple[2]}")
   # 3. Slicing Tuples
sliced_tuple = my_tuple[1:4]
print(f"Sliced tuple (from index 1 to 3): {sliced_tuple}")
   # 4. Concatenate Tuples
concatenated_tuple = my_tuple + (6, 7)
print(f"Concatenated tuple: {concatenated_tuple}")
   # 5. Repeat Tuples
repeated_tuple = my_tuple * 3
print(f"Repeated tuple (3 times): {repeated_tuple}")
   # 6. Check Membership
is_two_in_tuple = 2 in my_tuple
print(f"Is 2 in tuple: {is_two_in_tuple}")
   # 7. Find Index of an Element
index_of_4 = my_tuple.index(4)
print(f"Index of 4: {index_of_4}")
   # 8. Count Occurrences of an Element
count_of_1 = my_tuple.count(1)
print(f"Count of 1: {count_of_1}")
```

OUTPUT:



```
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/ELCOT/Desktop/Tamil/Subject Notes/Python Programming Lab/4.t
uple.py
Length of the tuple: 5
Element at index 2: 3
Sliced tuple (from index 1 to 3): (2, 3, 4)
Concatenated tuple: (1, 2, 3, 4, 5, 6, 7)
Repeated tuple (3 times): (1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
Is 2 in tuple: True
Index of 4: 3
Count of 1: 1
>>> 
```

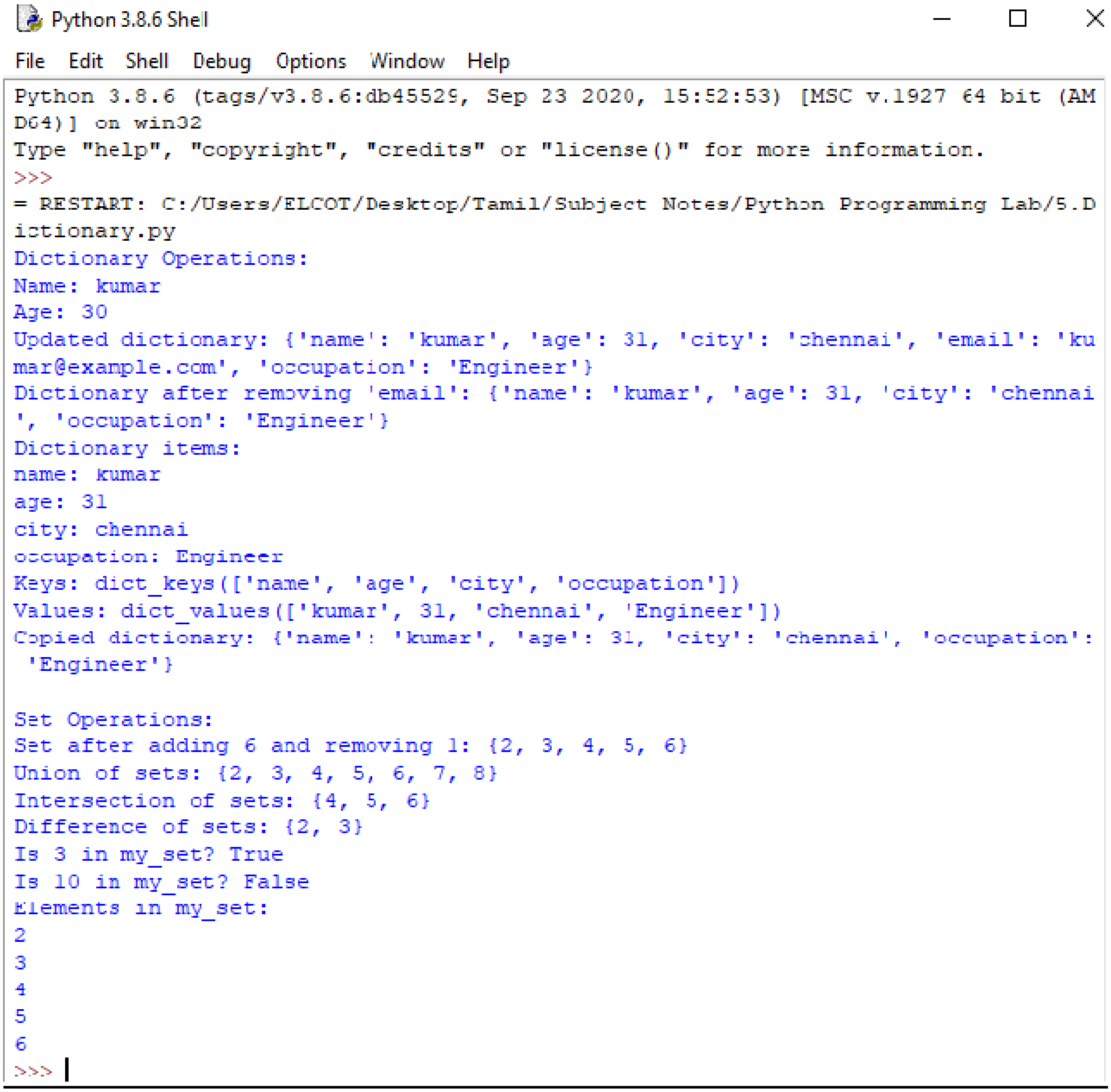
Program 5: **Program to implement Dictionary and sets**

```
# Dictionary Operations
    print("Dictionary Operations:")
    # Creating a dictionary
    my_dict = {
        'name': 'kumar',
        'age': 30,
        'city': 'chennai',
        'email': 'kumar@example.com'
    }
    #  Accessing Values
    print(f"Name: {my_dict['name']}")
    print(f"Age: {my_dict.get('age')}")
    # Adding or Updating Entries
    my_dict['age'] = 31  # Update
    my_dict['occupation'] = 'Engineer'  # Add
    print(f"Updated dictionary: {my_dict}")
    # Removing Entries
```

```
del my_dict['email']
print(f"Dictionary after removing 'email': {my_dict}")
#  Iterating Through Dictionary
print("Dictionary items:")
for key, value in my_dict.items():
    print(f"{key}: {value}")
#  Dictionary Keys and Values
print(f"Keys: {my_dict.keys()}")
print(f"Values: {my_dict.values()}")
# Dictionary Copy
dict_copy = my_dict.copy()
print(f"Copied dictionary: {dict_copy}")

# Set Operations
print("\nSet Operations:")
# Creating a set
my_set = {1, 2, 3, 4, 5}
another_set = {4, 5, 6, 7, 8}
#  Adding and Removing Elements
my_set.add(6)
my_set.remove(1)
print(f"Set after adding 6 and removing 1: {my_set}")
# Union of Sets
union_set = my_set.union(another_set)
print(f"Union of sets: {union_set}")
# Intersection of Sets
intersection_set = my_set.intersection(another_set)
print(f"Intersection of sets: {intersection_set}")
# Difference of Sets
difference_set = my_set.difference(another_set)
print(f"Difference of sets: {difference_set}")
```

```
# Checking Membership
print(f"Is 3 in my_set? {3 in my_set}")
print(f"Is 10 in my_set? {10 in my_set}")
#  Iterating Through a Set
print("Elements in my_set:")
for element in my_set:
    print(element)
```

OUTPUT:

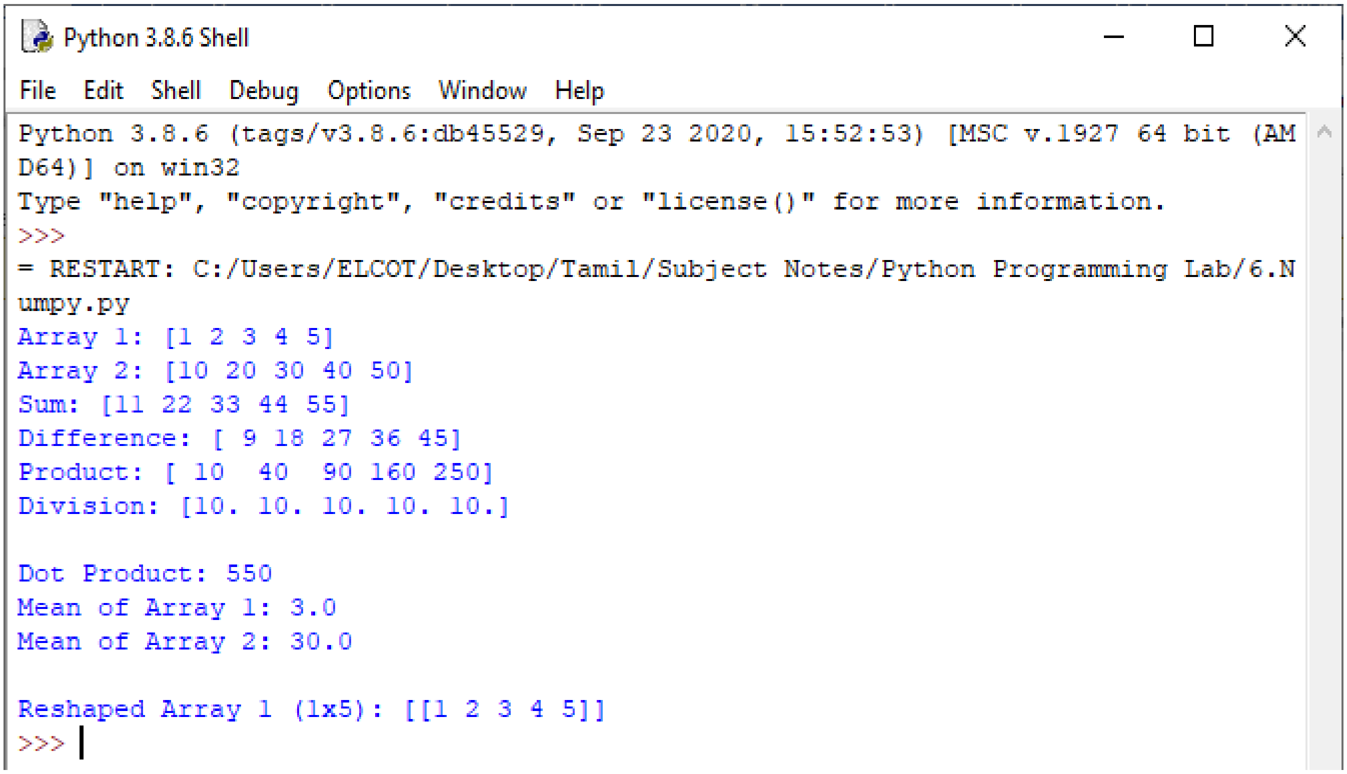


```
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/ELCOT/Desktop/Tamil/Subject Notes/Python Programming Lab/5.D
ictionary.py
Dictionary Operations:
Name: kumar
Age: 30
Updated dictionary: {'name': 'kumar', 'age': 31, 'city': 'chennai', 'email': 'ku
mar@example.com', 'occupation': 'Engineer'}
Dictionary after removing 'email': {'name': 'kumar', 'age': 31, 'city': 'chennai
', 'occupation': 'Engineer'}
Dictionary items:
name: kumar
age: 31
city: chennai
occupation: Engineer
Keys: dict_keys(['name', 'age', 'city', 'occupation'])
Values: dict_values(['kumar', 31, 'chennai', 'Engineer'])
Copied dictionary: {'name': 'kumar', 'age': 31, 'city': 'chennai', 'occupation':
 'Engineer'}

Set Operations:
Set after adding 6 and removing 1: {2, 3, 4, 5, 6}
Union of sets: {2, 3, 4, 5, 6, 7, 8}
Intersection of sets: {4, 5, 6}
Difference of sets: {2, 3}
Is 3 in my_set? True
Is 10 in my_set? False
Elements in my_set:
2
3
4
5
6
>>>
```

Program 6: **Program for operations on NumPy arrays.**

```
import numpy as np

# Create two arrays
array1 = np.array([1, 2, 3, 4, 5])
array2 = np.array([10, 20, 30, 40, 50])

# Basic arithmetic operations
sum_array = array1 + array2
difference_array = array2 - array1
product_array = array1 * array2
division_array = array2 / array1

# Print the results
print("Array 1:", array1)
print("Array 2:", array2)
print("Sum:", sum_array)
print("Difference:", difference_array)
print("Product:", product_array)
print("Division:", division_array)

# Additional operations
dot_product = np.dot(array1, array2)
mean_array1 = np.mean(array1)
mean_array2 = np.mean(array2)

print("\nDot Product:", dot_product)
print("Mean of Array 1:", mean_array1)
print("Mean of Array 2:", mean_array2)

# Reshaping arrays
reshaped_array = array1.reshape(1, 5)
print("\nReshaped Array 1 (1x5):", reshaped_array)
```

OUTPUT:

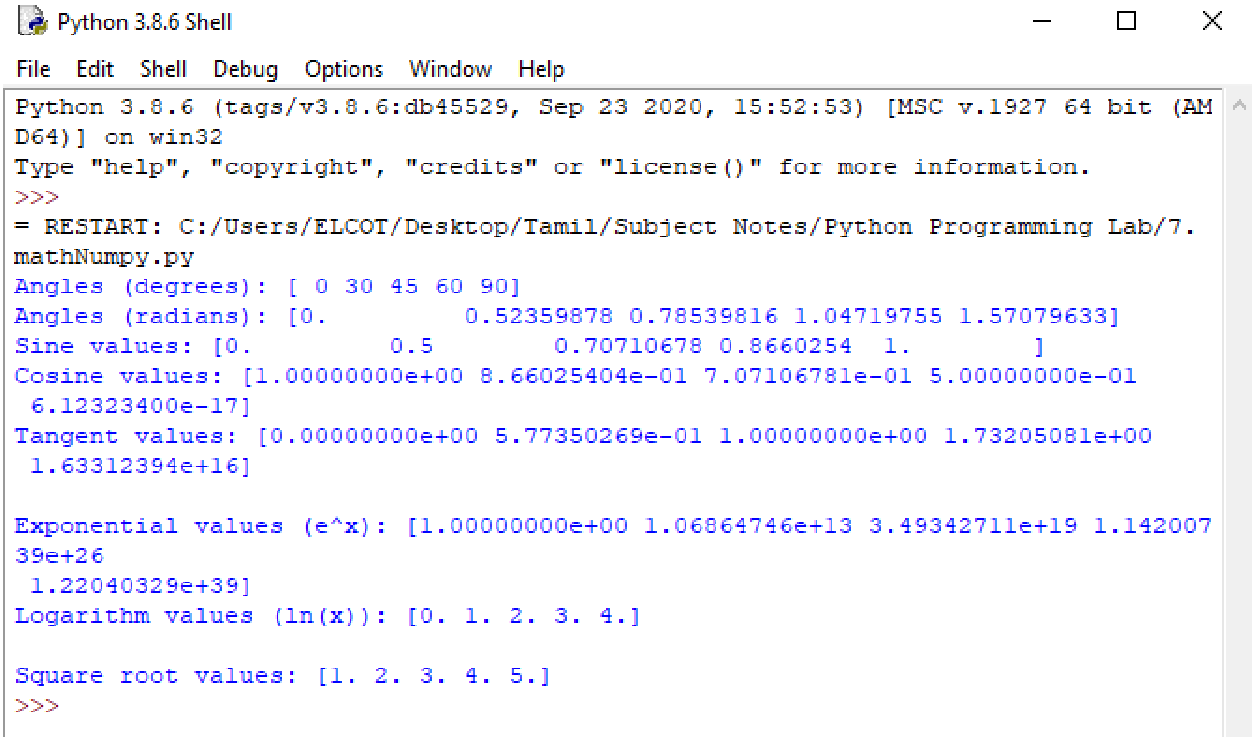


```
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/ELCOT/Desktop/Tamil/Subject Notes/Python Programming Lab/6.N
umpy.py
Array 1: [1 2 3 4 5]
Array 2: [10 20 30 40 50]
Sum: [11 22 33 44 55]
Difference: [ 9 18 27 36 45]
Product: [ 10  40  90 160 250]
Division: [10. 10. 10. 10. 10.]

Dot Product: 550
Mean of Array 1: 3.0
Mean of Array 2: 30.0

Reshaped Array 1 (1x5): [[1 2 3 4 5]]
>>> 
```

Program 7: **Program to test math functions using NumPy**

```
import numpy as np

# Create an array of angles in degrees
angles_degrees = np.array([0, 30, 45, 60, 90])

# Convert angles from degrees to radians
angles_radians = np.radians(angles_degrees)

# Test trigonometric functions
sin_values = np.sin(angles_radians)
cos_values = np.cos(angles_radians)
tan_values = np.tan(angles_radians)

# Print the results
print("Angles (degrees):", angles_degrees)
print("Angles (radians):", angles_radians)
print("Sine values:", sin_values)
print("Cosine values:", cos_values)
print("Tangent values:", tan_values)

# Test exponential and logarithmic functions
exponential_values = np.exp(angles_degrees)
logarithm_values = np.log(np.array([1, np.e, np.e**2, np.e**3, np.e**4]))

print("\nExponential values (e^x):", exponential_values)
print("Logarithm values (ln(x)):", logarithm_values)

# Test square root function
sqrt_values = np.sqrt(np.array([1, 4, 9, 16, 25]))

print("\nSquare root values:", sqrt_values)
```

OUTPUT:



```
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
= RESTART: C:/Users/ELCOT/Desktop/Tamil/Subject Notes/Python Programming Lab/7.
mathNumpy.py
Angles (degrees): [ 0 30 45 60 90]
Angles (radians): [0.         0.52359878 0.78539816 1.04719755 1.57079633]
Sine values: [0.         0.5        0.70710678 0.8660254  1.        ]
Cosine values: [1.00000000e+00 8.66025404e-01 7.07106781e-01 5.00000000e-01
 6.12323400e-17]
Tangent values: [0.00000000e+00 5.77350269e-01 1.00000000e+00 1.73205081e+00
 1.63312394e+16]

Exponential values (e^x): [1.00000000e+00 1.06864746e+13 3.49342711e+19 1.142007
39e+26
 1.22040329e+39]
Logarithm values (ln(x)): [0. 1. 2. 3. 4.]

Square root values: [1. 2. 3. 4. 5.]
>>> 
```