K. J. Somaiya Polytechnic, Mumbai-77

Batch No: C1

Enrollment No: FCOW19118

Experiment No.: 01

Experiment Name: Write python programs to understand Expressions, Variables, basic Math operations, Strings, Basic String Operation & String Methods. (Minimum three Programs based on math operations, Strings).

Experiment No: 01

Experiment Name: Write python programs to understand Expressions, Variables, basic Math operations, Strings, Basic String Operation & String Methods. (Minimum three Programs based on math operations, Strings).

Course Outcome:

O18RA72.1	Understand basics of Python programming.

Theory:

A. Operators In Python

Operators are the constructs which can manipulate the value of operands. Consider the expression 4 + 5 = 9. Here, 4 and 5 are called operands and + is called operator.

Types of Operators:

Python language supports the following types of operators:

- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

1. Python Arithmetic Operators

Assume variable a holds 10 and variable b holds 20, then:

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b=10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity) −	9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0

2. Python Comparison Operators

These operators compare the values on either side of them and decide the relation among them. They are also called Relational operators.

Assume variable a holds 10 and variable b holds 20, then:

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	(a != b) is true.
<>	If values of two operands are not equal, then condition becomes true.	(a <> b) is true. This is similar to != operator.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

3. Python Assignment Operators

Assume variable a holds 10 and variable b holds 20, then :

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+=	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-=	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*=	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/=	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / a
%=	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**=	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//=	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

4. Python Bitwise Operators

Bitwise operator works on bits and performs bit by bit operation. Assume if a = 60; and b = 13; Now in the binary format their values will be 0011 1100 and 0000 1101 respectively. Following table lists out the bitwise operators supported by Python language with an example each in those, we use the above two variables (a and b) as operands :

a = 0011 1100

b = 0000 1101

----

a&b = 0000 1100   a|b = 0011 1101   a^b = 0011 0001   ~a = 1100 0011

There are following Bitwise operators supported by Python language which are as follows :

Operator	Description	Example
& Binary AND	Operator copies a bit to the result if it exists in both operands	(a & b) (means 0000 1100)
| Binary OR	It copies a bit if it exists in either operand.	(a | b) = 61 (means 0011 1101)
^ Binary XOR	It copies the bit if it is set in one operand but not both.	(a ^ b) = 49 (means 0011 0001)
~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	(~a) = -61 (means 1100 0011 in 2's complement form due to a signed binary number.
<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	a << 2 = 240 (means 1111 0000)
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	a >> 2 = 15 (means 0000 1111)

5. Python Logical Operators

There are following logical operators supported by Python language. Assume variable a holds 10 and variable b holds 20 then:

Operator	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a and b) is true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.

6. Python Membership Operators

Python’s membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators as explained below :

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

7. Python Identity Operators

Identity operators compare the memory locations of two objects. There are two Identity operators explained below:

Operator	Description	Example
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here is results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here is not results in 1 if id(x) is not equal to id(y).

8. Python Operators Precedence

The following table lists all operators from highest precedence to lowest.

Sr. No.	Operator & Description
1	** Exponentiation (raise to the power)
2	~ + - Complement, unary plus and minus (method names for the last two are +@ and -@)
3	* / % // Multiply, divide, modulo and floor division
4	+ - Addition and subtraction
5	>> << Right and left bitwise shift
6	& Bitwise 'AND'
7	^ | Bitwise exclusive `OR' and regular `OR'
8	<= < > >= Comparison operators
9	<> == != Equality operators
10	= %= /= //= -= += *= **= Assignment operators
11	is is not Identity operators
12	in not in Membership operators
13	not or and Logical operators

B. What is String in Python?

- A string is a sequence of characters.
- A character is simply a symbol. For example, the English language has 26 characters.
- Computers do not deal with characters, they deal with numbers (binary). Even though you may see characters on your screen, internally it is stored and manipulated as a combination of 0s and 1s.
- This conversion of character to a number is called encoding, and the reverse process is decoding. ASCII and Unicode are some of the popular encodings used.
- In Python, a string is a sequence of Unicode characters. Unicode was introduced to include every character in all languages and bring uniformity in encoding. You can learn about Unicode from Python Unicode.

C. Python String Operations

1) Concatenation and Repetition

- There are many operations that can be performed with strings which makes it one of the most used data types in Python.
- Joining of two or more strings into a single one is called concatenation.
- The **+** operator does this in Python. Simply writing two string literals together also concatenates them.
- The ***** operator can be used to repeat the string for a given number of times.

```
# Python String Operations
str1 = 'Hello'
str2 ='World!'
# using +
print('str1 + str2 = ', str1 + str2)
# using *
print('str1 * 3 =', str1 * 3)
```

When we run the above program, we get the following output:

```
str1 + str2 =  HelloWorld!
str1 * 3 = HelloHelloHello
```

Writing two string literals together also concatenates them like + operator. If we want to concatenate strings in different lines, we can use parentheses.

E.g1.: >>> 'Hello "World!'

O/P: 'Hello World!'

E.g2.: >>> s = ('Hello '
...   'World')
>>> s

O/P: 'Hello World!'

2) Iterating Through a string

We can iterate through a string using a for loop. Here is an example to count the number of 'l's in a string.

# Iterating through a string

```
count = 0
for letter in 'Hello World':
    if(letter == 'l'):
        count += 1
print(count,'letters found')
```

O/p:

```
3 letters found
```

3) String Membership operator [‘in’ and ‘not in’]

We can test if a substring exists within a string or not, using the keyword in.

>>> 'a' in 'program'	True
>>> 'at' not in 'battle'	False

4) Slice and Range Slice

Python slicing is about obtaining a sub-string from the given string by slicing it respectively from start to end.

Python slicing can be done in two ways.

- slice() Constructor
- Extending Indexing

a. slice() Constructor

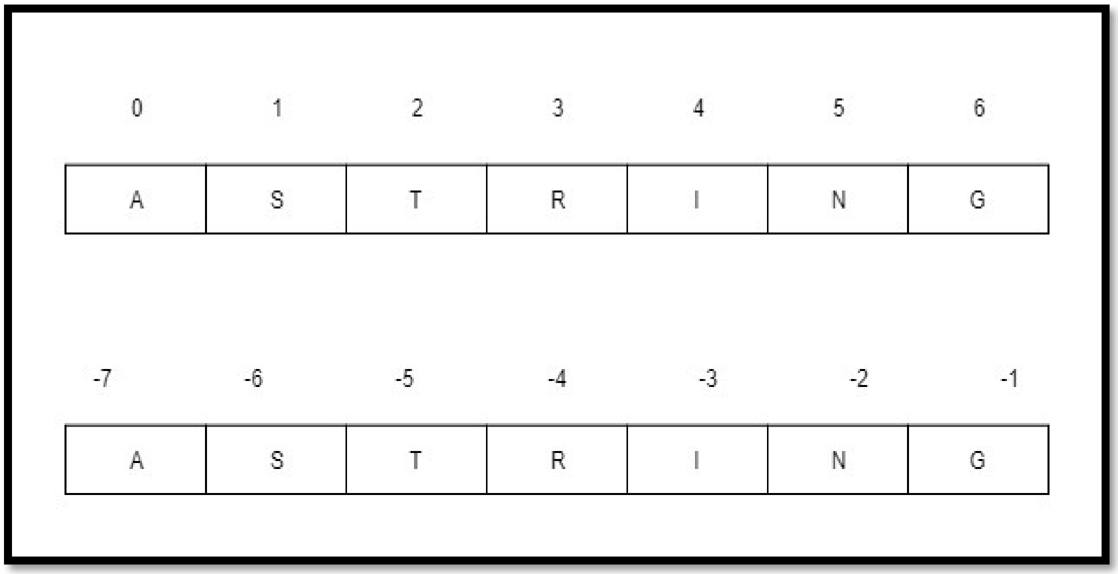
The slice() constructor creates a slice object representing the set of indices specified by range(start, stop, step).

Syntax: 1) slice(stop) 2) slice(start, stop, step)

Parameters:

start: Starting index where the slicing of object starts.
stop: Ending index where the slicing of object stops.
step: It is an optional argument that determines the increment between each index for slicing.
Return Type: Returns a sliced object containing elements in the given range only.

Index tracker for positive and negative index :
Negative comes into considers when tracking the string in reverse.



Example:

```
# Python program to demonstrate
# string slicing
# String slicing
String ='ASTRING'
# Using slice constructor
s1 = slice(3)
s2 = slice(1, 5, 2)
s3 = slice(-1, -12, -2)
print("String slicing")
print(String[s1])
print(String[s2])
print(String[s3])
```

Output:

```
String slicing
AST
SR
GITA
```

b. Extending indexing

In Python, indexing syntax can be used as a substitute for the slice object. This is an easy and convenient way to slice a string both syntax wise and execution wise.

Syntax:

```
string[start:end:step]
```

start, end and step have the same mechanism as slice() constructor.

Example:

```
# Python program to demonstrate
# string slicing
# String slicing
String ='ASTRING'
# Using indexing sequence
print(String[:3])
print(String[1:5:2])
print(String[-1:-12:-2])
# Prints string in reverse
print("\nReverse String")
print(String[::-1])
```

Output:

```
AST
SR
GITA
Reverse String
GNIRTSA
```

D. Methods for string processing

1) Python String capitalize()

Converts first character to Capital Letter

2) Python String center()

Pads string with specified character

3) Python String count()

returns occurrences of substring in string

4) **Python String find()**

Returns the index of first occurrence of substring

5) **Python String index()**

Returns Index of Substring

6) **Python String isalnum()**

Checks Alphanumeric Character

7) **Python String isalpha()**

Checks if All Characters are Alphabets

8) **Python String isdecimal()**

Checks Decimal Characters

9) **Python String isdigit()**

Checks Digit Characters

10) **Python String isidentifier()**

Checks for Valid Identifier

11) **Python String islower()**

Checks if all Alphabets in a String are Lowercase

12) **Python String isnumeric()**

Checks Numeric Characters

13) **Python String istitle()**

Checks for Titlecased String

14) **Python String isupper()**

returns if all characters are uppercase characters

15) Python String lower()

returns lowercased string

16) Python String replace()

Replaces Substring Inside

17) Python String split()

Splits String from Left

18) Python String title()

Returns a Title Cased String

19) Python String upper()

returns uppercased string

Implementation and Output

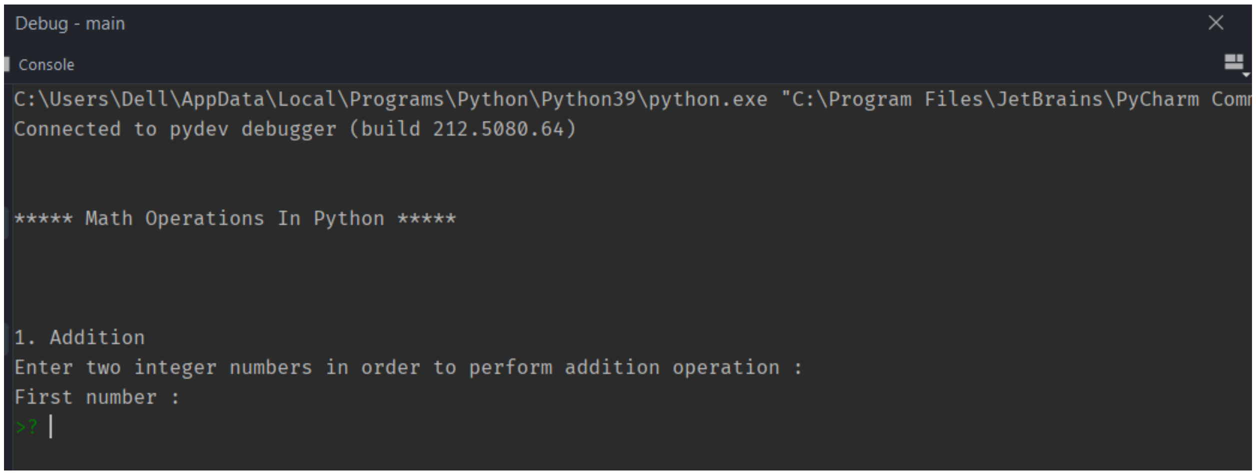
1. Math Operations in Python

Program:

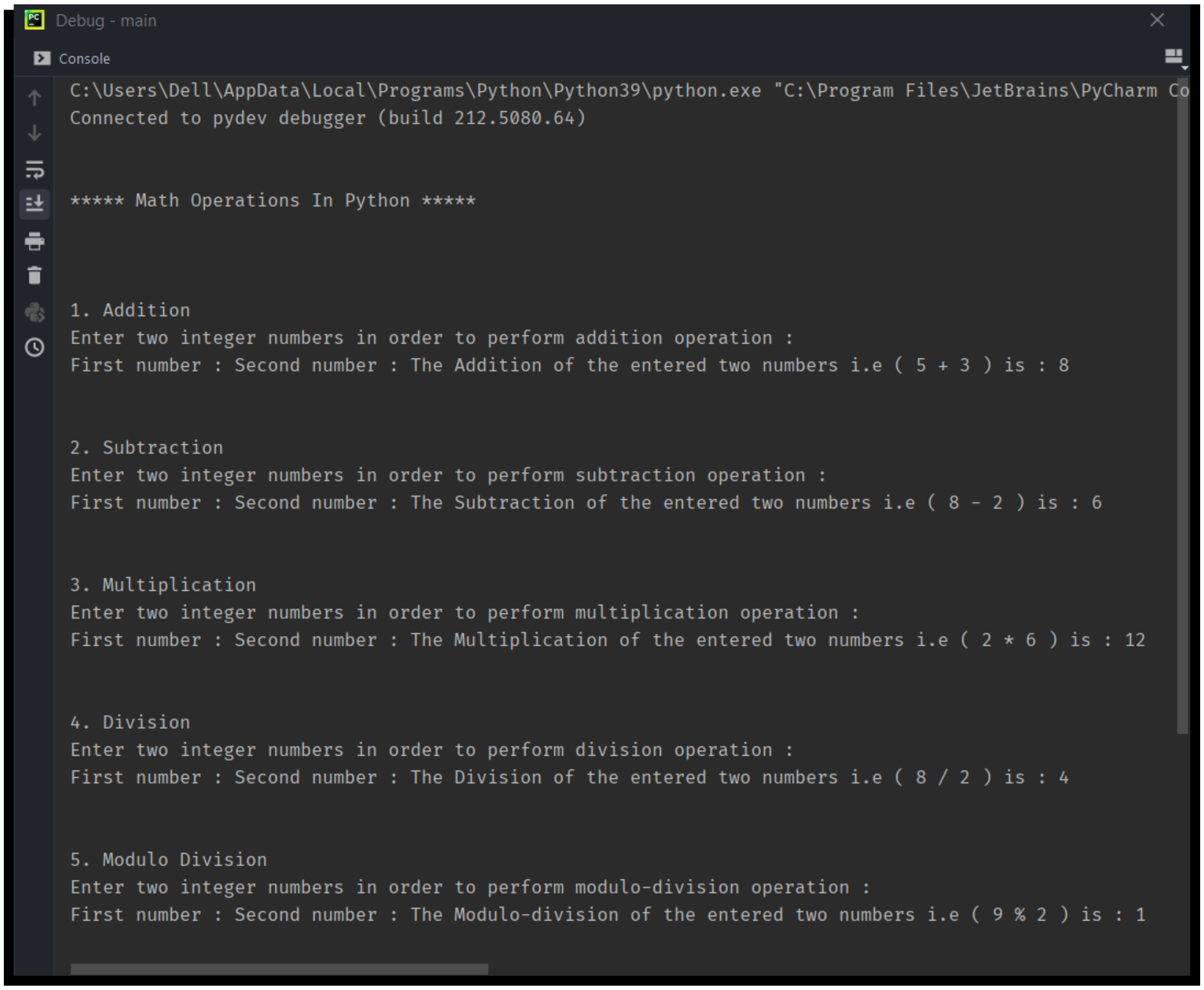
```
print("\n")
print("***** Math Operations In Python *****")
print("\n\n")
print("1. Addition")
print("Enter two integer numbers in order to perform addition operation :")
number_1=int(input("First number : "))
number_2=int(input("Second number : "))
addition=number_1+number_2
```

```
print("The Addition of the entered two numbers i.e ( %d + %d ) is : %d "%(number_1,number_2,addition))
print("\n")
print("2. Subtraction")
print("Enter two integer numbers in order to perform subtraction operation :")
number_1=int(input("First number : "))
number_2=int(input("Second number : "))
subtraction=number_1-number_2
print("The Subtraction of the entered two numbers i.e ( %d - %d ) is : %d "%(number_1,number_2,subtraction))
print("\n")
print("3. Multiplication")
print("Enter two integer numbers in order to perform multiplication operation :")
number_1=int(input("First number : "))
number_2=int(input("Second number : "))
multiplication=number_1*number_2
print("The Multiplication of the entered two numbers i.e ( %d * %d ) is : %d "%(number_1,number_2,multiplication))
print("\n")
print("4. Division")
print("Enter two integer numbers in order to perform division operation :")
number_1=int(input("First number : "))
number_2=int(input("Second number : "))
division=number_1/number_2
print("The Division of the entered two numbers i.e ( %d / %d ) is : %d "%(number_1,number_2,division))
print("\n")
print("5. Modulo Division")
print("Enter two integer numbers in order to perform modulo-division operation :")
number_1=int(input("First number : "))
number_2=int(input("Second number : "))
modulo_division=number_1%number_2
print("The Modulo-division of the entered two numbers i.e ( %d %% %d ) is : %d "%(number_1,number_2,modulo_division))
print("\n")
print("6. Exponentiation")
print("Enter two integer numbers in order to perform exponentiation operation :")
number_1=int(input("First number as a base : "))
number_2=int(input("Second number as a exponent or power : "))
exponentiation=number_1**number_2
print("The exponentiation of the entered two numbers i.e ( %d ** %d ) is : %d "%(number_1,number_2,exponentiation))
print("\n")
print("7.Floor Division")
print("Enter two integer numbers in order to perform floor-division operation :")
number_1=int(input("First number : "))
number_2=int(input("Second number : "))
floor_division=number_1//number_2
print("The Floor-division of the entered two numbers i.e ( %d // %d ) is : %d "%(number_1,number_2,floor_division))
```

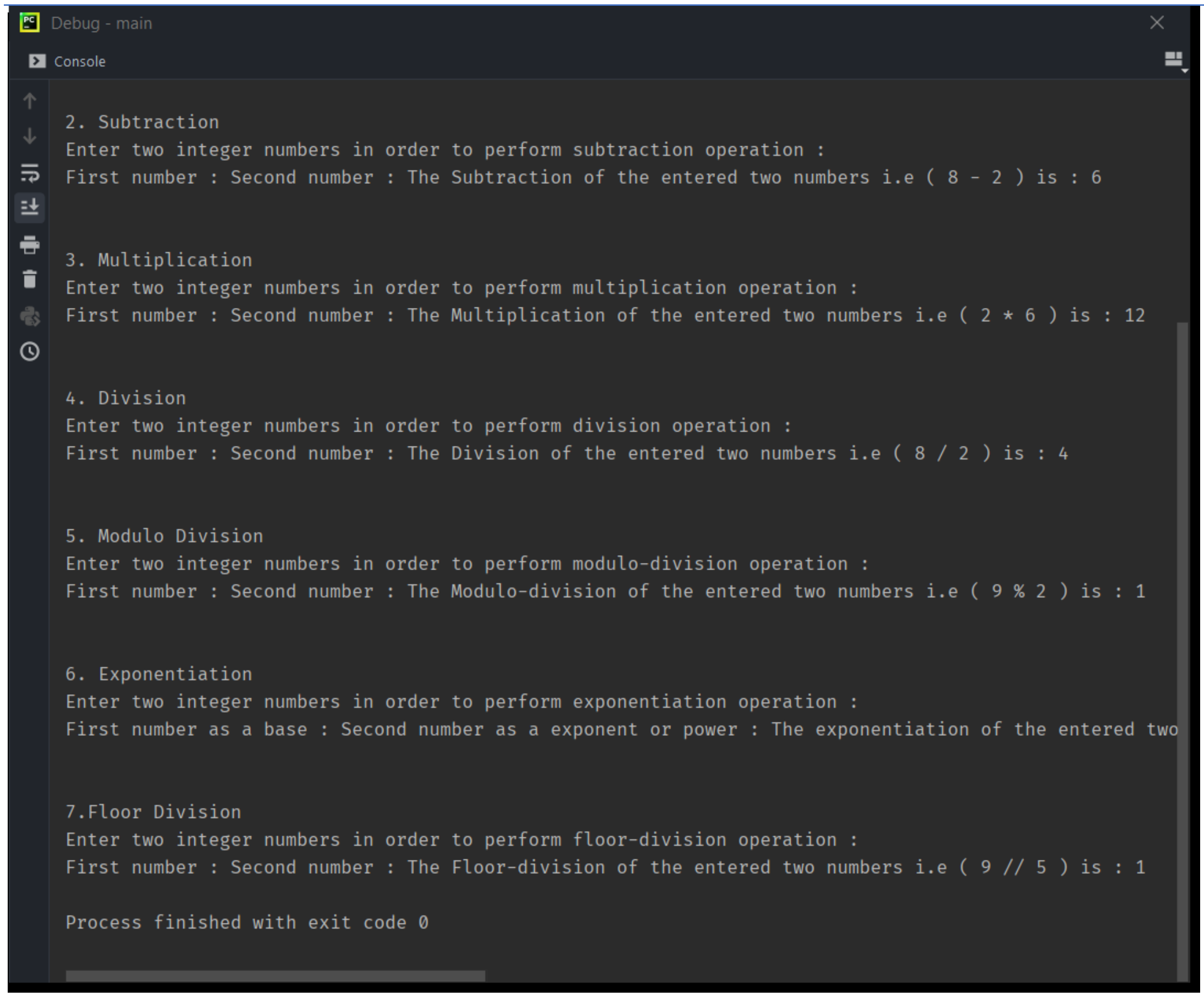
Output:



```
Debug - main
Console
C:\Users\Dell\AppData\Local\Programs\Python\Python39\python.exe "C:\Program Files\JetBrains\PyCharm Comm
Connected to pydev debugger (build 212.5080.64)


***** Math Operations In Python *****



1. Addition
Enter two integer numbers in order to perform addition operation :
First number :
>?
```



```
Debug - main
Console
C:\Users\Dell\AppData\Local\Programs\Python\Python39\python.exe "C:\Program Files\JetBrains\PyCharm Co
Connected to pydev debugger (build 212.5080.64)


***** Math Operations In Python *****



1. Addition
Enter two integer numbers in order to perform addition operation :
First number : Second number : The Addition of the entered two numbers i.e ( 5 + 3 ) is : 8


2. Subtraction
Enter two integer numbers in order to perform subtraction operation :
First number : Second number : The Subtraction of the entered two numbers i.e ( 8 - 2 ) is : 6


3. Multiplication
Enter two integer numbers in order to perform multiplication operation :
First number : Second number : The Multiplication of the entered two numbers i.e ( 2 * 6 ) is : 12


4. Division
Enter two integer numbers in order to perform division operation :
First number : Second number : The Division of the entered two numbers i.e ( 8 / 2 ) is : 4


5. Modulo Division
Enter two integer numbers in order to perform modulo-division operation :
First number : Second number : The Modulo-division of the entered two numbers i.e ( 9 % 2 ) is : 1
```


Debug - main
Console

```
2. Subtraction
Enter two integer numbers in order to perform subtraction operation :
First number : Second number : The Subtraction of the entered two numbers i.e ( 8 - 2 ) is : 6


3. Multiplication
Enter two integer numbers in order to perform multiplication operation :
First number : Second number : The Multiplication of the entered two numbers i.e ( 2 * 6 ) is : 12


4. Division
Enter two integer numbers in order to perform division operation :
First number : Second number : The Division of the entered two numbers i.e ( 8 / 2 ) is : 4


5. Modulo Division
Enter two integer numbers in order to perform modulo-division operation :
First number : Second number : The Modulo-division of the entered two numbers i.e ( 9 % 2 ) is : 1


6. Exponentiation
Enter two integer numbers in order to perform exponentiation operation :
First number as a base : Second number as a exponent or power : The exponentiation of the entered two


7.Floor Division
Enter two integer numbers in order to perform floor-division operation :
First number : Second number : The Floor-division of the entered two numbers i.e ( 9 // 5 ) is : 1

Process finished with exit code 0
```

2. String Operations in Python

Program:

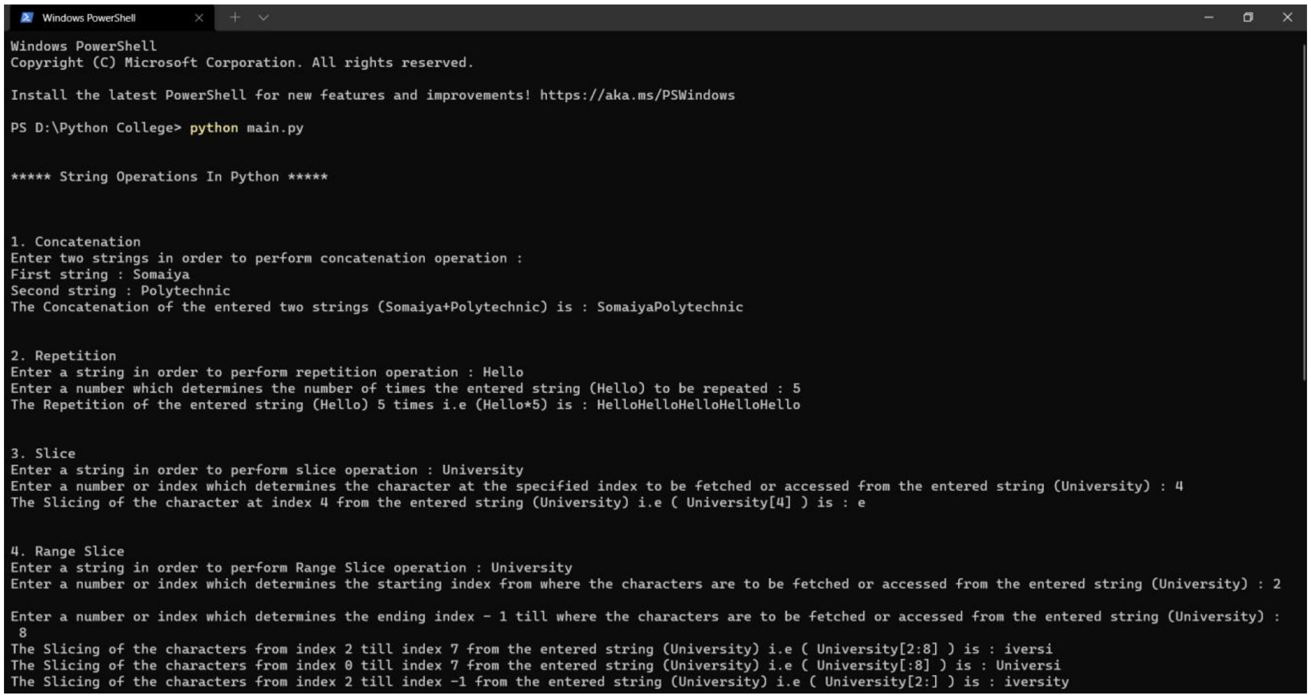
```
print("\n")
print("***** String Operations In Python *****")
print("\n\n")
print("1. Concatenation")
print("Enter two strings in order to perform concatenation operation :")
string_1=input("First string : ")
```

```
string_2=input("Second string : ")
concatenation=string_1+string_2
print("The Concatenation of the entered two strings (%s+%s) is : %s"%(string_1,string_2,concatenation))
print("\n")
print("2. Repetition")
string=input("Enter a string in order to perform repetition operation : ")
number=int(input("Enter a number which determines the number of times the entered string (%s) to be repeated : "%(string,)))
repetition=string*number
print("The Repetition of the entered string (%s) %d times i.e (%s*%d) is : %s"%(string,number,string,number,repetition))
print("\n")
print("3. Slice")
string=input("Enter a string in order to perform slice operation : ")
index=int(input("Enter a number or index which determines the character at the specified index to be fetched or accessed from the entered string (%s) : "%(string,)))
slicing=string[index]
print("The Slicing of the character at index %d from the entered string (%s) i.e ( %s[%d] ) is : %s"%(index,string,string,index,slicing))
print("\n")
print("4. Range Slice")
string=input("Enter a string in order to perform Range Slice operation : ")
start_index=int(input("Enter a number or index which determines the starting index from where the characters are to be fetched or accessed from the entered string (%s) : "%(string,)))
```

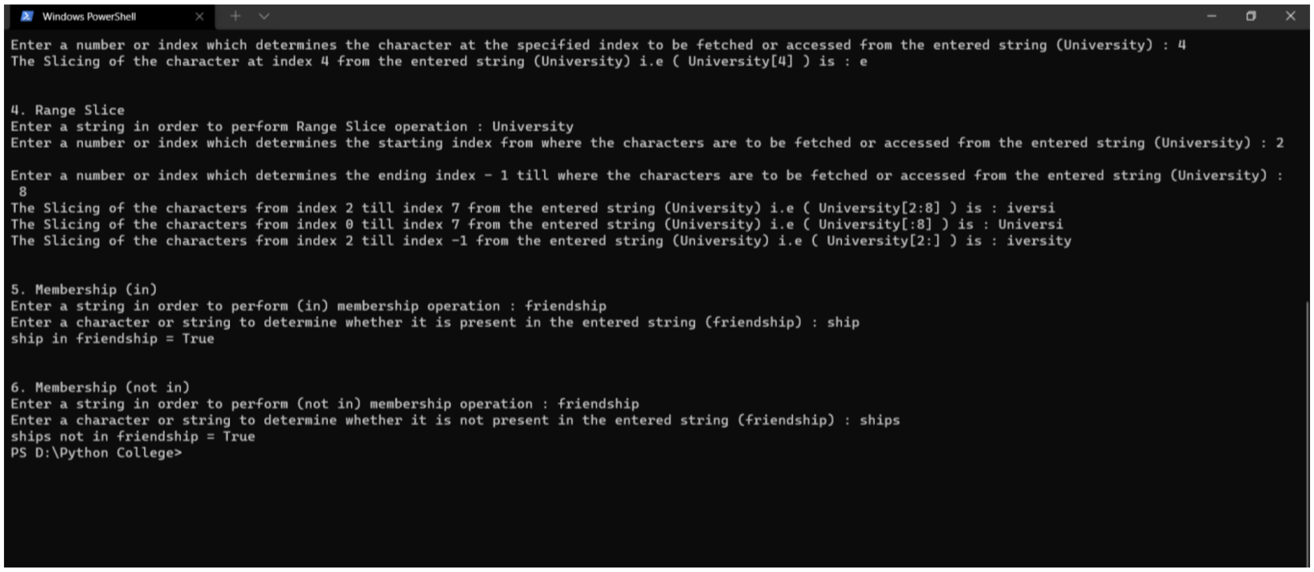
```
end_index=int(input("Enter a number or index which determines the ending index - 1 till where the characters are to be fetched or accessed from the entered string (%s) : "%(string,)))
range_slicing=string[start_index:end_index]
print("The Slicing of the characters from index %d till index %d from the entered string (%s) i.e ( %s[%d:%d] ) is : %s"%(start_index,end_index-1,string,string,start_index,end_index,range_slicing))
print("The Slicing of the characters from index 0 till index %d from the entered string (%s) i.e ( %s[:%d] ) is : %s"%(end_index-1,string,string,end_index,string[:end_index]))
print("The Slicing of the characters from index %d till index -1 from the entered string (%s) i.e ( %s[%d:] ) is : %s"%(start_index,string,string,start_index,string[start_index:]))
print("\n")
print("5. Membership (in)")
string_1=input("Enter a string in order to perform (in) membership operation : ")
string_2=input("Enter a character or string to determine whether it is present in the entered string (%s) : "%(string_1,))
in_membership=string_2 in string_1
print("%s in %s = %s"%(string_2,string_1,in_membership))
print("\n")
print("6. Membership (not in)")
string_1=input("Enter a string in order to perform (not in) membership operation : ")
string_2=input("Enter a character or string to determine whether it is not present in the entered string (%s) : "%(string_1,))
```

```
not_in_membership=string_2 not in string_1
print("%s not in %s = %s"%(string_2,string_1,not_in_membership))
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\Python College> python main.py


***** String Operations In Python *****



1. Concatenation
Enter two strings in order to perform concatenation operation :
First string : Somaiya
Second string : Polytechnic
The Concatenation of the entered two strings (Somaiya+Polytechnic) is : SomaiyaPolytechnic


2. Repetition
Enter a string in order to perform repetition operation : Hello
Enter a number which determines the number of times the entered string (Hello) to be repeated : 5
The Repetition of the entered string (Hello) 5 times i.e (Hello*5) is : HelloHelloHelloHelloHello


3. Slice
Enter a string in order to perform slice operation : University
Enter a number or index which determines the character at the specified index to be fetched or accessed from the entered string (University) : 4
The Slicing of the character at index 4 from the entered string (University) i.e ( University[4] ) is : e


4. Range Slice
Enter a string in order to perform Range Slice operation : University
Enter a number or index which determines the starting index from where the characters are to be fetched or accessed from the entered string (University) : 2

Enter a number or index which determines the ending index - 1 till where the characters are to be fetched or accessed from the entered string (University) :
 8
The Slicing of the characters from index 2 till index 7 from the entered string (University) i.e ( University[2:8] ) is : iversi
The Slicing of the characters from index 0 till index 7 from the entered string (University) i.e ( University[:8] ) is : Universi
The Slicing of the characters from index 2 till index -1 from the entered string (University) i.e ( University[2:] ) is : iversity
```



```
Enter a number or index which determines the character at the specified index to be fetched or accessed from the entered string (University) : 4
The Slicing of the character at index 4 from the entered string (University) i.e ( University[4] ) is : e


4. Range Slice
Enter a string in order to perform Range Slice operation : University
Enter a number or index which determines the starting index from where the characters are to be fetched or accessed from the entered string (University) : 2

Enter a number or index which determines the ending index - 1 till where the characters are to be fetched or accessed from the entered string (University) :
 8
The Slicing of the characters from index 2 till index 7 from the entered string (University) i.e ( University[2:8] ) is : iversi
The Slicing of the characters from index 0 till index 7 from the entered string (University) i.e ( University[:8] ) is : Universi
The Slicing of the characters from index 2 till index -1 from the entered string (University) i.e ( University[2:] ) is : iversity


5. Membership (in)
Enter a string in order to perform (in) membership operation : friendship
Enter a character or string to determine whether it is present in the entered string (friendship) : ship
ship in friendship = True


6. Membership (not in)
Enter a string in order to perform (not in) membership operation : friendship
Enter a character or string to determine whether it is not present in the entered string (friendship) : ships
ships not in friendship = True
PS D:\Python College>
```

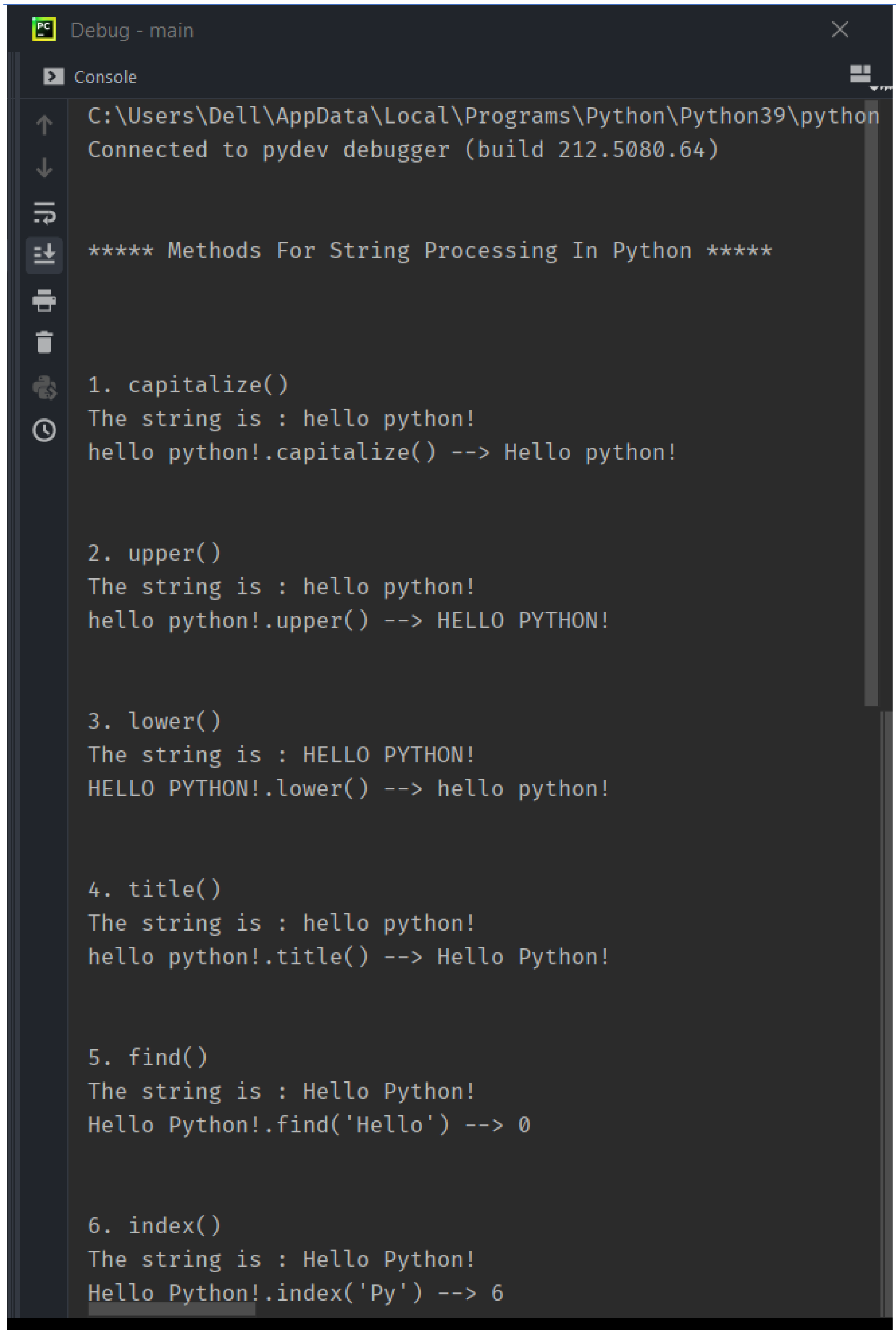
3. Methods for string processing in Python

Program:

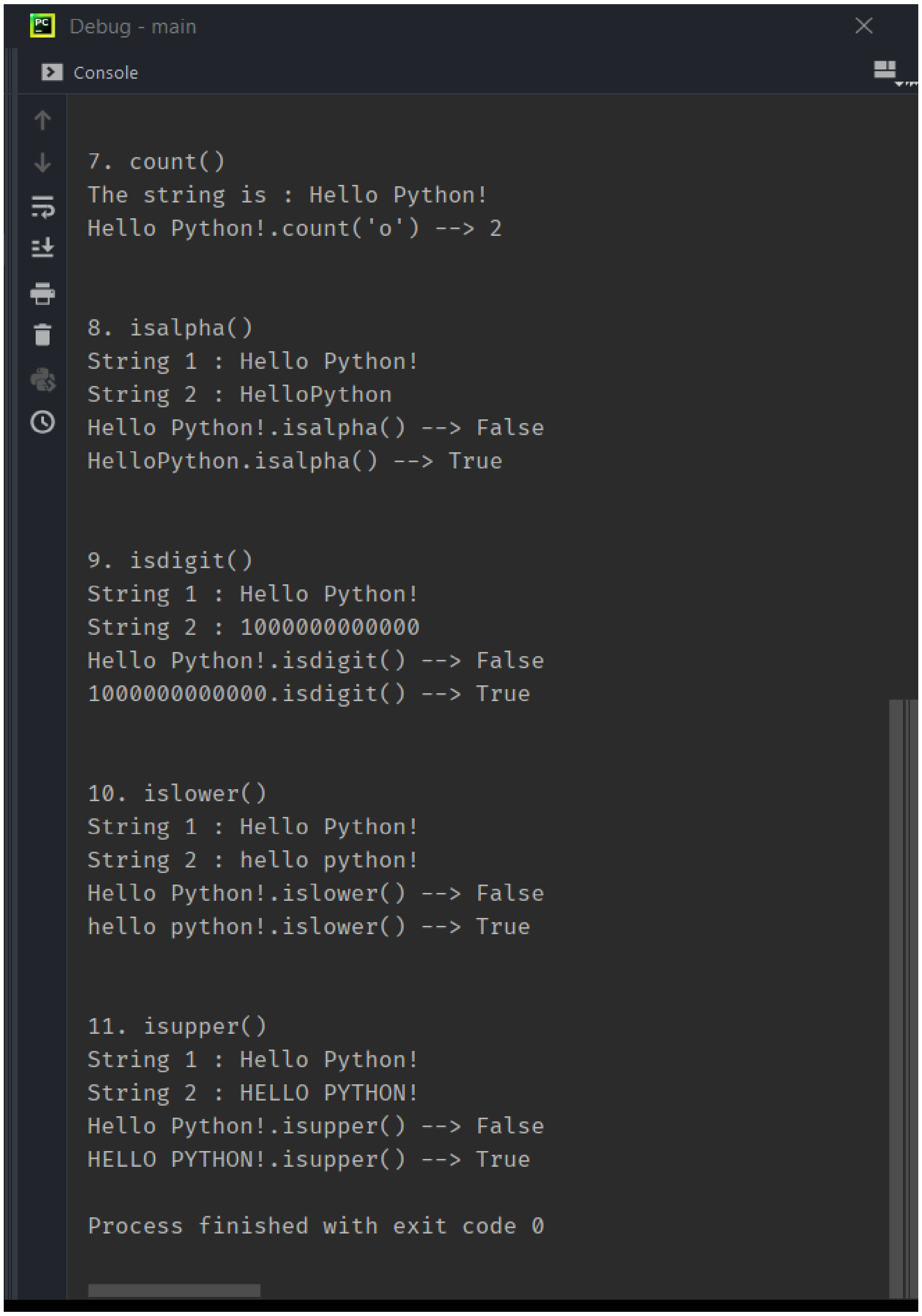
```
print("\n")
print("***** Methods For String Processing In Python *****")
print("\n\n")
print("1. capitalize()")
string="hello python!"
print("The string is : %s"%(string,))
print("%s.capitalize() --> %s"%(string,string.capitalize()))
print("\n")
print("2. upper()")
string="hello python!"
print("The string is : %s"%(string,))
print("%s.upper() --> %s"%(string,string.upper()))
print("\n")
print("3. lower()")
string="HELLO PYTHON!"
print("The string is : %s"%(string,))
print("%s.lower() --> %s"%(string,string.lower()))
print("\n")
print("4. title()")
string="hello python!"
print("The string is : %s"%(string,))
print("%s.title() --> %s"%(string,string.title()))
print("\n")
print("5. find()")
string="Hello Python!"
print("The string is : %s"%(string,))
print("%s.find('Hello') --> %s"%(string,string.find("Hello")))
print("\n")
print("6. index()")
string="Hello Python!"
print("The string is : %s"%(string,))
print("%s.index('Py') --> %s"%(string,string.index("Py")))
print("\n")
print("7. count()")
string="Hello Python!"
print("The string is : %s"%(string,))
print("%s.count('o') --> %s"%(string,string.count("o")))
print("\n")
print("8. isalpha()")
string_1="Hello Python!"
string_2="HelloPython"
print("String 1 : %s"%(string_1,))
print("String 2 : %s"%(string_2,))
print("%s.isalpha() --> %s"%(string_1,string_1.isalpha()))
print("%s.isalpha() --> %s"%(string_2,string_2.isalpha()))
print("\n")
print("9. isdigit()")
string_1="Hello Python!"
```

```
string_2="1000000000000"
print("String 1 : %s"%(string_1,))
print("String 2 : %s"%(string_2,))
print("%s.isdigit() --> %s"%(string_1,string_1.isdigit()))
print("%s.isdigit() --> %s"%(string_2,string_2.isdigit()))
print("\n")
print("10. islower()")
string_1="Hello Python!"
string_2="hello python!"
print("String 1 : %s"%(string_1,))
print("String 2 : %s"%(string_2,))
print("%s.islower() --> %s"%(string_1,string_1.islower()))
print("%s.islower() --> %s"%(string_2,string_2.islower()))
print("\n")
print("11. isupper()")
string_1="Hello Python!"
string_2="HELLO PYTHON!"
print("String 1 : %s"%(string_1,))
print("String 2 : %s"%(string_2,))
print("%s.isupper() --> %s"%(string_1,string_1.isupper()))
print("%s.isupper() --> %s"%(string_2,string_2.isupper()))
```

Output:



```
C:\Users\Dell\AppData\Local\Programs\Python\Python39\python
Connected to pydev debugger (build 212.5080.64)


***** Methods For String Processing In Python *****



1. capitalize()
The string is : hello python!
hello python!.capitalize() --> Hello python!


2. upper()
The string is : hello python!
hello python!.upper() --> HELLO PYTHON!


3. lower()
The string is : HELLO PYTHON!
HELLO PYTHON!.lower() --> hello python!


4. title()
The string is : hello python!
hello python!.title() --> Hello Python!


5. find()
The string is : Hello Python!
Hello Python!.find('Hello') --> 0


6. index()
The string is : Hello Python!
Hello Python!.index('Py') --> 6
```



```
7. count()
The string is : Hello Python!
Hello Python!.count('o') --> 2


8. isalpha()
String 1 : Hello Python!
String 2 : HelloPython
Hello Python!.isalpha() --> False
HelloPython.isalpha() --> True


9. isdigit()
String 1 : Hello Python!
String 2 : 1000000000000
Hello Python!.isdigit() --> False
1000000000000.isdigit() --> True


10. islower()
String 1 : Hello Python!
String 2 : hello python!
Hello Python!.islower() --> False
hello python!.islower() --> True


11. isupper()
String 1 : Hello Python!
String 2 : HELLO PYTHON!
Hello Python!.isupper() --> False
HELLO PYTHON!.isupper() --> True

Process finished with exit code 0
```

Conclusion: Thus, we have written python programs to understand Expressions, Variables, basic Math operations, Strings, Basic String Operation & String Methods.