PRACTICAL 1

Aim: To write, test and debug basic Python programs.

Theory: Python is a widely used general-purpose, high level programming language. Python was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

Python Variables: Variables are containers for storing data values.
Creating Variables: Python has no command for declaring a variable. A variable is created the moment you first assign a value to it.

```python
x = 5
y = "John"
print(x)
print(y)
```

Variables do not need to be declared with any particular type and can even change type after they have been set.

```
x = 4       # x is of type int
x = "Sally" # x is now of type str
print(x)
```

Casting: If you want to specify the data type of a variable, this can be done with casting.

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
```

Python Built-in Data Types:
Variables can store data of different types, and different types can do different things. Python has the following data types built-in by default, in these categories:

Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`
None Type:	`NoneType`

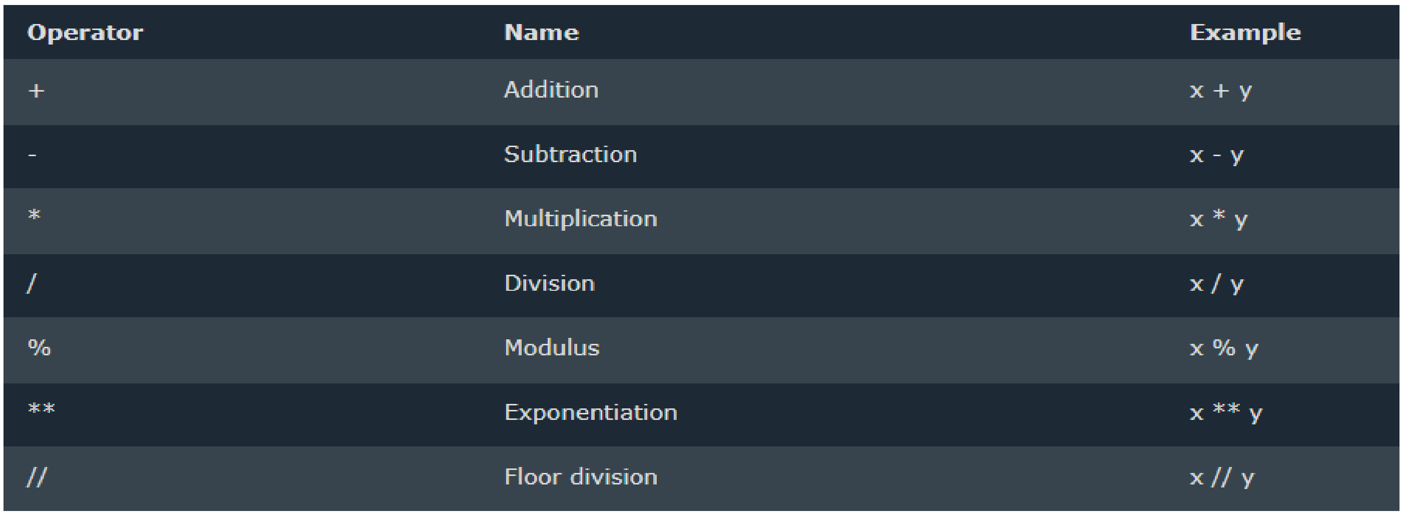
Setting the Specific Data Type:

Example	Data Type
x = str("Hello World")	str
x = int(20)	int
x = float(20.5)	float
x = complex(1j)	complex
x = list(("apple", "banana", "cherry"))	list
x = tuple(("apple", "banana", "cherry"))	tuple
x = range(6)	range
x = dict(name="John", age=36)	dict
x = set(("apple", "banana", "cherry"))	set
x = frozenset(("apple", "banana", "cherry"))	frozenset
x = bool(5)	bool
x = bytes(5)	bytes

Python Operators: Used to perform operations on variables values. Python divides the operators in the following groups:

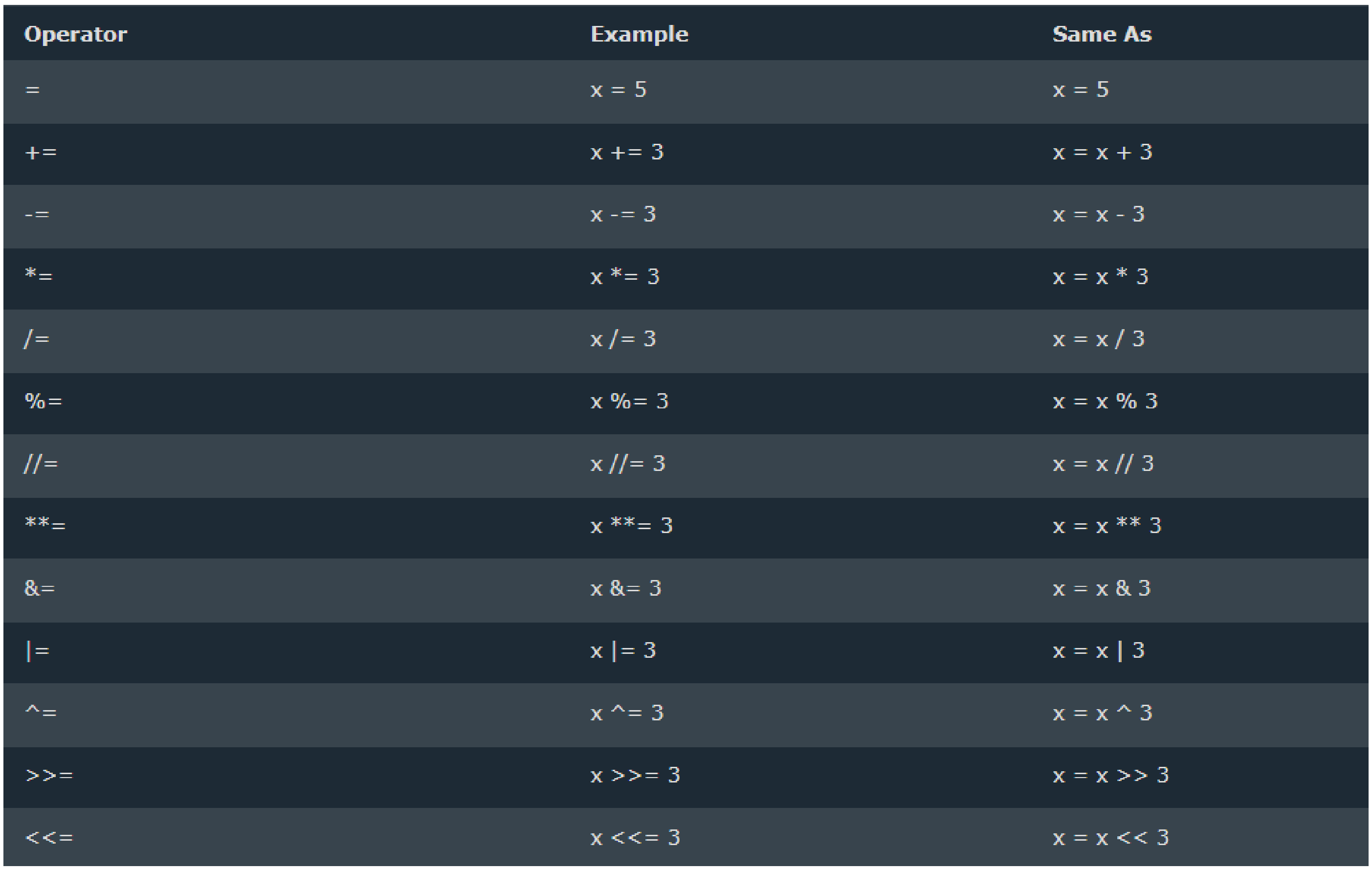
- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Arithmetic Operators: Used with numeric values to perform common mathematical operations:



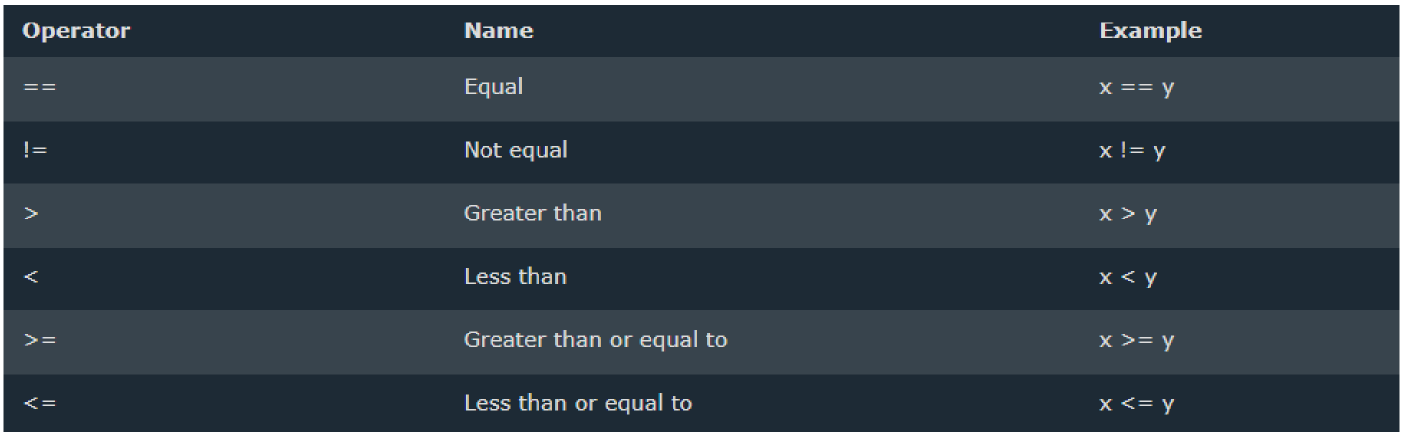
Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Assignment Operators: Used to assign values to variables:



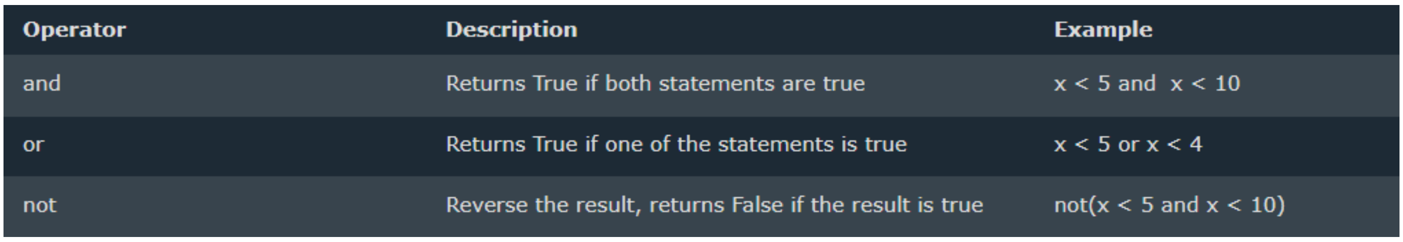
Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Comparison Operators: Used to compare two values.



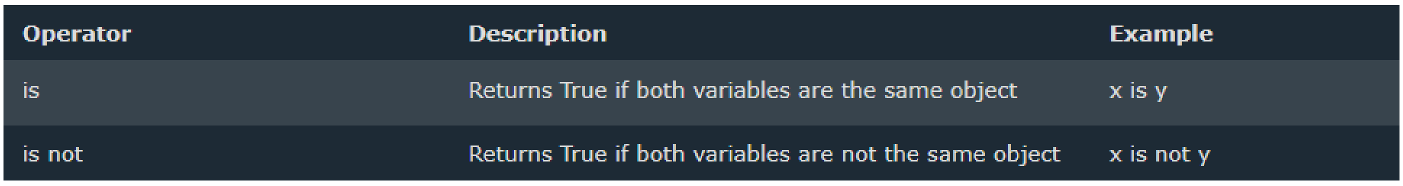
Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical operators: Used to combine conditional statements:



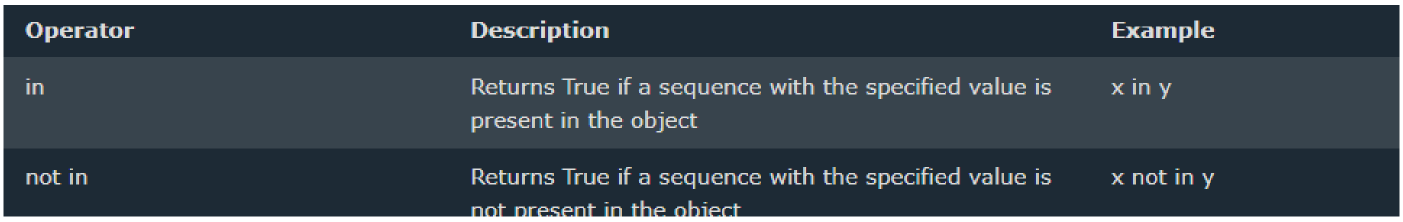
Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Identity operators: Used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location.



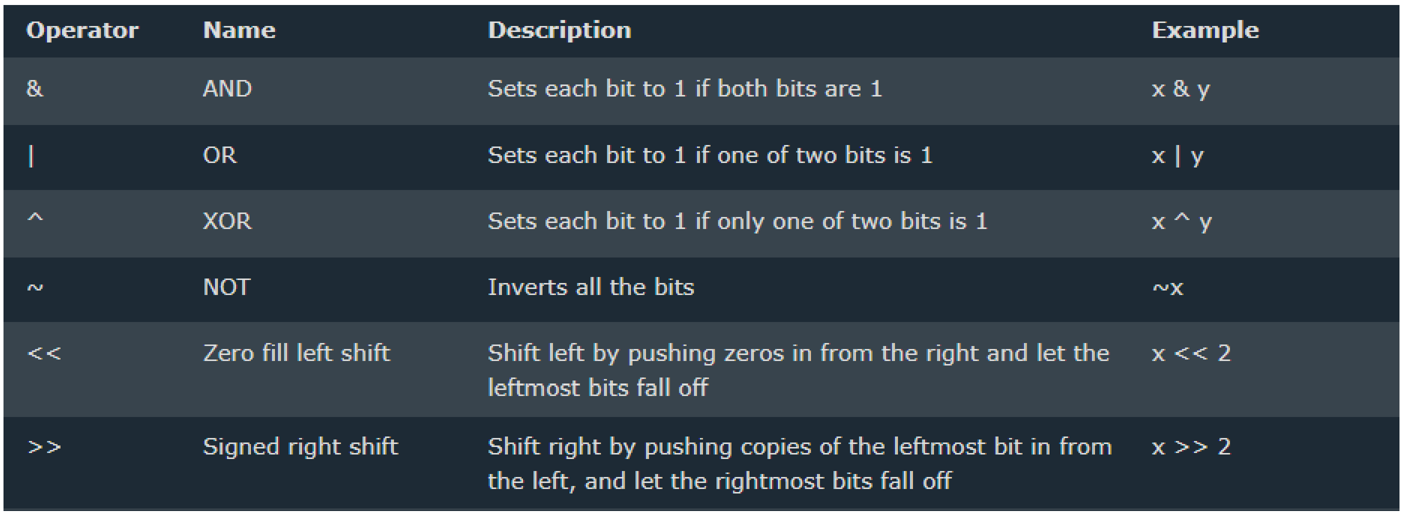
Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Membership operators: Used to test if a sequence is presented in an object.



Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Bitwise operators: Used to compare (binary) numbers:



Operator	Name	Description	Example
&	AND	Sets each bit to 1 if both bits are 1	x & y
|	OR	Sets each bit to 1 if one of two bits is 1	x | y
^	XOR	Sets each bit to 1 if only one of two bits is 1	x ^ y
~	NOT	Inverts all the bits	~x
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off	x << 2
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off	x >> 2

Program 1: Write a program to add three numbers.

Code:

```
first = int(input("Enter the First number: "));
second = int(input("Enter the Second number: "));
third = int(input("Enter the Third number: "));
sum = first + second + third;
print(str("The sum of these three numbers is ") + str(sum));
```

Output:

```
Enter the First number: 453
Enter the Second number: 212
Enter the Third number: 344
The sum of these three numbers is 1009
PS C:\Users\Aman Shrivastav\Desktop\Practicals\Python\Practical 1>
```

Program 2A: Write a program to swap two numbers using the third variable.

Code:

```
a = int(input("Enter the first number: "));
b = int(input("Enter the second number: "));
temp = a;
a = b;
b = temp;
print("After Swapping: "+ str(a),str(b))
```

Output:

```
Enter the first number: 45
Enter the second number: 77
After Swapping using the third variable: 77 45
PS C:\Users\Aman Shrivastav\Desktop\Practicals\Python\Practical 1>
```

Program 2B: Write a program to swap two numbers without using the third variable.

Code:

```
a = int(input("Enter the first number: "));
b = int(input("Enter the second number: "));
a = a + b
b = a - b
a = a - b
print("After Swapping: "+ str(a),str(b))
```

Output:

```
Enter the first number: 32
Enter the second number: 90
After Swapping without using the third variable: 90 32
PS C:\Users\Aman Shrivastav\Desktop\Practicals\Python\Practical 1>
```

Program 3: Write a program to calculate the Area of Triangle.

Code:

```
a = int(input("Enter the Height of the Triangle: "))
b = int(input("Enter the Base of the Triangle: "))
area = 1/2 * a * b;
print("The Area of the Triangle is: " + str(area));
```

Output:

```
Enter the Height of the Triangle: 12
Enter the Base of the Triangle: 20
The Area of the Triangle is: 120.0
PS C:\Users\Aman Shrivastav\Desktop\Practicals\Python\Practical 1>
```

Program 4: Write a program to solve a Quadratic Equation

Code:

```
import cmath
a = int(input("Enter 'a' for ax^2 + bx + c: "))
b = int(input("Enter 'b' for ax^2 + bx + c: "))
c = int(input("Enter 'c' for ax^2 + bx + c: "))
#  calculate the discriminant
discriminant = (b**2) - (4*a*c)
# find two solutions
sol1 = (-b-cmath.sqrt(discriminant))/(2*a)
sol2 = (-b+cmath.sqrt(discriminant))/(2*a)
print('The Solutions are {0} and {1}'.format(sol1,sol2))
```

Output:

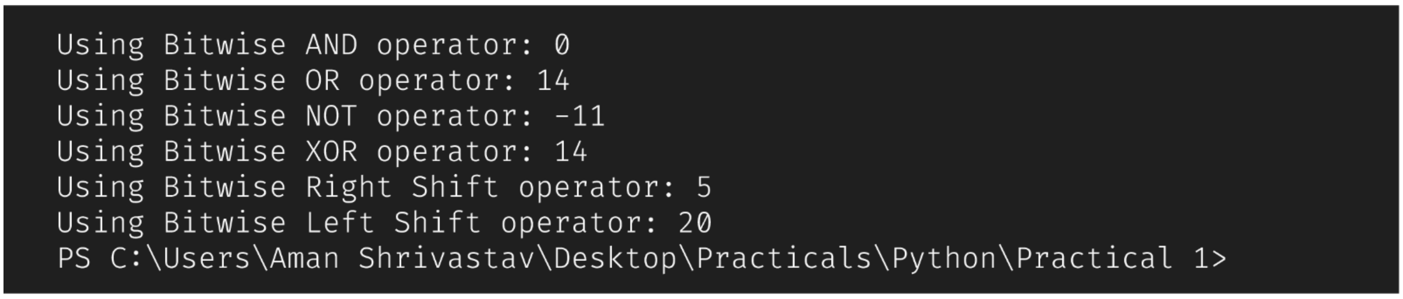
```
Enter 'a' for ax^2 + bx + c: 21
Enter 'b' for ax^2 + bx + c: 42
Enter 'c' for ax^2 + bx + c: 21
The Solutions are (-1+0j) and (-1+0j)
PS C:\Users\Aman Shrivastav\Desktop\Practicals\Python\Practical 1>
```

Program 5: Write a program displaying the use of Bitwise Operators:

Code:

```
a = 10  # 1010 (Binary)
b = 4  # 0100 (Binary)
print(str("Using Bitwise AND operator:"),str(a & b));
print(str("Using Bitwise OR operator:"),str(a | b));
print(str("Using Bitwise NOT operator:"),str(~a));
print(str("Using Bitwise XOR operator:"),str(a ^ b));
print(str("Using Bitwise Right Shift operator:"),str(a >> 1));
print(str("Using Bitwise Left Shift operator:"),str(a << 1));
```

Output:



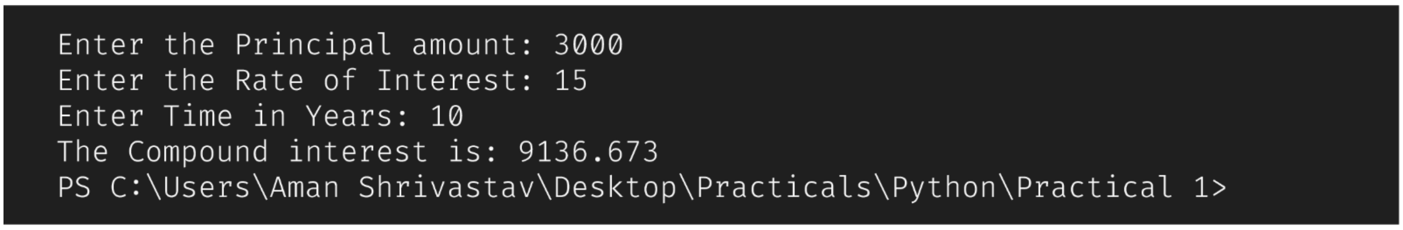
```
Using Bitwise AND operator: 0
Using Bitwise OR operator: 14
Using Bitwise NOT operator: -11
Using Bitwise XOR operator: 14
Using Bitwise Right Shift operator: 5
Using Bitwise Left Shift operator: 20
PS C:\Users\Aman Shrivastav\Desktop\Practicals\Python\Practical 1>
```

Program 6: Write a program to calculate the Compound Interest given all required values.

Code:

```
principal = int(input("Enter the Principal amount: "))
rate = int(input("Enter the Rate of Interest: "))
time = int(input("Enter Time in Years: "))
amount = principal * (pow((1 + rate / 100), time))
CI = amount - principal;
print("The Compound interest is:", round(CI,3))
```

Output:



```
Enter the Principal amount: 3000
Enter the Rate of Interest: 15
Enter Time in Years: 10
The Compound interest is: 9136.673
PS C:\Users\Aman Shrivastav\Desktop\Practicals\Python\Practical 1>
```

Program 7: Write a program to generate a random number between 0 and 100.

Code:

```
import random
print(str("The Random Number:"),random.randrange(0,100))
```

Output:

```
The Random Number: 20
PS C:\Users\Aman Shrivastav\Desktop\Practicals\Python\Practical 1>
```

Program 8: Write a program to display the calendar for January 2019.

Code:

```
import calendar
print(calendar.month(2019,1))
```

Output:

```
    January 2019
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 31

PS C:\Users\Aman Shrivastav\Desktop\Practicals\Python\Practical 1>
```

Program 9: Write a program to add two binary numbers

Code:

```
b1='100010' #Decimal value: 34
b2='101001' #Decimal value: 41

# Passing the base value of 2 for binary to the int() function
result = bin(int(b1,2) + int(b2,2))
# To get rid of the suffix '0b' at the start
print("The Addition of "+ b1 +" and " + b2 +" is:",result[2:])
```

Output:

```
The Addition of 100010 and 101001 is: 1001011
PS C:\Users\Aman Shrivastav\Desktop\Practicals\Python\Practical 1>
```

Conclusion: This program has equipped us with the foundational skills to create, evaluate, and debug Python code effectively. With these fundamental skills we are ready to explore more advanced programming challenges and opportunities.