## Q.1. Create two int type variables, apply addition, subtraction, division and multiplications and store the results in variables. Then print the data in the following format by calling the variables:

First variable is __ & second variable is __. Addition: __ + __ = __ Subtraction: __ - __ = __ Multiplication: __ * __ = __ Division: __ / __ = __

In [7]:
```
a = 7
b = 17
add = a + b
sub = b - a
mult = a * b
div = b / a

print(f"First variable is {a} & second variable is {b}")
print(f"Addition: {a} + {b} = {add}")
print(f"Subtraction: {b} - {a} = {sub}")
print(f"Multiplication: {a} * {b} = {mult}")
print(f"Division: {b} / {a} = {div}")
```

```
First variable is 7 & second variable is 17
Addition: 7 + 17 = 24
Subtraction: 17 - 7 = 10
Multiplication: 7 * 17 = 119
Division: 17 / 7 = 2.4285714285714284
```

## Q.2. What is the difference between the following operators:

(i) '/' & '//' (ii) '**' & '^'

(1) Both operators / and // are use for division depending on the desired results. When / used with two numbers, it performs regular division, returning a float as the result, even if the division is evenly divisible. Eg: 4/2=2.0

Operator // is known as floor division, which returns the largest integer that is less than or equal to the exact division result. Eg: 7/2=3

(2) The ** operator is used for exponentiation or raising a number to a power. It takes two operands: the base and the exponent. Eg: 2 ** 3 = 8 The ^ Operator in Python, is used for the bitwise XOR (exclusive OR) operation. It performs a bitwise comparison of corresponding bits in two integers and returns a new integer with the result. Eg: 5 ^ 3 = 6

## Q.3. List the logical operators.

The Logical Operators are given as follows:

'and': The "and" operator returns True if both operands are True, and False otherwise. Eg: True and False returns False.

'or': The "or" operator returns True if at least one of the operands is True, and False if both operands are False. Eg: True or False returns True.

'not': The "not" operator returns the negation of the operand. If the operand is True, it returns False, and if the operand is False, it returns True. Eg: not True returns False.

## Q.4. Explain right shift operator and left shift operator with examples.

The right shift (>>) and left shift (<<) operators are used for shifting the bits of a number to the right or left, respectively.

The right shift operator (>>) shifts the bits of a number to the right by a specified number of positions. Each bit of the number is shifted to the right by n positions, and the vacated positions are filled with zeros.

In [10]:
```
#Eg:
8>>2
```

Out[10]: 2

The left shift operator (<<) shifts the bits of a number to the left by a specified number of positions. Each bit of the number is shifted to the left by n positions, and the vacated positions on the right side are filled with zeros.

In [15]:
```
#Eg:
3<<2
```

Out[15]: 12

## Q.5. Create a list containing int type data of length 15. Then write a code to check if 10 is present in the list or not.

In [18]:
```
lst = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]

# code to find whether list contains 10 or not

if 10 in lst:
    print("10 is present in the list.")
else:
    print("10 is not present in the list.")
```

```
10 is present in the list.
```