In [2]:

```
#Unary operations: it is done with one variable

#Unary +(plus)
#produces the positive value of the input
#(equivalent to the multiplication with +1)
num_x = 5
print(num_x)
print(+num_x)

#Unary -(minus):
#produces the negative value of the input
#(equivalent to the multiplication with -1)
num_x = 5
print(num_x)
print(-num_x)

#Unary ~(invert):
#We use unary ~ (invert) operation by
#adding a '~' before a variable or data.
#It produces a bitwise inverse of a given data.
#Simply, for any data x,
#a bitwise inverse is defined in python as -(x+1).
num_x = 5
print(num_x)
print(~num_x) #-((num_x)+1) = -(5+1) = -6
```

```
5
5
5
-5
5
-6
```

In [3]:

```
#(Works with int, and boolean.
#For booleans, True and False will be valued as 1 and 0 respectively.)

#True = 1 False = 0

#Unary addition
print(+True) # (1*(+1))
print(+False) # (0*(+1))

#Unary subtraction
print(-True) # (1*(-1))
print(-False) # (0*(-1))

#Unary invert
print(~True) # -(x+1) = -(1+1) = -2
print(~False) # -(x+1) = -(0+1) = -1

```

```
1
0
-1
0
-2
-1
```

In [4]:

```
# Arithmetic Operation

#Addition
#ADDS VALUES ON EITHER SIDE OF THE OPERATOR
num1 = 5
num2 = 2
print(num1+num2)

```

```
7
```

In [5]:

```
# Arithmetic Operation

#Subtraction
#SUBTRACTS RIGHT HAND OPERATION FROM LEFT HAND OPERATION
num1 = 5
num2 = 2
print(num1-num2)

```

```
3
```

In [6]:

```
# Arithmetic Operation

#Multiplication
#MULTIPLIES VALUES ON EITHER SIDE OF THE OPERATOR
num1 = 5
num2 = 2
print(num1*num2)

```

```
10
```

In [7]:

```
# Arithmetic Operation

#Division
#DIVIDE LEFT HAND OPERAND BY RIGHT HAND OPERAND
num1 = 10
num2 = 5
print(num1/num2) #Ans in float

```

```
2.0
```

In [22]:

```
# Arithmetic Operation

#Modulus = REMAINDER OPERATOR

#Example 1
num1 = 10
num2 = 5
print("Example 1: ", num1%num2) #remainder of 10/5

#Example 2
num3 = 3
num4 = 4
print("Example 2: ", num3%num4) #remainder of 3/4

# Mod Operator: [LEFT TO COVER]

#Example 1
num1 = 10
num2 = 5
print("Example 1:", num1%num2) #remainder of 10/5

#Example 2
num3 = 3
num4 = 4
print("Example 2:", num3%num4) #remainder of 3/4

#For negative values
# z = x - y*(x//y)
# x%y = (a+b)mod y = [(a mod y)+(b mod y)]mod y

num3 = -3
num4 = -8
print("Example 3: ", num4%num3)

# -3)-8(2
#    -6
#    -2

num3 = -3
num4 = 8
# z = x - y*(x//y) #x=8 y=-3
# x%y = (a+b)mod y = [(a mod y)+(b mod y)]mod y
#ans aim: denominator er sign or 0
print("Example 4: ", num4%num3) #8%-3 #denominator=-3 #numerator=8
#numerator denominator
#8-3 = 5-3 = 2-3 = -1


num3 = 3
num4 = -8
# z = x - y*(x//y) #x=-8 y=3
# x%y = (a+b)mod y = [(a mod y)+(b mod y)]mod y
#ans aim: denominator er sign or 0
print("Example 5: ", num4%num3) #8%-3 #denominator=+3 #numerator=-8
#numerator denominator
#-8+3 = -5+3 = -2+3 = 1


```

```
Example 1:  0
Example 1:  0
Example 2:  3
Example 1: 0
Example 2: 3
Example 3:  -2
Example 4:  -1
Example 5:  1
```

In [9]:

```
# Arithmetic Operation

#Exponentiation = Replacement of power operator
num = 3
print(num**3) #3 to the power 3
```

```
27
```

In [10]:

```
# Arithmetic Operation

#Floor Division
#Returns floor value for both integer and floating point arguments.

#5/2 = 2.5 Floor = 2
print("Floor Division", 5//2) #ans in int as both sides have int
#Floor division = always gives min number
print("Floor Division", 5.0//2)
print("Floor Division", 5//2.0)
print("Floor Division", 5.0//2.0)
print("Floor Division", -5//2)
print("Floor Division", -5.0//2)

#Comparing to Normal Division; ans in float no matter what
print("Normal Division", -5/2)
```

```
Floor Division 2
Floor Division 2.0
Floor Division 2.0
Floor Division 2.0
Floor Division -3
Floor Division -3.0
Normal Division -2.5
```

In [11]:

```
#Python Assignment Operators

x=5 # = assigns value on RHS to LHS
print(x)

x+=3
#means x=x+3; incrementing x by 3; adds value on RHS to value on LHS
print(x)

x-=3
#means x=x-3; decrementing x by 3; subtracts RHS from LHS
print(x)

x*=3
#means x=x*3; multiplying x's value by 3
print(x)

x/=3
#means x=x/3; dividing x's value by 3
print(x)

x%=3
#means x=x%3
print(x)

x**=3
#means x=x**3
print(x)

#x = 8

x//=3
#means x=x//3 so, 8.0//3
print(x)
```

```
5
8
5
15
5.0
2.0
8.0
2.0
```

In [14]:

```python
#Python Membership Operators

#Two types: in and not in

#in:
#Returns True
#if a sequence with the specified value is present in the object

#not in:
#Returns True
#if a sequence with the specified value is not present in the object

#in
x = ["apple", "banana"]
print("mango" in x)

y = "Hello world"
print("H" in y)
print("S" in y)

#not in
m = "Hello world"
print("H" not in m)

mn = "Hello world"
print("h" in mn)

n = ["apple", "banana"]
print("Mango" not in n)
```

```
False
True
False
False
False
True
```

In [ ]:

```
#Logical Operator

#And
#Or
#not

and (logical AND): The logical and returns True if both values are True.
                   Otherwise, returns False.

or (logical OR): The logical or returns False if both values are False.
                 Otherwise, returns True.

not (Logical NOT): Returns True, if False is given and vise versa.

And Operator:
False and True = False
True and False = False
False and False = False
True and True = True

Or Operator:
False or True = True
True or False = True
False or False = False
True or True = True

not Operator: [LEFT TO COVER]
if A = True
not A = not True = False
if A = False
not A = not False = True


#Comparison or Relational Operator
#== (equal)
#!= (not equal)
#> (greater than)
#< (less than)
#>= (greater than or equal)
#<= (less than or equal)
```

In [13]:

```
#Operator Precedence

print(True or (False and True))

#Note:
#False and True = False
#False and False = False
#True and True = True
#True and False = False

#True or False = True
#False or True = True
#False or False = False
#True or True = True

#True or (False)
#True or False
#True
```

```
True
```

In [12]:

```
print( 9 - 5 // 2 * 13 + 2 ** 2)

# here, Exponentiation (**) has the highest precedence.
# So, 2**2 will be executed first
# print(9 - 5 // 2 * 13 + 2 ** 2)
# print(9 - 5 // 2 * 13 + 4)

# in the next step, floor division(//),
# and multiplication(*) has the same precedence
# if the precedence is the same then,
# we need to execute from left to right. So, 5 // 2

# print(9 - 5 // 2 * 13 + 4) print(9 - 2 * 13 + 4)

# then, * has the highest precedence.
# So 2 * 13 executes
# print(9 - 2 * 13 + 4) print(9 - 26 + 4)

# Here,- and + has the same precedence,
# so again left to right.
# print(9 - 26 + 4) print(-17 + 4) print(-13)
# Output: -13
```

```
-13
```

In [ ]:

```
#Bitwise & operator = &
#Bitwise or operator = |
#Bitwise nor operator = ~
#Bitwise xor operator = ^
#Bitwise left shift = <<
#Bitwise right shift = >>

#True = 1 False = 0
And Operator:
False and True = False
True and False = False
False and False = False
True and True = True

0&1=0
1&0=0
0&0=0
1&1=1

#True = 1 False = 0
Or Operator:
False or True = True
True or False = True
False or False = False
True or True = True

0|1=1
1|0=1
0|0=0
1|1=1


not Operator: [LEFT TO COVER]
if A = True
not A = not True = False
if A = False
not A = not False = True

~1=0
~0=1


Xor Operator:
False xor True = True
True xor False = True
False xor False = False
True xor True = False

0^1=1
1^0=1
0^0=0
1^1=0

Bitwise AND operator: Returns 1 if both the bits are 1 else 0.
example:
a = 10 = 1010 (Binary)
b = 4 =  0100 (Binary)

a & b = 1010
```

```
         &
        0100
      = 0000
      = 0 (Decimal)

Bitwise or operator: Returns 1 if either of the bit is 1 else 0.
example:
a = 10 = 1010 (Binary)
b = 4 =  0100 (Binary)

a | b = 1010
         |
        0100
      = 1110
      = 14 (Decimal)


Bitwise not operator: Returns one’s complement of the number.
example:
a = 10 = 1010 (Binary)

~a = ~1010
   = -(1010 + 1)
   = -(1011)
   = -11 (Decimal)

Bitwise xor operator:
Returns 1 if one of the bits is 1 and the other is 0 else returns false.
1 and 1 thakle 0 dibe
0 and 0 thakle 0 dibe
example:
a = 10 = 1010 (Binary)
b = 4 =  0100 (Binary)

a & b = 1010
         ^
        0100
      = 1110
      = 14 (Decimal)

```

In [23]:

```
a = 10 #1010 (Binary)
b = 4 #0100 (Binary)
print(a&b)

# a & b = 1010
#           &
#          0100
#        = 0000
#        = 0 (Decimal)
```

```
0
```

In [15]:

```
#Input function
#Previously, we used to store values in variables

store = "CS"

#Now, we want to take input from user

store = input ("Enter name: ") #Mention what to input. Good practice.
print("Name of the user is: ", store)
```

```
Enter name: xyz
Name of the user is:  xyz
```

In [16]:

```
#Limitation of taking input:
#All the input user enters gets converted into String.
#Even if user enters an integer, it is considered to be a string.

store = input ("Enter a number: ")
print(store)
print(type(store))
```

```
Enter a number: 1
1
<class 'str'>
```

In [17]:

```
#So, what can we do about this?
#Casting: Conversion of datatype

store = int(input ("Enter a number: "))
#The input given will be converted to an integer
print(store)
print(type(store))
```

```
Enter a number: 4
4
<class 'int'>
```

In [18]:

```
#More examples of datatype conversion:

x = 2.6
print(x)
x = int(x)
print(x)
y = int(x)
print(y)
```

```
2.6
2
2
```

In [2]:

```
# int(): constructs an integer number from various data types
# such as strings
# (the input string has to consist of numbers without any decimal points,
# basically whole numbers),
# and float-point numbers
# (by rounding up to a whole number,
# basically it truncates the decimal part).
# For example:

print(int("12"))
print(type(int("12")))

print(int(12.34))
print(type(int(12.34)))

# print(int("12.34")) => will give error
# print(int("12b")) => will give error
```

```
12
<class 'int'>
12
<class 'int'>


-----------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
 last)
/var/folders/b8/q3zmxcts1wv9d7xr2_lrs4hr0000gn/T/ipykernel_780/1429950
636.py in <module>
     15
     16 # print(int("12.34"))
---> 17 print(int("12b"))

ValueError: invalid literal for int() with base 10: '12b'
```

In [20]:

```
# float():
# constructs a floating-point number
# from various data types such as integer numbers, and strings
# (the input string has to be a whole number or a floating point number).
# For example:

print(float(12))
print(type(float(12)))

print(float("34"))
print(type(float("34")))

print(float("34.1234"))
print(type(float("34.1234")))

# print(float("34.5b12"))
# print(float("345b12"))
```

```
12.0
<class 'float'>
34.0
<class 'float'>
34.1234
<class 'float'>
```

In [21]:

```
#Deleting a variable
text = "My name is XYZ"
print(text)
del text
print(text)
```

```
My name is XYZ

-----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
 last)
/var/folders/b8/q3zmxcts1wv9d7xr2_lrs4hr0000gn/T/ipykernel_6247/193857
0882.py in <module>
      3 print(text)
      4 del text
----> 5 print(text)

NameError: name 'text' is not defined
```

In [ ]:

```

```