Based on the position of the operator expression can be catagorised into 3 types

1. Infix
2. Prefix
3. Postfix

Based on the datatype of the result obtained after evaluation, an expression can be categorized into following types -

1. Constant Expression : 8 + 9 - 2
2. Integral Expression (which involves integer arithmetic) Ex - a = 10

```
b = 5
c = a + b
```

3. Floating Point Expression: Ex- a * b / 3
4. Relational Expression: Ex- c = a > b
5. Logical Expression: Ex- a>b && b!=0
6. Bitwise Expression: Ex- c = a & b
7. Assignment Expression: Ex- a = b + c

Operation on String

String Concatenation

The process of combining 2 String is called Concatenation.

In [2]:
```
print("Hello"+'How are you')
```

```
HelloHow are you
```

In [4]:
```
print("Hello"+'5.1')
```

```
Hello5.1
```

In [5]:
```
print("Hello "+ str(5.1))
```

```
Hello 5.1
```

Here we have converted the float variable into String using Type Casting.

In [7]:
```
print(5*'Hello ')
```

```
Hello Hello Hello Hello Hello
```

Though String cannot be added with the number but it can be multiply with an integer and it results in repeating the same String.

In [8]:
```
print("Hello " * 5)
```

```
Hello Hello Hello Hello Hello
```

In [9]:
```
print("Hello" * 5.0)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_1976\192843477.py in <module>
----> 1 print("Hello" * 5.0)

TypeError: can't multiply sequence by non-int of type 'float'
```

String Slicing

In [10]:
```
str = 'Python is Easy to Learn!'
print(str[0])
```

```
P
```

In [11]:
```
print(str)
```

```
Python is Easy to Learn!
```

In [12]:
```
print(str[3:9])
```

```
hon is
```

In [16]:
```
print(str)
print(str[3:8])
print(str[3:11])
```

```
Python is Easy to Learn!
hon i
hon is E
```

In [17]:
```
print(str[4:])
```

```
on is Easy to Learn!
```

In [18]:
```
print(str[-1])
```

```
!
```

print(str[-1]) it will give the last element of the String

In [20]:
```
print(str)
print(str[-2])
```

```
Python is Easy to Learn!
n
```

In [26]:
```
str = "Welcome"
print(str) # Welcome
print(str[2]) # l
print(str[3:]) # come
print(str[3:5]) # co
print(str[-3]) # o
print(str[2:-3]) # lc
print(str[5]) # m
print(str*2) # WelcomeWelcome
```

```
Welcome
l
come
co
o
lc
m
WelcomeWelcome
```

Tuples

A Tuple is similar to List it consist of a number of values separated by comma ',' and enclosed by parenthesis.

- The main difference between a Tuple and List is that
- We can change the value in the list but not in a tuple.
- List is mutable(Ready to change).
- Tuple is immutable (Cannot be changed).

In [31]:
```
Tup = ('a','bc',70,1.23)
print(type(Tup))
```

```
<class 'tuple'>
```

In [30]:
```
Tup = ('a', 'bc', 70, 1.23)
```

Brackets used by the followings

- Tuple = ()
- List = []
- Dictionary = {}

In [36]:
```
print(Tup)
print(Tup[0])
print(Tup[3])
print(Tup[1:3]) # 1:n-1
print(Tup[1:4])
print(Tup[2:])
```

```
('a', 'bc', 70, 1.23)
a
1.23
('bc', 70)
('bc', 70, 1.23)
(70, 1.23)
```

In [39]:
```
Tup1 = (1,'234')
print(Tup)
print(Tup1)
print(Tup+Tup1)
```

```
('a', 'bc', 70, 1.23)
(1, '234')
('a', 'bc', 70, 1.23, 1, '234')
```

This will concatenate 2 Tuples Tup[] and Tup1[]

In [40]:
```
print(Tup * 2)
```

```
('a', 'bc', 70, 1.23, 'a', 'bc', 70, 1.23)
```

Create a List & do the following operations on the List.

1. List1 = [1,'cd',78,1.23]
2. List2 = ['d',78]

Qa. Print first element of the List1. Qb. Print elements starting from 2 to 3rd of List1. Qc. Print elements starting from 2nd to last of List1. Qd. Print concatenate 2 List. Qe. Print the last element of the List.

In [42]:
```
List1 = [1,'cd',78,1.23]
List2 = ['d',78]
print(List1)
#print(List2)
print(List1[0])
print(List1[2:4])
print(List1[2:])
print(List1+List2)
print(List1[-1])
```

```
[1, 'cd', 78, 1.23]
1
[78, 1.23]
[78, 1.23]
[1, 'cd', 78, 1.23, 'd', 78]
1.23
```

Dictionary

Dictionary stores data in **Key:Value pair**

- **The Key value are usually Strings and values can be of any datatype.**
- **The key value pairs are enclosed within braces{}**

- **Each Key value pairs are seperated by a colon ':'**
- **List and Dictionary are both mutable datatype, there value can be changed.**

In [43]:
```
#         Key : value   ,   Key  : value, Key:value
Dict = {"item":"Chocolate","price":100, "amount":25}
print(type(Dict))
```

```
<class 'dict'>
```

In [44]:
```
print(Dict)
```

```
{'item': 'Chocolate', 'price': 100, 'amount': 25}
```

In [48]:
```
print(Dict["item"])
print(Dict["Chocolate"]) # Value cannot be used as Key
```

```
Chocolate
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_1976\634470880.py in <module>
      1 print(Dict["item"])
----> 2 print(Dict["Chocolate"])

KeyError: 'Chocolate'
```

In [49]:
```
a = input("Enter a value: ")
b = input("Enter another value: ")
c = a + b
print(c)
```

```
Enter a value: 56
Enter another value: 64
5664
```

In [50]:
```
a = int(input("Enter a value: "))
b = int(input("Enter another value: "))
c = a + b
print(c)
```

```
Enter a value: 66
Enter another value: 64
130
```

Whenever we recieve input through keyboard it is by default treated as String and for proper execution we have to typecast in corresponding datatype as shown above.

Function of type conversion operation

1. **int(x)**
2. **float(x)**
3. **long(x)**
4. **str(x) - It converts x to a String**
5. **tuple(x) - It converts x to a Tuple**
6. **set(x) -**
7. **ord(x) - It converts a single character to its integer value.**

8. **oct(x) - It converts x to Octal**
9. **hex(x) - It converts x to Hexadecimal**
10. **char(x) - It converts x to Character**
11. **dict(x) - It converts x to dictionary**

In [ ]: