In [ ]:
```
conditional statement
```

In [ ]:
```
if condition:
    expresion
    body
elif condition:
    body
else:
    body
```

In [ ]:
```
if condition:
    body:
    if condition:
        body
    else:
        body
```

In [7]:
```
# int--integer
# str--string
# boolean---True or False

# input()#get value from the user
# type()------
# b=input("Enter Number 1 : ")
a=45.9#---float
print(type(a))
print(b)
```

```
<class 'float'>
45
```

In [12]:
```
a=int(input("Enter a number"))
print(a)
print(type(a))
```

```
Enter a number45
45
<class 'int'>
```

In [10]:
```
# casting----changing one datatype to another
a='10'
# int--str---float
# datatype(value_to_be_changed)
b=int(a)
print(type(a))
print(type(b))
```

```
<class 'str'>
<class 'int'>
```

In [15]:
```
#program to check whther the number is positive negative or zero
a=int(input("Enter a number: "))
if a>0:
    print("Number is positive")
elif a==0:
    print("Number is zero")
else:
    print("Number is negative")
```

```
Enter a number: 0
Number is zero
```

In [16]:
```
a=int(input("Enter a number: "))
if a>=0:
    if a==0:
        print("Number is zero")
    elif a>0:
        print("Number is positive")
elif a<0:
    print("Number is negative")
#nested
#ladder if else
```

```
Enter a number: 45
Number is positive
```

In [ ]:
```
print("Hello")
print("Hello")
```

In [ ]:
```
# loops is used to perform any repetitive task multiple number of times
# two types of loops in python
# 1)for loop
# 2)while loop
```

In [ ]:
```
for iterating_var in sequence_collections:
    body
```

In [ ]:
```
"hello"----collection of characters
[1,2,3,4,5,6]-----collections of integer
(1,4,"datatypes")---collections tuple
set{45,28}
dictionary
range(start,stop,step)
range(1,10,+2)-----
we we not mention or give starting point then 0
we not mention step then then 1 step
range(-1,-15,-1)
-9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 4 8
+---left to right
-____right to left
```

In [ ]:
```
Sequence
input
type(9)
casting
conditional opeartor
```

In [19]:
```
type("alka")
```

Out[19]:
```
str
```