The flow control statements are divided into three categories :

1. Conditional statements (if, if-else, if-elif-else)
2. Transfer statements (for, while)
3. Iterative statements (break, continue, pass)

##Condition Statement

condition statements act depending on whether a given condition is true or false.

In [6]:

```
number = 6
if number > 5:
  # Calculate square
  print(number * number)
```

```
36
```

In [7]:

```
a = 100
b = 10
if b>a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

```
b is not greater than a
```

In [9]:

```
a = 100
b = 10
if b>a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

```
a is greater than b
```

In [10]:

```
#Nested-if
a = int(input('Enter first number '))
b = int(input('Enter second number '))

if a > b:
    if a == b:
        print(a, 'and', b, 'are equal')
    else:
        print(a, 'is greater than', b)
else:
    print(a, 'is smaller than', b)
```

```
Enter first number 10
Enter second number 20
10 is smaller than 20
```

In [19]:

```
x = 41
if x < 10:
  print("Above Ten")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20!")
```

In [20]:

```
#Instead of writing a block after the colon, we can write a statement immediately after the
number = 56
if number > 0: print("positive")
else: print("negative")
```

```
positive
```

##Transfer Statement

For Loop : Using for loop, we can iterate any sequence or iterable variable. The sequence can be string, list, dictionary, set, or tuple.

In [21]:

```
for i in range(1, 11):
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

In [23]:

```
for i in range(1, 11):
    print(i, end=" ")
```

```
1 2 3 4 5 6 7 8 9 10
```

In [24]:

```
for x in range(3,30,3):
    print(x)
```

```
3
6
9
12
15
18
21
24
27
```

In [25]:

```
fruits = ["apple", "banana","cherry"]
for i in fruits:
  print(i)
```

```
apple
banana
cherry
```

In [27]:

```
#Nested for Loop
adj = ["tasty"]
fruits = ["apple", "banana","cherry"]
for x in adj:
  for y in fruits:
    print(x,y)
```

```
tasty apple
tasty banana
tasty cherry
```

While loop : The while loop statement repeatedly executes a code block while a particular condition is true./ as long as a condition is true.

In [30]:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

```
1
2
3
4
5
```

In [31]:

```
#Example to calculate the sum of first ten numbers

num = 10
sum = 0
i = 1
while i <= num:
    sum = sum + i
    i = i + 1
print("Sum of first 10 number is:", sum)
```

```
Sum of first 10 number is: 55
```

##Iterative Statement

Break Statement : Python break statement brings control out of the loop. If it matches the condition then it will end the loop instantly.

###For loop with break statement

In [33]:

```
List = [1,2,3,4]
for x in List:
  if x == 2:
    break
  print(x)
```

```
1
```

In [35]:

```
for num in range(10):
    if num > 5:
        break
    print(num)
```

```
0
1
2
3
4
5
```

###For loop with continue statement

Continue Statement : Python continue statement returns the control to the beginning of the loop. If it's matches the condition, then skip that value and continue the loop.

In [38]:

```
for letter in "NetTechIndia":
  if letter == 'e':
    continue
  print("Current Letter", letter)
```

```
Current Letter N
Current Letter t
Current Letter T
Current Letter c
Current Letter h
Current Letter I
Current Letter n
Current Letter d
Current Letter i
Current Letter a
```

In [39]:

```
for letter in "NetTechIndia":
  if letter == 'e':
    continue
  print(letter,end="")
```

```
NtTchIndia
```

In [40]:

```
for num in range(3, 8):
    if num == 5:
        continue
    else:
        print(num)
```

```
3
4
6
7
```

###For loop with pass statement

Pass Statement : In Python, pas is a null statement. The interpreter does not ignore a pass statement, but nothing happened and statement results into no operation.

In [41]:

```
months = ['January', 'June', 'March', 'April']
for mon in months:
    pass
print(months)
```

```
['January', 'June', 'March', 'April']
```

In [42]:

```
a = 3
b = 200
if b>a:
```

```
  File "<ipython-input-42-be7c4f690158>", line 3
    if b>a:
           ^
SyntaxError: unexpected EOF while parsing
```

In [43]:

```
a = 3
b = 200
if b>a:
  pass
```

###While loop with break statement

In [46]:

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

```
1
2
3
```

###While loop with continue statement

In [48]:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

```
1
2
4
5
6
```

###While loop with pass statement

In [52]:

```
n = 4
while n > 0:
  n = n - 1
  pass
  print(n)
```

```
3
2
1
0
```

###For loop with else

In [56]:

```
for i in range(1,5):
  print(i)
else:
  print("No break")
```

```
1
2
3
4
No break
```

###While loop with else

In [57]:

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

```
1
2
3
4
5
i is no longer less than 6
```

###Nested for loop

In [58]:

```
#Example: Write a nested for loop program to print multiplication table in Python

# outer loop
for i in range(1, 11):
    # nested loop
    # to iterate from 1 to 10
    for j in range(1, 11):
        # print multiplication
        print(i * j, end=' ')
    print()
```

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
```

In [59]:

```
#Another most common use of nested loop is to print various star and number patterns.

rows = 5
# outer loop
for i in range(1, rows + 1):
    # inner loop
    for j in range(1, i + 1):
        print("*", end=" ")
    print('')
```

```
*
* *
* * *
* * * *
* * * * *
```

###Nested while loop

In [61]:

```
fruits = ['apple', 'banana', 'cherry']
# outer loop
for x in fruits:
    # inner while loop
    count = 0
    while count < 5:
        print(x, end=' ')
        # increment counter
        count = count + 1
    print()
```

```
apple apple apple apple apple
banana banana banana banana banana
cherry cherry cherry cherry cherry
```

In [ ]: