Loops :- Loops are used to execute a block of code repeatedly as long as a certain condition is true or for a specific number of iterations.

Types :-

- while
- for

while loop :- The while loop executes a block of codes as long as a specified condition is true. It continuously checks the condition before each iteration and stops when the condition becomes false.

Syntax :-

```
while condition:
    # code block to be executed repeatedly
```

Code :-

```
candies = 10
while candies > 0:
    print ("Giving a candy to a friend!")
    candies -= 1
    print (candies)
```

O/P :-

```
Giving a candy to a friend!
```

for loop :- A for loop is a way to repeat a block of code for each item in a collection (like a list) or for a specific range of numbers.

Syntax :-

```
for variable in range (start, stop, step):
    # code block to be executed for each variable
```

Code :-

```
candies = 10
for i in range (0, candies):
    print ("given to friend")
```

O/P :- 0-9 given to friend

for loop for sequence :

The for loop is used to iterate over a sequence (such as a list, tuple, string, or dictionary) and execute a block of code for each item in the sequence.

Syntax :

```
for item in sequence:
    # code block to be executed for each item
```

Code :

```
message = "Hello, world!"
for i in message:
    print (i)
```

O/P :

```
H
e
l
l
o
,
 
w
o
r
l
d
!
```

length

```
length = len (message)
for i in range (length):
    print (i)
```

Nested loops :

Nested loops refer to the situation where one loop is placed inside another loop. This allows you to execute a set of instructions repeatedly.

Syntax :

```
for outer_var in outer_sequence:
    # code block of the outer loop
    for inner_var in inner_sequence:
        # code block of the inner loop
```

Code : Nested loop to generate a multiplication table from 1 to 5

```
for i in range (1,6):
    for j in range (1,11):
        print (f"{i} * {j} = {i * j}")
```

Break :
If during the execution of the loop Python interpreter encounters break, it immediately stops the loop execution and exits out of it.

Syntax :

```
while condition :
# code block inside the loop
  if some - condition :
    break # Exit the loop if the condition is met
```

Code :

```
candies = 10
for i in range (candies):
- Print ("Giving a candy to a friend!")
  If candies - i == 5 :
  Print ("only 5 candies left. stopping distribution")
  break
```

Continue :

Continue Statement is used to skip the rest of the current iteration in a loop and move to the next iteration immediately.

Syntax :

```
while condition : # code block inside the loop
  if some - condition
  continue # skip this iteration
```

Code :

```
candies = 10
for i in range (condition):
  if candies - i == 5 :
```

o/p Print ("only 5 candies left: skipping this turn!")

```
  continue
Print ("Giving a candy to a friend!")
```