

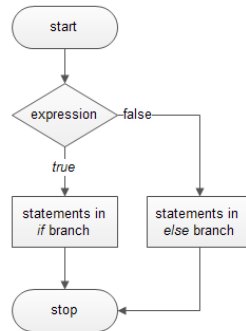
LOOPS

IF-ELIF-ELSE Statement

Used where there are different possible actions for different conditions.

Pseudocode:

```
if email is A or B or C:  
    email is Important  
elif email is D or E or ....:  
    email is Promotions  
else:  
    email is Spam
```



Nested IF

If-elif-else within another if-elif-else

Eg.

```
# check if number is greater than or equal to zero  
if x >= 0:  
    # check if it is equal to zero or greater than zero  
    if x == 0:  
        print('Number is zero')  
    else:  
        print('Number is positive')  
else:  
    print('Number is negative')
```

WHILE Loop:

Used to repeat a section of code an unknown number of times until a specific condition is met.

Eg. how many times a given number can be divided by 2 before it is less than or equal to 1

Pseudocode:

Choose a number

Set initial count = 0

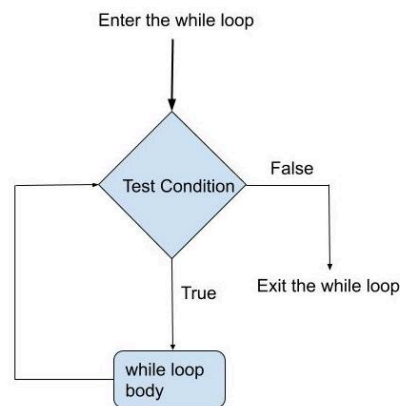
While the number > 1:

 Divide the number by 2

 Increase our count by 1

end

It repeats a statement or group of statements while a given condition is TRUE.



```
number = 5
count = 0
while number > 1:
    number = number / 2
    count += 1
print(count)
```

```
factors = []
num = 1
while num <= 100:
    if num % 10 == 0:
        factors.append(num)
    num += 1
print(factors)
```

FOR Loop

Used to repeat a specific block of code a known number of times

Different from WHILE Loop: number of times is not known beforehand in while loop

We know exactly how many times a loop will execute before the loop starts

For Loop	Nested For Loop
<pre>graph TD; Start([for each item in sequence]) --> Decision{Last item reached?}; Decision -- Yes --> Exit([Exit loop]); Decision -- No --> Body[Body of for]; Body --> Decision;</pre>	<pre>graph TD; Start([for each element in the sequence]) --> Decision1{Is this the last element of outer loop?}; Decision1 -- Yes --> EndOuter([end of outer loop]); Decision1 -- No --> Decision2{Is this the last element of inner loop?}; Decision2 -- Yes --> Decision1; Decision2 -- No --> InnerLoop[inner loop statements]; InnerLoop --> Decision2;</pre>
<pre>#Sum of numbers from 1-10 total = 0 for i in range(1,11): total = total + i</pre>	<pre>#If number divisible by 2 or 3 divisible = [] num = 1 for num in range(1,51): if num%2 == 0 or num%3 ==0: divisible.append(num) num += 1 print(divisible, num)</pre>

LOOP CONTROL STATEMENTS

To change the way a normal loop is functioning

Control Statement	Description	Code	
break	Terminates the loop and control shifts to the first statement outside it	<pre>#Get out of loop when 5 is present count = 0 while count < 10: count += 1 If count == 5: break print('inside loop', count) print('Outside while loop')</pre>	<pre> graph TD Start([Enter loop]) --> Test{test expression of loop} Test -- True --> Break{break?} Break -- Yes --> Exit([Exit Loop]) Break -- No --> Body[Remaining body of loop] Body --> Test Test -- False --> Exit </pre>
continue	Causes the loop to skip the remainder of its body and shifts to the next item in the sequence	<pre>#Print numbers 1-100 except multiples of 3 multiples = [] for i in range(1,101): if (i%3 == 0): continue else: multiples.append(i) print(multiples)</pre>	<pre> graph TD Start([Enter loop]) --> Test{test expression of loop} Test -- True --> Continue{continue?} Continue -- Yes --> Test Continue -- No --> Body[Remaining body of loop] Body --> Test Test -- False --> Exit([Exit Loop]) </pre>
pass	Used when the statement is required but we don't want any output. Generally used as a placeholder for functionality to be added later - do nothing	<pre>#Print TMLC on every 3rd count</pre>	<pre> graph TD Start([Enter loop]) --> Test{test expression of loop} Test -- True --> Pass{pass?} Pass -- True/No --> Body[Remaining body of the loop] Body --> Test Test -- False --> Exit([Exit loop]) </pre>