Session Tres

Loops

Sometimes you'll need to do repetitive things in your code. That's what loops are for!

We will discuss two types of loops:

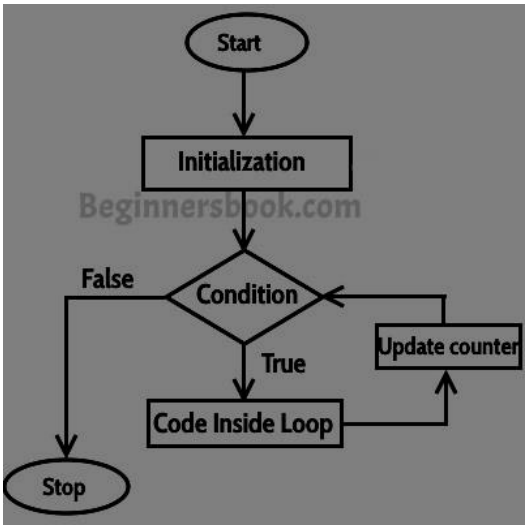
1. `while` Loop
2. `for` Loop

while loop

`while` loops repeat as long as a certain boolean condition is met. For example:

```
while True:
      print("Hahahaha")
```

This will repeat the print function endlessly. But we need to control how many times the loop runs, right? It's done by splitting it into three sections – initialization of the counter/iterator, condition, increment/decrement of the iterator.



Now let's see how we code it.

Suppose we have to count from 0 to 100 in our code.

```
count = 0               #  initial value of the iterator

while count <= 100:     # the condition for the loop to continue
      print(count)      # code inside the loop. It can be multiple lines
      count = count + 1 # update/increment the counter
```

Now make a program to count from 100 to 0.

Problems

1. Print all the even number from 1 to 50.
2. Take an integer input from the user, and use that integer to make a table as output.
3. Make multiplication tables for 1 to 20.
4. Write a program to find factorial of an input.
5. Take an integer as input and show if it's prime or not.

for loop

for loops are awesome. Because unlike while, you can do all three steps (initialization, condition, increment/decrement) in just one line!

```
# Prints out the numbers 0,1,2,3,4
for x in range(5):
    print(x)

# Prints out 3,4,5
for x in range(3, 6):
    print(x)

# Prints out 3,5,7
for x in range(3, 8, 2):
    print(x)
```

But there is something more to python for loops. We can run python for loops with lists.

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


primes = [2, 3, 5, 7]

for prime in primes:
      print(prime)
```

Problems:

Solve all the problems that is mentioned earlier, but this time with for loop!

Break and continue statement:

break statement does what it looks like – it breaks out of the loop!

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

With the continue statement we can stop the current iteration of the loop, and continue with the next:

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