Loops
Lesson Plan: Class 07 / PRG / 04

Overall goal of the lesson	**Children will be able to use while and for loops.**
Prior knowledge required	Concepts of instructions and programming.

MODULE 1: **Module time:** 35 minutes

Goal:	To get the children to use while loops .
Description:	Children will be able to find out the need for loops and use the syntax of while loops.
Material required:	**Physical:** Writing material to copy down the instructions written on blackboard. **Electronic:** A computer with a python compiler (version 3.x) for each student and the teacher. If there is an internet connection, https://www.codechef.com/ide can be used. A projector with the computer screen of the teacher projected for the students to see.
Procedure Summary:	Teacher will demonstrate the difference through contrasting examples and activities.

Procedure Details:	• Start by asking students to print "Hello" 5 times . (ESR : >>>print("Hello"*5) or 5 print statements)
	• Ask them to print "Hello", 100 times. (ESR : Too lengthy , >>>print("Hello"*100)).
	• Now, ask them to print numbers 1 to 5 . (ESR : 5 print statements or one statement with all the numbers)
	• Ask students to print first 1000 natural numbers. (ESR : This is too lengthy , too much of effort)
	• So, you could see that there are certain tasks which require are repetitive or has a pattern and takes a lot of effort. Like printing 1 to 1000 would require us to write 1000 print statements. But it has a simple pattern which can be followed by a computer and simply do this for us.
	• Imagine how easy would it be if computer could do it on his own by just a few instructions from our side .
	• So , let's study about the concept called **loops.**
	• **Loops** has a literal meaning that it is a process, the end of which is connected to the beginning.
	• We are going to study about two types of loops and let's see how they simplify things for us.
	  ○ while loop
	  ○ for loop
	
	While Loop :
	• What does while mean ? (ESR : until)
	• So , we set a condition in it and until the condition is met , the instructions under it keep repeating.
	• Let's learn the syntax of this loop :
	  ○ >>>while <condition> :
	  < Instructions >
	• Let's take an example , Say we want to print first 5 natural numbers .
	We can write ,
	a = 1 #Some variable

	while a<=5 : #Notice Colon
	print (a) #Notice Indentation
	a=a+1
	
	Here , the output will be
	1
	2
	3
	4
	5
	● Let's analyze (dry run) the program .
	We initiated a variable to start the program (a = 1) . Then we checked for the condition (While a<=5 , here 1 <= 5 is true so the program prints(a) i.e. 1 and then makes a=a+1 i.e a=2 , checks the condition again a<=5 , 2<=5 , true , therefore again prints(a) , 2 and a = a+1 , i.e. 3 ... so on until print(a) , prints 5 . now a= a+1 , a=6 ; a<=5 , 6<=5 , False , therefore It comes out of the loop)

Assessment:	● Print "Hello!" , 100 times , using while loop ● Ask students to write a program to print the first 100 whole numbers . ● Print first N natural numbers using while loop. ● Print numbers from 200-500 using while loop. Extend it to start from some N and till some M
Information Broadcast:	

MODULE 2: **Module time:** 35 minutes

Goal:	To get the children to use for loops (in sequence , range , continue and break , else)
Description:	Children will be able to use for loops .
Material required:	**Physical:** Writing material to copy down the instructions written on blackboard. **Electronic:** A computer with a python compiler (version 3.x) for each student and the teacher. If there is an internet connection, https://www.codechef.com/ide can be used. A projector with the computer screen of the teacher projected for the students to see.
Procedure Summary:	Teacher will demonstrate through examples.
Procedure Details:	**For Loop :** ● Now , let's study about second type of loops , called for loop. ● Now , you ask what's the difference between two loops : The *for* loop runs for a fixed amount. while the *while* loop runs until the loop condition is true. **Range** ● Let's take an example : ○ for x in range(0,3): #notice colons print(x) #notice indentation ○ Will print the following : 0 1 2 i.e. the values of x were equal to 0 in the first go , then 1 and finally 2 . The loop stopped when x was equal to 3. We can also do the above program using range(3) because range by default starts it from 0 . ● What can we do if we want to print numbers from 1 to 5 ?

ESR : for x in range (1 ,6):

print(x)

- So, you see, here we were using for loop in a range , we were covering all the values of the range so for range(a , b) , loop repeats itself from a to b-1 .

Sequence :

- Now , let's see some other methods of using **for loop** .
- We can also use a single variable to read all the elements of a sequence
- Let's say ['hello' , 12 , 'world'] is a sequence with three elements in it . We want to print all of these .

```
for  i  in ['hello' , 12 , 'world'] :
        print (i)
```

Will print :

hello

12

world

We can also store the sequence in a variable and use that directly . e.g the above program can also be done as :

```
a  = ['hello' , 12 , 'world']
for  i  in a :
        print(i)
```

- Ask students to create a list of 10 names and print all of them using a for loop.

Continue and break :

- Loops iterate over a block of code until test expression is false, but sometimes we wish to terminate the current iteration or even the whole loop without checking test expression.The break and continue statements are used in these cases.
- The break statement terminates the current loop and resumes execution at the next statement outside the loop.
- The most common use for break is when some we need to exit the loop due to some external condition. The break statement can be used in both while and for loops.
- Let's take the following example :

```
for i in range(10):
        if i==5:
                break
        print(i)
   i=i+1
```

Will print :

0
1
2
3
4

So , you can see that the we came out of loop when value of i was 5.

- The continue statement returns the control to the **beginning** of the while loop. The continue statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop. The continue statement can be used in both while and for loops.
- Let's take an example :

```
for i in range(10):
        if i==5:
                continue
        print(i)
```

Will print :

0
1
2
3
4
6 **#Notice 5 is missing**
7
8
9

- You'll see that **5** was missing from the output , can you think of why did it happen ?

The loop worked fine with the values of i from 0 to 4 but when i was equal to 5 , it checked if i==5 and it was true , therefore it executed the **continue** statement and as discussed , **continue** sends the control to the beginning of the loop .

Therefore , instead of going to print(i) , it went to the beginning and changed value of i by 1 in the range .

for-else :

- **We know that a loop can end in two ways :**
  - **When it completes all the iterations**
  - **When it finds a break**
- **We might be interested to know that through which condition did the loop end .**
- One method is to set a flag and then check it once the loop ends. Another is to use the else clause.
- The else clause executes when the loop completes normally. This means that the loop did not encounter any break.
- Consider this example :

```
for i in range(10):
        print(i)
        if i == 9:
                print("Too big - I'm giving up!")
                break
else:
        print("Completed successfully")
```

- What do you think will be the output of this particular code?

ESR :

0
1
2
3
4
5
6
7
8

	9 **Too big - I'm giving up!** You see, now we know that the loop didn't complete. But , got terminated because of a break .
Assessment:	● Print "Hello!" , 100 times , using for loop ● print first 100 natural numbers using for loop. ● print numbers from 200-500 using for loop. ● print multiplication tables for the following : ○ 2 ○ 5 ○ 19 ○ 574 ● print first 50 prime numbers
Information Broadcast:	