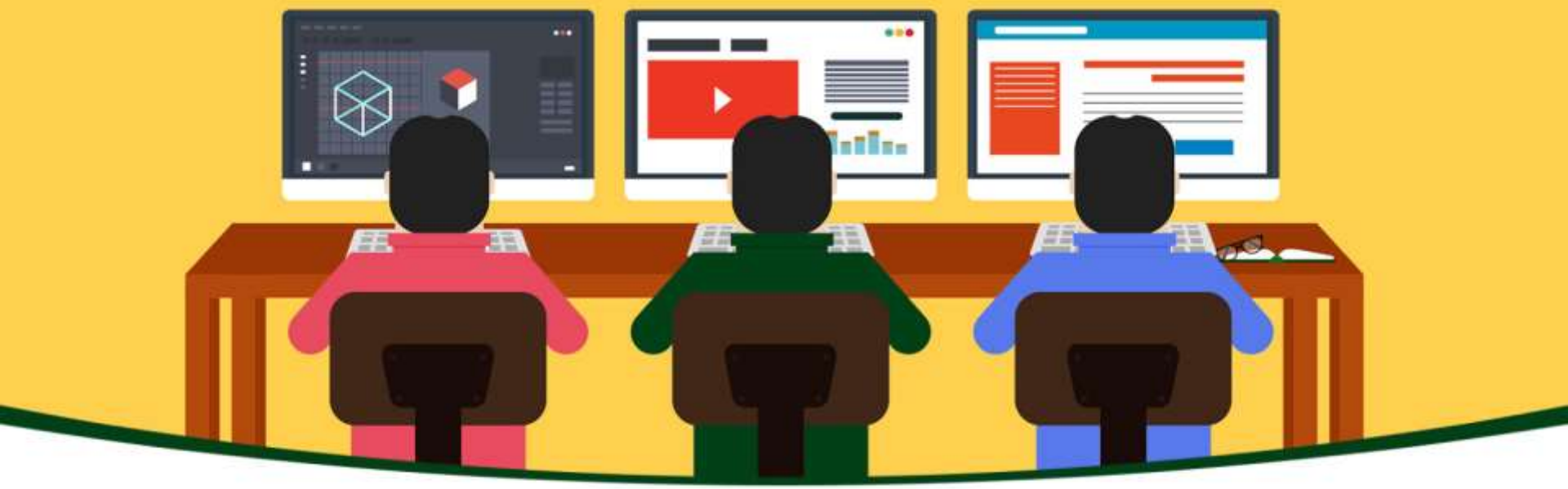


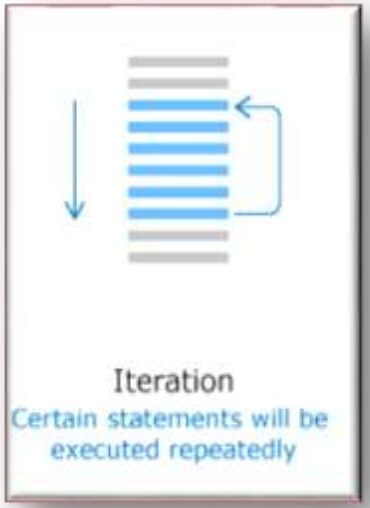
RAJALAKSHMI ENGINEERING COLLEGE

PYTHON PROGRAMMING

CONTENTS

- Iteration Control Structure(Looping)
  - while
  - for
  - range function
  - break
  - continue
  - Nested loops

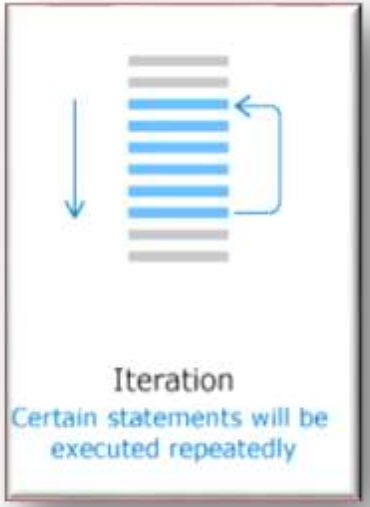
Iteration Control Structure



Iteration Control Structures

while loop

for loop



break statement

continue statement

pass statement

Iteration Control Structures

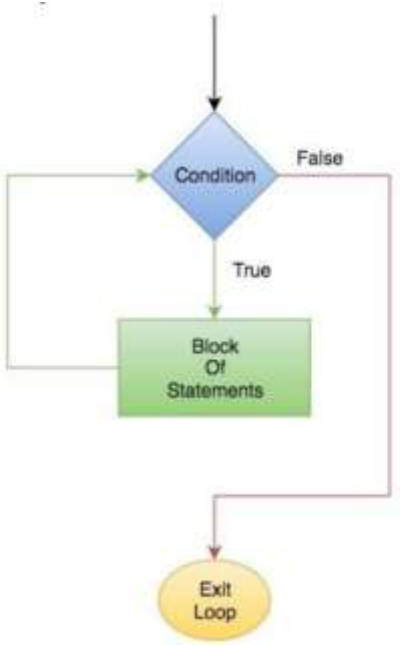
while loop

Repeats the execution of the statements within the while block when the condition is true . when the condition is false exits the while block.

Syntax

```
while condition:
        statement(s)
```

Flowchart of While Loop



Example

```
x = 0
while (x < 5):
        print(x)
        x = x + 1
Output :
0
1
2
3
4
```

```
x = 1
while (x <= 5):
        print("REC")
        x = x + 1
Output :
REC
REC
REC
REC
REC
```

Example

Create a program that computes the average of a collection of values entered by the user. The user will enter 0 as a sentinel value to indicate that no further values will be provided. Your program should display an appropriate error message if the first value entered by the user is 0.**Hint: Because the 0 marks the end of the input it should not be included in the average.**

Input:
List of Numbers
Eg:1,2,3,9,8,7, 0

Output:
Find Average

Logic:
Find Sum ,then…
Average=Sum/total no of values

Total no of values???
Find Sum and Total no of values
Finally find average

```
sum=0
no_of_values=0
num=int(input())
if(num==0):
    print(0)
else:
    while(num!=0):
        sum=sum+num
        no_of_values+=1
        num=int(input())
    average=sum/no_of_values
    print("The average is:",average)
```

Prog. To find **digit sum**

```
num = int(input("No.?"))
ds = 0
while num>0 :
        ds = ds +num % 10
        num = num // 10
print("Digit Sum :", ds)
```

Prog. To find **reverse**

```
num = int(input("No.?"))
rev = 0
while num>0 :
        d =  num % 10
        rev = rev*10 + d
        num = num // 10
print("Reverse :", rev)
```

Example

Lets take an Airport Scenario,

Conveyer belt rotation for collecting baggage

```
#counting conveyer belt rotation
no_of_baggages=int(input("enter the total no of baggages:"))
baggages_picked=0
rotation_cnt=0
while(no_of_baggages>0):
    baggages_picked=int(input("Enter baggages picked:"))
    no_of_baggages-=baggages_picked
    rotation_cnt+=1
    print("Remaining Baggages", no_of_baggages)
print("No of rotation:",rotation_cnt)
```

checks whether it is a palindrome or not

```
n=int(input("Enter number:"))
temp=n
rev=0
while(n>0):
    dig=n%10
    rev=rev*10+dig
    n=n//10
if(temp==rev):
    print("The number is a palindrome!")
else:
    print("The number isn't a palindrome!")
```

```
Case 1
Enter number:121
The number is a palindrome!

Case 2
Enter number:567
The number isn't a palindrome!
```

Iteration Control Structures

for loop

Syntax

```
for iterating_variable in sequence:
    statement(s)
```

Repeats the execution of set of statements for a specific number of times.

```
for number in 1,2,3,4,5:
    print("The number is:",number)
```

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

for loop with string

e.g.1:

```
for ch in "Hello" :
        print(ch)
```

Output:

H
e
l
l
o

e.g. 2:

```
T = "Hello"
for ch in T :
    print(ch)
```

Output:

H
e
l
l
o

for loop with string

e.g.1:

```
T = "Hello"
Len= len(T)
for i in range(Len) :
    print(T[i])
```

Output:

H
e
l
l
o

e.g. 2:

```
T = "Hello"
Len= len(T)
for i in range(0,Len) :
    print(T[i])
```

Output:

H
e
l
l
o

What if we want the loop to run from 1 to 100?

Should we manually create a sequence of values from 1 to 100?

In Python, there is an easy way to achieve this by using a built-in function

range(start , end, step)

start - Starting number of the sequence

end - Generate number up to end, but not including this number

step - Difference between each number in the sequence

```python
start=1
end=10
step=2
for number in range (start, end, step):
    print("The current number is ", number)
```

Example

```
for i in range(1,5):
    print(i)

Output :
1
2
3
4
```

```
for i in [1,2,3,4] :
    print("REC")

Output :
REC
REC
REC
REC
```

```
for i in range(5):
    print(i)

Output :
0
1
2
3
4
```

```
for i in range(2,9,2):
    print(i)

Output :
2
4
6
8
```

Iteration Control Structures

break

When we want to stop a loop based on an external condition, we can use **break statement**

```
sum=0
n=0
for a in 1,2,3,4,5,0:
    if(a==0):
        break
    sum=sum+a
    n=n+1
average=sum/n
print("the average is:",average)
```

Iteration Control Structures

continue

When we want to skip the remaining portion of the loop and continue with next iteration , we can use **continue statement**

```
for passenger in "A","A", "FC", "C", "FA",  "SP", "A", "A":
    if(passenger=="FC" or passenger=="FA"):
        print("No check required")
        continue

    if(passenger=="SP"):
        print("Declare emergency in the airport")
        break

    if(passenger=="A" or passenger=="C"):
        print("Proceed with normal security check")

    print("Check the person")
    print("Check for cabin baggage")
```

Iteration Control Structures

pass

Pass statement is never executed. Behaves like a Placeholder for future code

```
x = "Joy"
if x == "John":
    print ("Name:",x)
elif x == "Joy":
    pass
else:
    print ("in else")
```

Iteration Control Structures

Nested loops

A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop":

Syntax :

```
initialization
while(condition):
        initialization of inner loop
        while(conition):
                ------
                ------
                Update expression of inner loop
Update expression of outer loop
```

Example

```
# To Print Pyramid
'''
1
1 2
1 2 3
1 2 3 4'''
i=1
while(i<=4):
      j=1
      while(j<=i):
            print(j,end='  ')
            j+=1
      print("")
      i+=1

# To Print Pyramid
'''
5
5 4
5 4 3
5 4 3 2
5 4 3 2 1'''
i=5
while(i>=1):
      j=5
      while(j>=i):
            print(j,end='  ')
            j-=1
      print("")
      i-=1
```

Syntax

```
for iterating_var in sequence:
    for iterating_var in sequence:
        statements(s)
    statements(s)
```

```
# To Print Pyramid
'''
1
1 2
1 2 3
1 2 3 4'''

for i in range(1,5):
    for j in range(1,i+1):
        print(j,end=' ')
    print("")


# To Print Pyramid
'''
5
5 4
5 4 3
5 4 3 2
5 4 3 2 1'''

for i in range(5,0,-1):
    for j in range(5,i-1,-1):
        print(j,end=' ')
    print("")
```

```
for  i  in range ( 1, 6 ):
        print( )
        for j  in range (1,  i + 1):
                print("@", end=" ")
```

```
@
@@
@@@
@@@@
@@@@@
```

Print the Prime Number between the given Interval.

```
lower_value = int(input ("Enter the Lowest Range Value: "))
upper_value = int(input ("Enter the Upper Range Value: "))

print ("The Prime Numbers in the range are: ")
for number in range (lower_value, upper_value + 1):
    if number > 1:
        for i in range (2, number):
            if (number % i) == 0:
                break
        else:
            print (number,end=" ")
```

```
Enter the Lowest Range Value: 10
Enter the Upper Range Value: 50
The Prime Numbers in the range are:
11 13 17 19 23 29 31 37 41 43 47
```


THANK
YOU