

Python

Control flow

Chapter 14



Suresh techs

Suresh techs – Technology and Entertainment

Subscribe

Subscribe: You don't miss our future videos

Doubts

Follow In Instagram
#sureshtechs

- Link in the description
- Clear your doubts chatting with me

Very important

- Control flow statements are very important to build complex and performant programs in any language.

Suresh Techs

Real life scenarios

- Need six for last ball to win
- RRR movie released, if the rating is below less than or equal to 5 it is average, if it is between 5 and 8 then hit, if it is greater than 8 then super hit
- How do we do it in programming?

Suresh Techs



IF

If

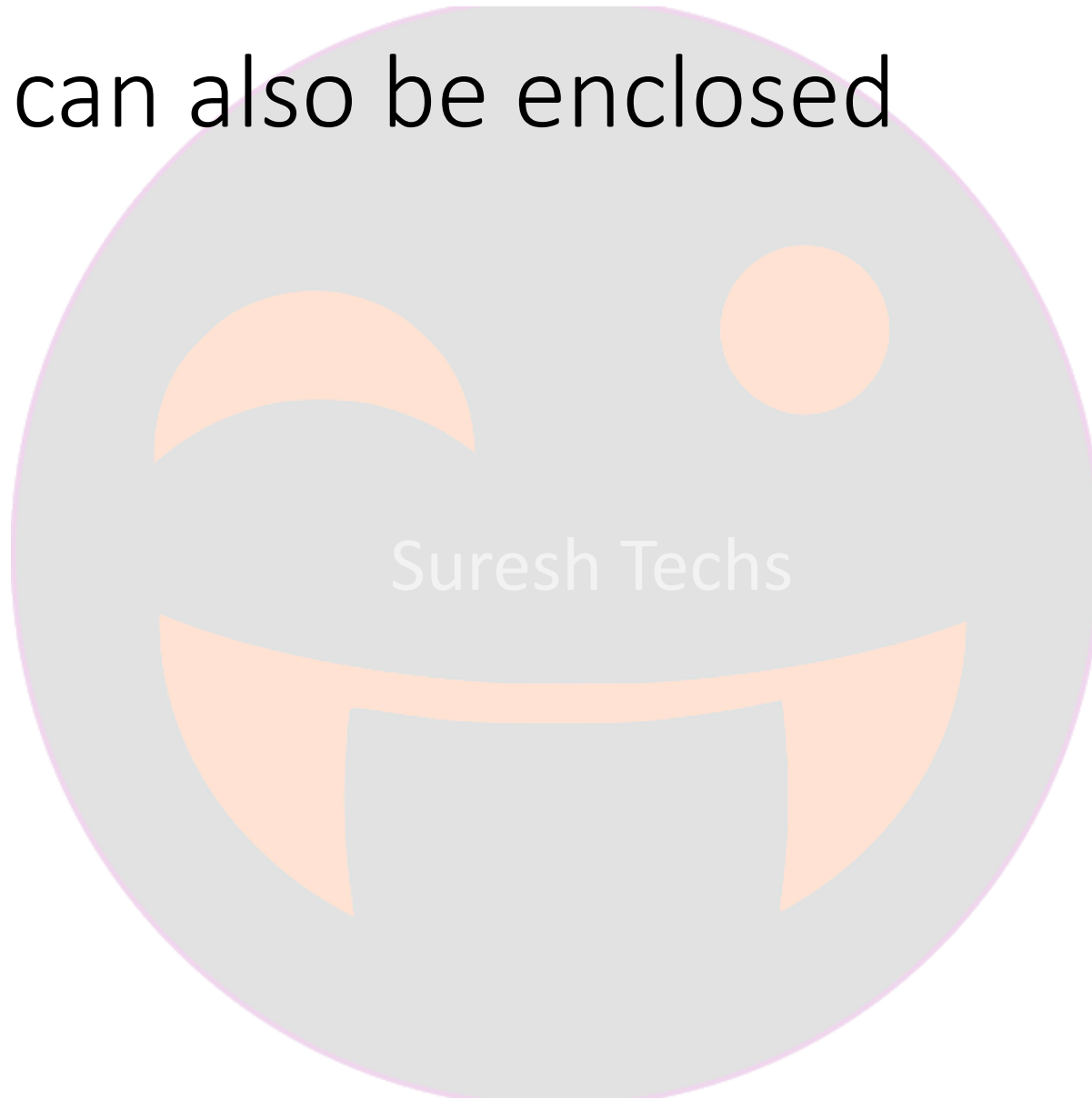
- if condition:
 # Statements to execute



Suresh Techs

Condition can also be enclosed

- If (x>10):
 #statements



if else

- What if you want to do something if the condition doesn't meet
- if (condition):

Executes if block

else:

Executes else block

Suresh Techs

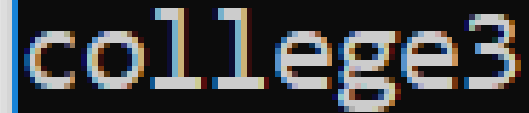
Let's write a program to check whether a number is even or odd?

Nested if

- If inside another if
- Write a program to check whether a number is divided by both 2 and 4
- Ex:
 - input: 8 – congrats, divided by both 2 and 4.
 - Input: 10 – sorry
 - Input: 7 – sorry

Suresh Techs

if-elif ladder



college3

- Counselling for B.Tech colleges.
- Based on EAMCET rank we prefer colleges...
- rank = 37400, write a program to assign college1 if the rank is less than or equal to 1000, assign college2 if the rank is between 1000 and 10000, assign college3 if the rank is between 10000 and 40000. If the rank is above 40000 then say "Sorry no colleges available for your rank"

Shorthand if

- For single statements we can use shorthand operation

```
x=10  
if x<20: print('greater')
```

greater

Shorthand if else

- If else can be written one line when there is only one statement to be executed in both if and else
- Syntax: statementIfTrue if condition else statementIfFalse

```
x=10
if (x%2==0):
    print('even')
else:
    print('odd')
```

```
print('even') if (x%2==0) else print('odd')
```



FOR LOOP

Iterable?

- Any sequence which you can iterate over is called iterable
- Ex: List, tuple, string etc

Suresh Techs

Let's talk about range

- Used to generate sequence of numbers
- `range(10)`
- Returns sequence of numbers starting from 0 (by default) and increments by 1 (by default) and stops before a specified number
- `range(start, stop, step)`

For Loop

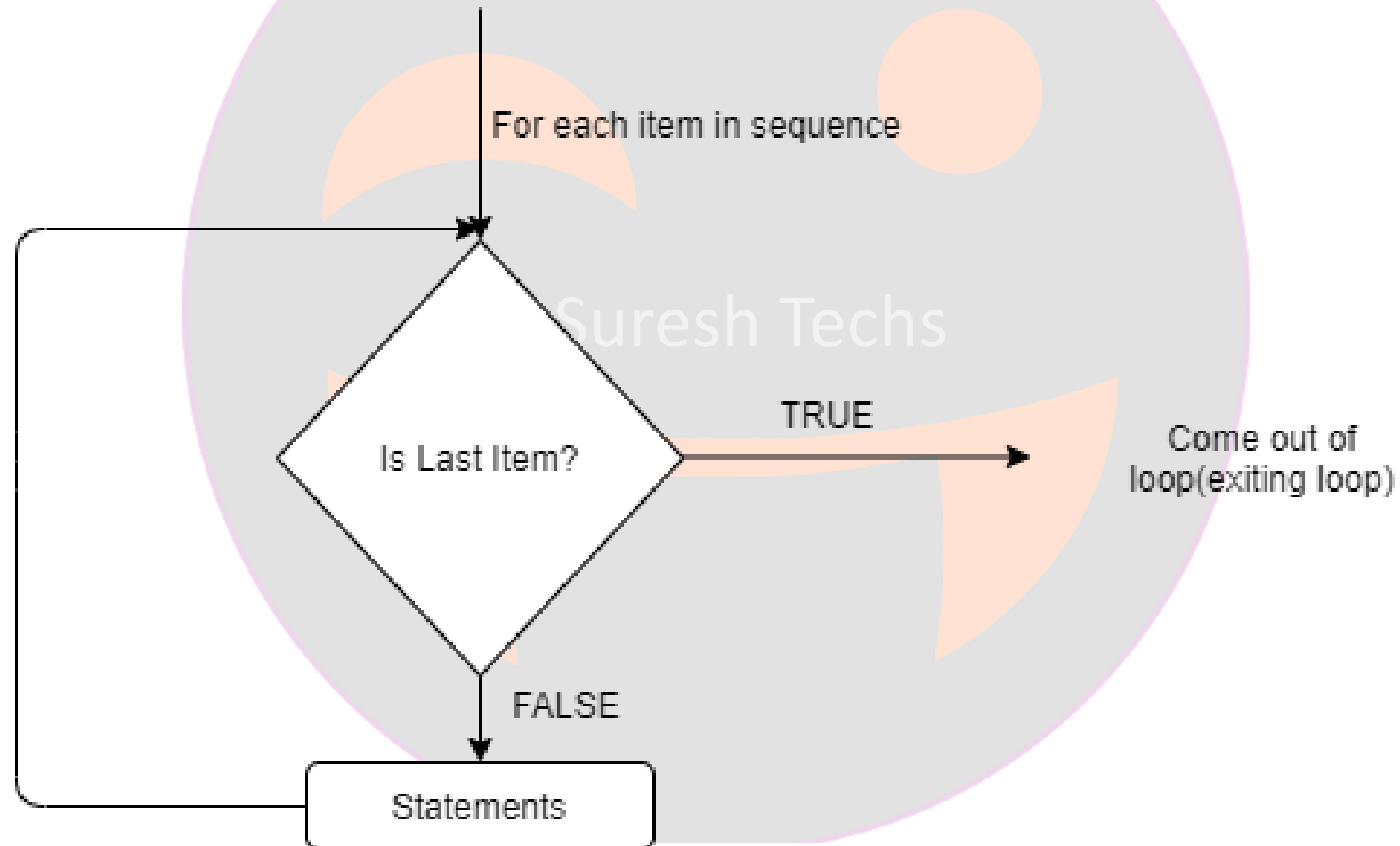
- Used for sequential traversing
- Fixed number of iterations(definite)

```
for var in iterable:  
    # statements
```

The var takes the value of next item from the iterable each time through the loop

Suresh Techs

For loop flow



Few programs

- Write a program to print 1 to 10 numbers
- Write a program to print numbers from 100 to 1 in one line
- Write a program to print your 2 characters from your name and repeat it from 1 to 10

Suresh Techs

For – iterating over list

```
friends = ['suresh', 'john', 'marry']  
for element in friends:  
    print(element)
```

Write a program to iterate over a tuple

```
friends = ('suresh','john','marry')  
for element in friends:  
    print(element)
```

How to iterate over a dictionary?

```
numbers = {1:'one',2:'two',3:'three'}  
  
for element in numbers:  
    print(element)  
    print(numbers[element])
```

```
1  
one  
2  
two  
3  
three
```

Check if your name has a vowel?

- Find vowels and consonant in your name
- Find if your name has a vowel in your name?
- **How do we come out of the loop, how to do we control the loops?**

Suresh Techs

How do you control this loop?

- Control statements
 - break
 - continue
 - pass
- **break: Returns control out of the loop directly**
- continue: Returns control to the beginning of the loop
- pass: Used as a placeholder for future implementation of loops, functions etc.

Suresh Techs

break

```
numbers = {1:'one', 2:'two', 3:'three'}  
  
for element in numbers:  
    print(element)  
    break  
    print(numbers[element])
```

1

continue

```
numbers = {1:'one', 2:'two', 3:'three'}  
  
for element in numbers:  
    print(element)  
    continue  
    print(numbers[element])
```

1
2
3

pass

- Used as a placeholder for future implementation of loops, functions etc.
- Interpreter will ignore a comment whereas pass will not be ignored
- However, nothing happens when the pass is executed
- Use case: while creating projects, sometimes we will keep functions but we are not sure what kind of implementation has to be done inside it. So we will just put pass and later we will write some implementation logic inside it. So that there will be no errors till that time.

pass

```
numbers = {1:'one',2:'two',3:'three'}  
for element in numbers:  
    pass
```

Program to print total marks



TRY YOUR OWN



WHILE LOOP

While loop

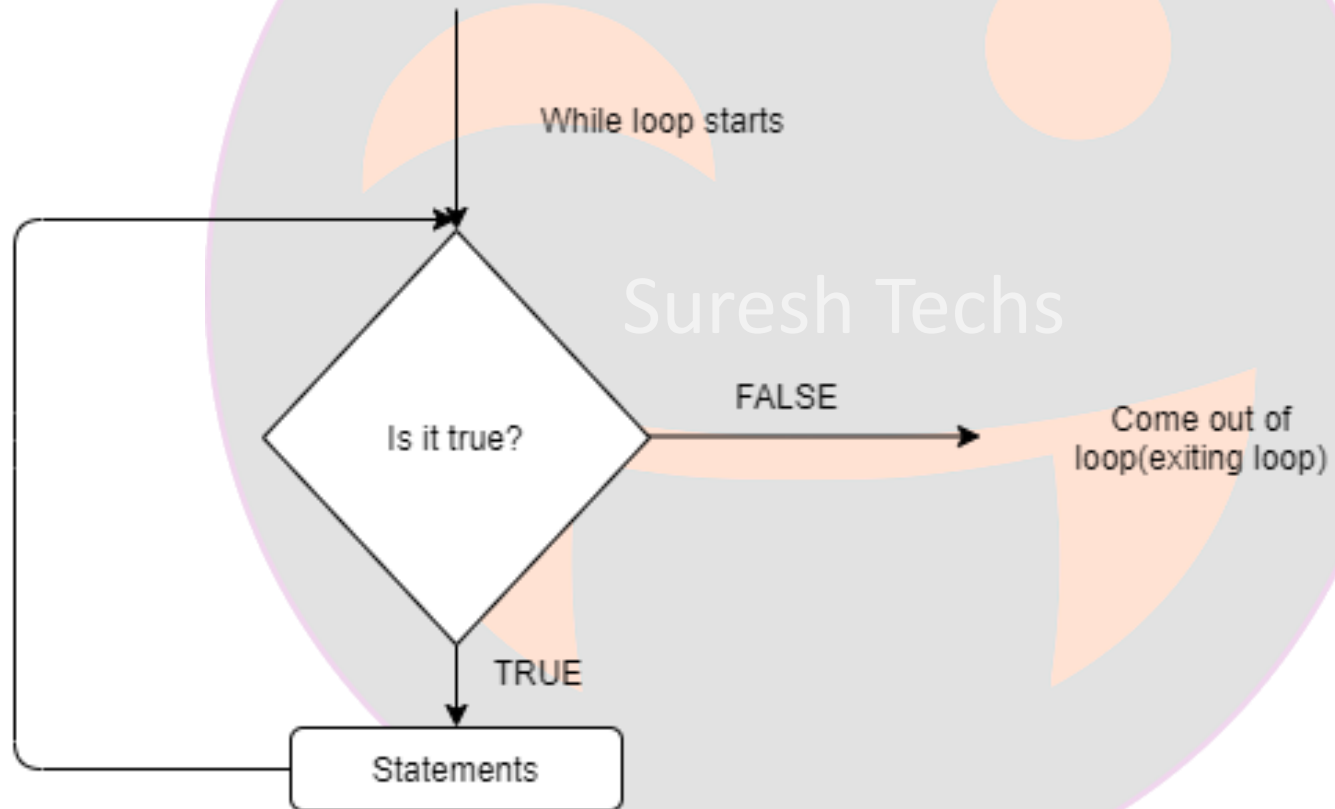
- indefinite iteration
- You never know how many number of times loop executes

while expression:

body of the loop

Suresh Techs

While loop flow



Programs

- Write a program to print 1 to 10
- Book cricket:
 - Consider you have a book of 30 pages
 - Flip the book till you get 500
 - Print all the values(scores)
 - Print number of flips taken to complete the 500 score

Suresh Techs

break, continue and pass

- Same as for loop, no difference at all

Suresh Techs



Few more, not required now.

- enumerate()
- iteritem()
- items()
- zip()
- sorted()
- reversed()





PYTHON

చిన్న ప్రో చిట్టికలో వచ్చేస్తా

#sureshtechs