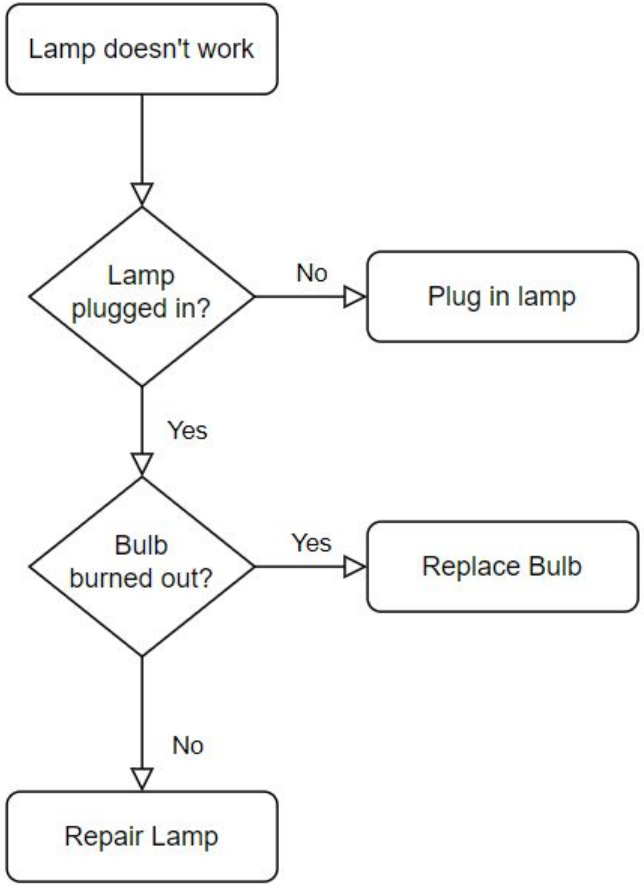
MODULE 3

Control Structures

CONDITION


Lamp doesn't work
Lamp plugged in?
No
Plug in lamp
Yes
Bulb burned out?
Yes
Replace Bulb
No
Repair Lamp

DECISION MAKING AND BRANCHING

- When problem involves decision making to do specific task conditional statements can be used.
- The following are the conditional statements provided by Python.
- if
- if-else
- nested if
- if-elif statements

IF - STATEMENT

- Condition must be a statement that evaluates to a boolean value (True or False).

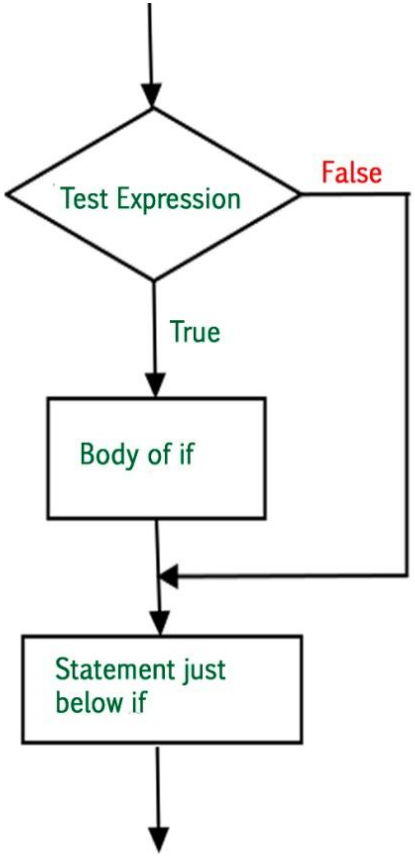
- **Syntax:**

if test_expression/condition:

statements

Ends with Colon

Note: If the condition is false , this "if statement" is skipped



EXAMPLE 1

- # if statement example

Indentation

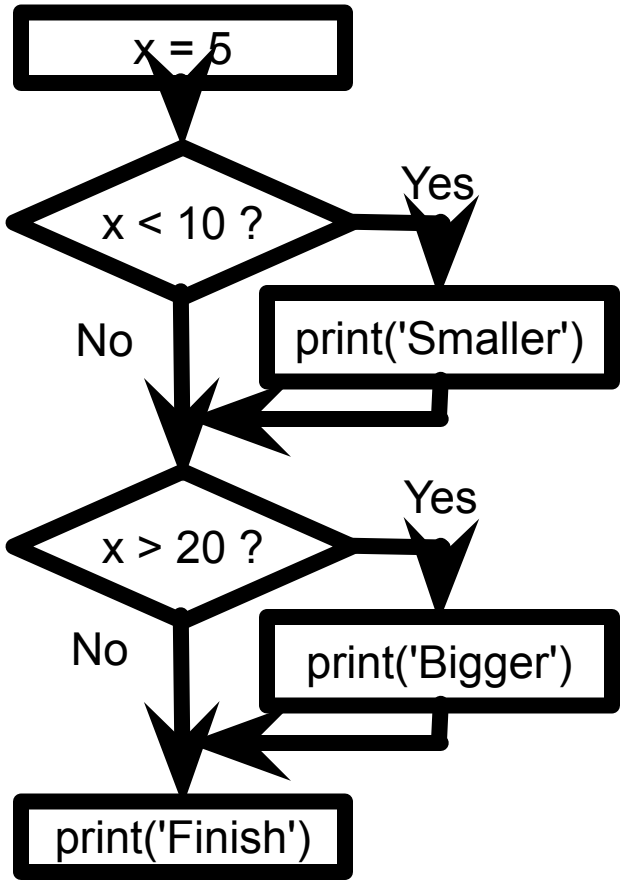
```
if 10 > 5:
    print("10 greater than 5")
print("Program ended")
```

Output:

10 greater than 5
Program ended

- Indentation(White space) is used to delimit the block of code. As shown in the above example it is mandatory to use indentation in Python3 coding.

EXAMPLE 2



Program:

```
x = 5
if x < 10:
    print('Smaller')
if x > 20:
    print('Bigger')

print('Finish')
```

Output:

```
Smaller
Finish
```

Comparison operator

- Boolean expressions ask a question and produce a Yes or No result which we use to control program flow
- Boolean expressions using comparison operators evaluate to True / False or Yes / No

Comparison operators look at variables but do not change the variables

Python	Meaning
<	Less than
<=	Less than or Equal to
==	Equal to
>=	Greater than or Equal to
>	Greater than
!=	Not equal

Remember: “=” is used for assignment.

EXAMPLE 3

```
x = 5
if x == 5 :
    print('Equals 5')
if x > 4 :
   print('Greater than 4')
if  x > 5 :
    print('Greater than 5')
if x > 6 : print(‘Greater than 6')
if x <= 5 :
    print('Less than or Equals 5')
if x != 6 :
    print('Not equal 6')
```

OUTPUT

Equals 5

Greater than 4

Less than or Equals 5

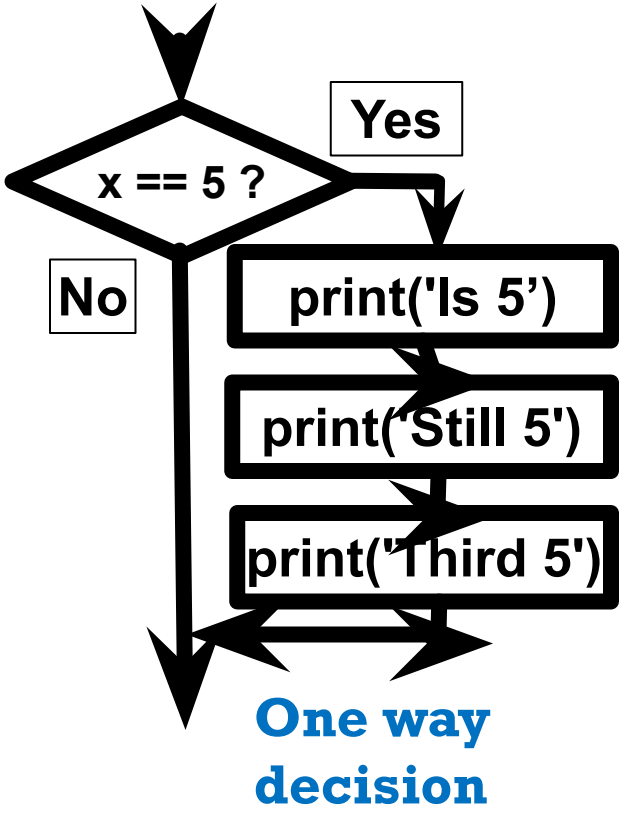
Not equal 6

EXAMPLE 4

```
x = 5
print('Before 5')
if  x == 5 :
    print('Is 5')
    print('Is Still 5')
    print('Third 5')
print('Afterwards 5')
print('Before 6')
if x == 6 :
    print('Is 6')
    print('Is Still 6')
    print('Third 6')
print('Afterwards 6')
```

```
Before 5

Is 5
Is Still 5
Third 5
Afterwards 5
Before 6
Afterwards 6
```



EXAMPLE 5

Write a Python program that checks if the word "World" is present in the string “s”. If the word "World" is found, append an exclamation mark to the string s. Finally, print the modified string.
Note: Use if statement

Input :
Hello World

Output:
Hello World!

EXAMPLE 5

```
# if statement example

s = "Hello World"

if  "World" in s:

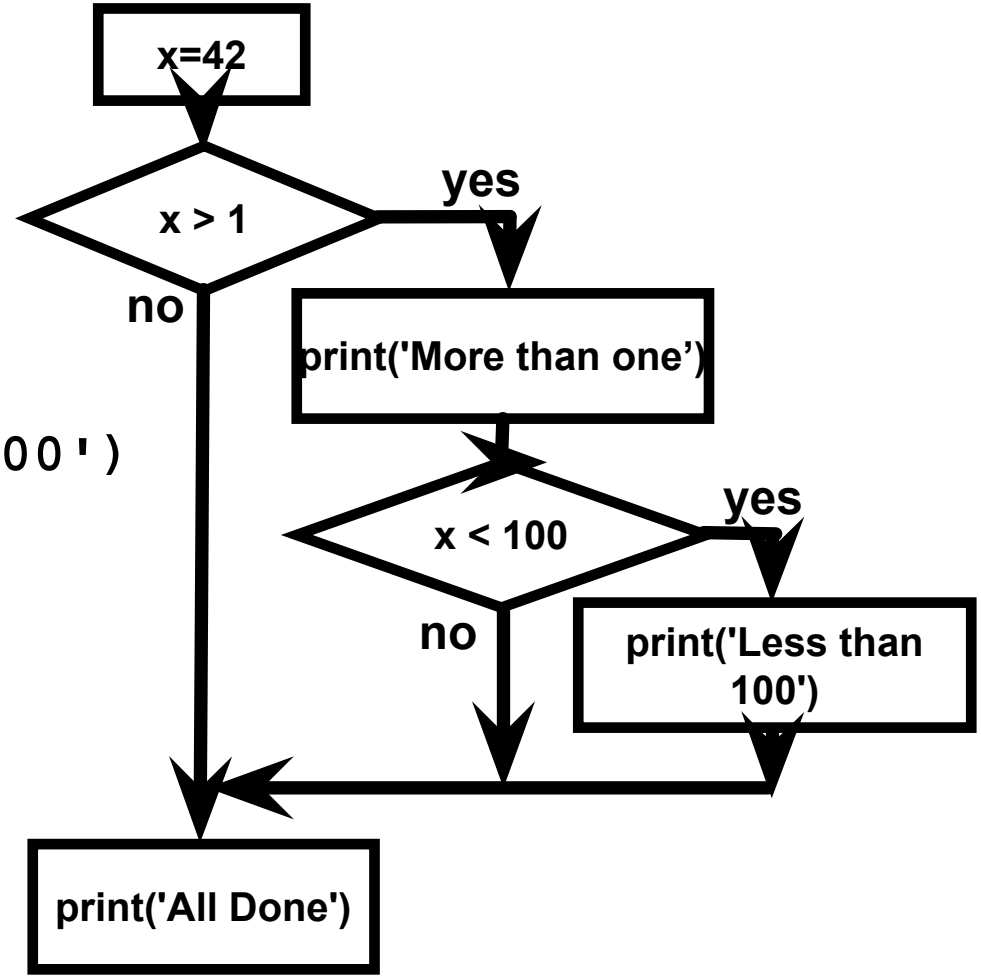
    s=s+"!"

print(s)
```

Output:
Hello World!

Nested if

```
x = 42
if x > 1 :
    print('More than one')
    if x < 100 :
        print('Less than 100')
print('All done')
```

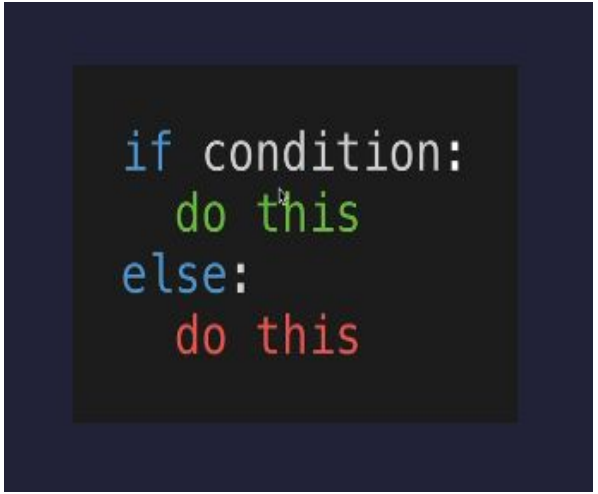


Output:

More than one
Less than 100
All done

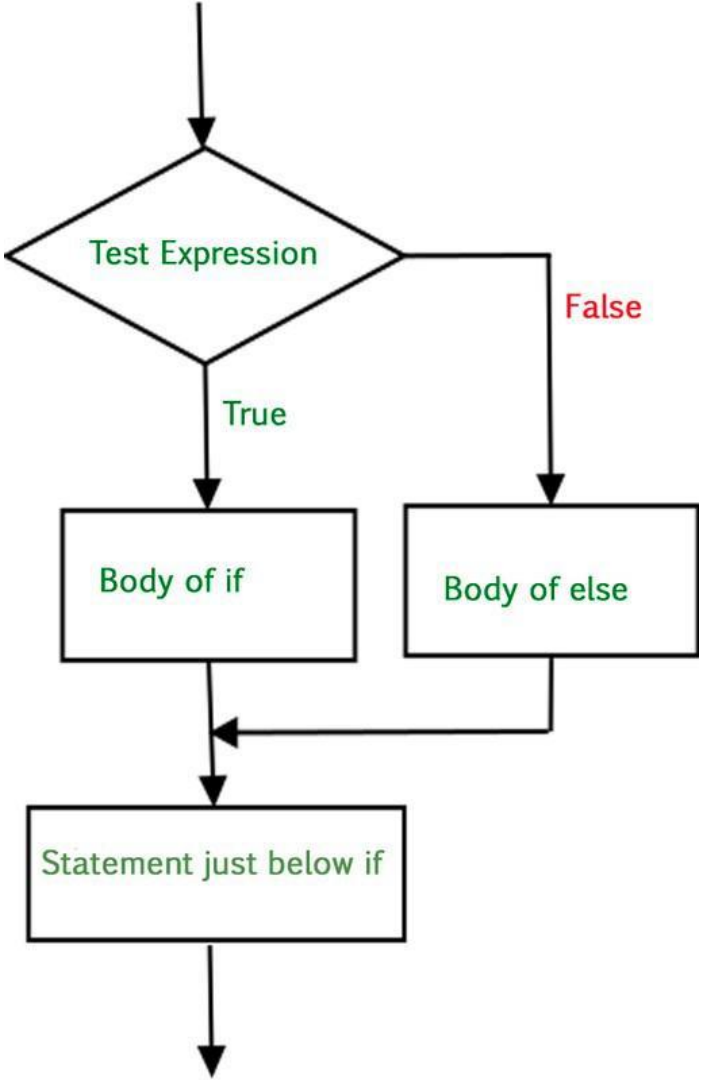
IF ELSE STATEMENT

- It has both true and false blocks. Either true or false block would be executed based on the condition evaluation of the if statement.



- **Syntax:**

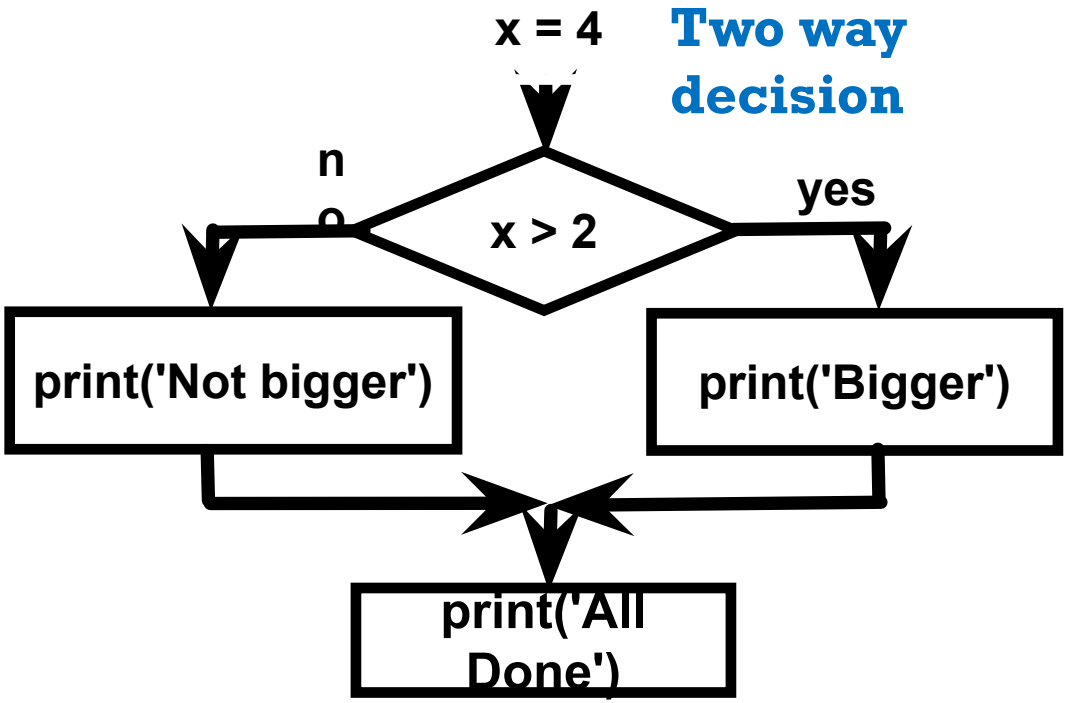
```
if test_expression:
    statements
else:
    statements
```


Test Expression
False
True
Body of if
Body of else
Statement just below if

EXAMPLE 1

```
x = 4
if x > 2 :
    print('Bigger')
else :
    print('Smaller')

print('All done')
```

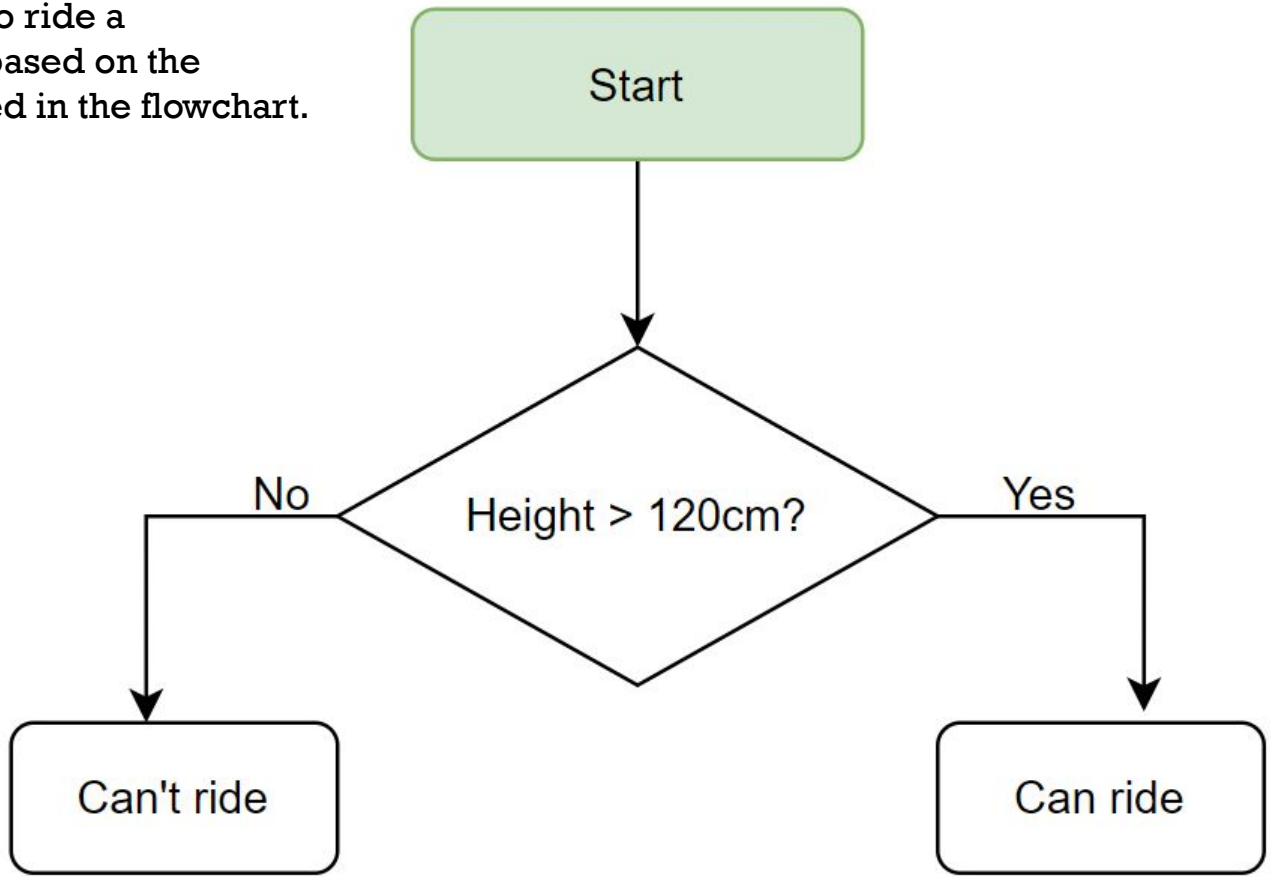


Output:

Bigger

All done

Write a code to ride a rollercoaster based on the condition stated in the flowchart.



EXAMPLE 2

Write a Python program that defines a variable x and sets it to the value 1000. Use an if-else statement to check if x is less than 100. If x is less than 100, print "x is less than 100". Otherwise, print "The value of x is 1000". Finally, print "End of Program".

EXAMPLE 2

```
# if else statement example
x = 1000
if (x<100):
    print(“x is less than 1000")
else:
    print(“The value of x is
1000")
print ("End of Program")
```

Output:

The value of x is 1000
End of Program

EXAMPLE 3

Check whether the number is odd or even

```
# if else statement example

N=int(input("Enter n:"))

if (N%2==0):

    print(N,"is even")

else:

   print(N,"is Odd")
```

Output:

Enter n: 5

5 is Odd

NESTED IF ELSE STATEMENT

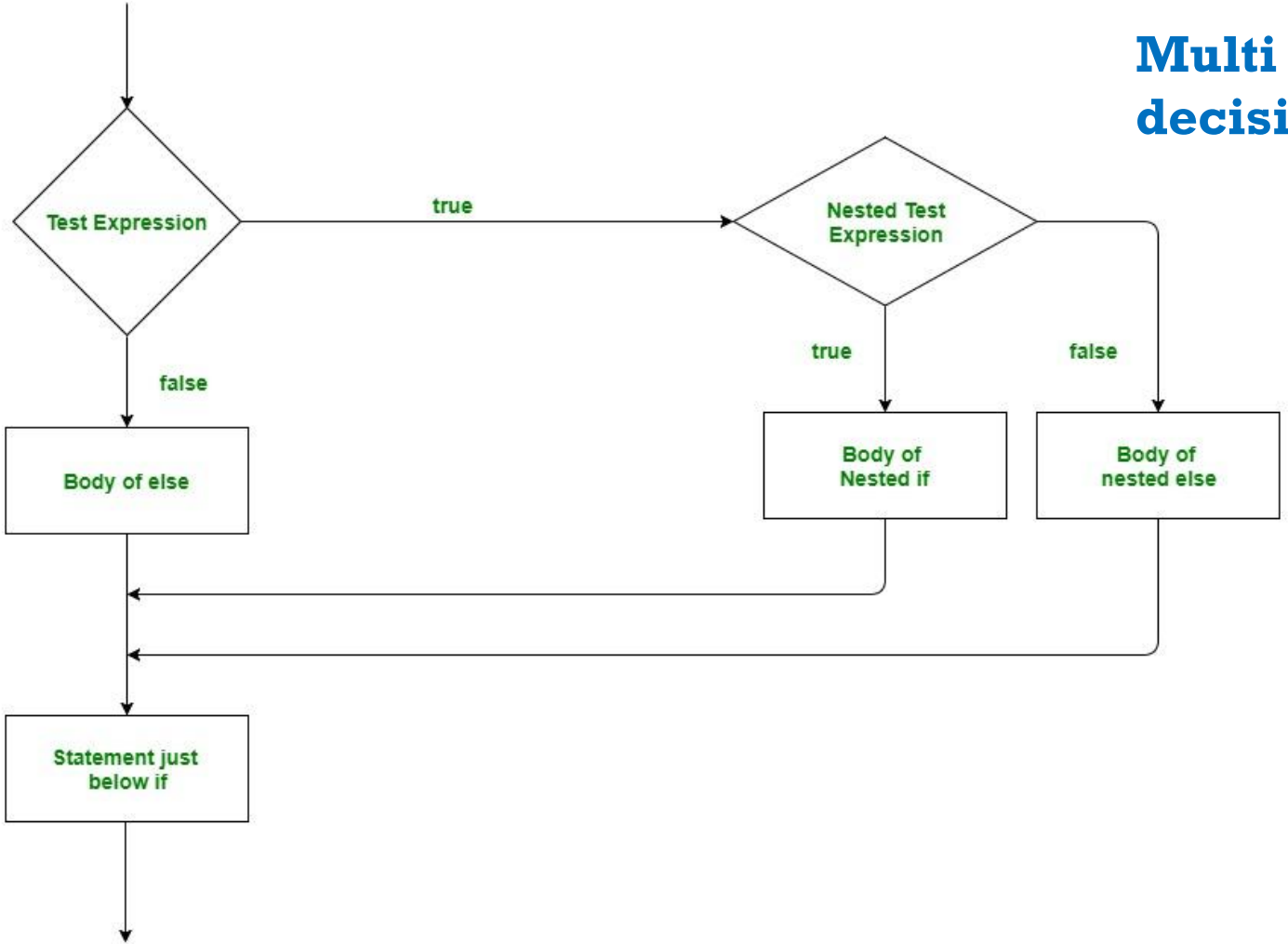
- if statement can also be checked inside other if statement.
- This conditional statement is called nested if statement.
- This means that inner if condition will be checked only if outer if condition is true and by this, we can see multiple conditions to be satisfied.

NESTED IF ELSE STATEMENT

- **Syntax:**

```
if test_expression:
    if test_expression:
     statements
    else:
     statements
else:
    Body of else
```

Multi way decision



EXAMPLE 1

```
# NESTED IF ELSE statement example
x = 20
y = 30
if x>=y:
    print("x is greater than y")
    if x==y:
        print("x is equal to y")
    else:
        print("x is greater than y")
else:
    print("x is smaller than y")
```

Output:

x is smaller than y

IF ELSE CHAIN STATEMENT

- **Syntax:**

```
if test_expression:
    statement 1
else:
    if test_expression:
        statement 2
    else:
        statements
```

EXAMPLE 1

```
# IF ELSE CHAIN statement example
letter = "A"
if letter == "B":
    print("letter is B")
else:
    if letter =="C":
        print("letter is C")
    else:
        if letter == "A":
            print("letter is A")
        else:
            print("letter is neither A,B or C")
```

Output:

letter is A

Example

You can also chain if..else statement with more than one condition.

```
# if..else chain statement
letter = "A"

if letter == "B":
  print("letter is B")

else:

  if letter == "C":
    print("letter is C")

  else:

    if letter == "A":
      print("letter is A")

    else:
      print("letter isn't A, B and C")
```

Output:

```
letter is A
```

IF-ELIF-ELSE STATEMENT

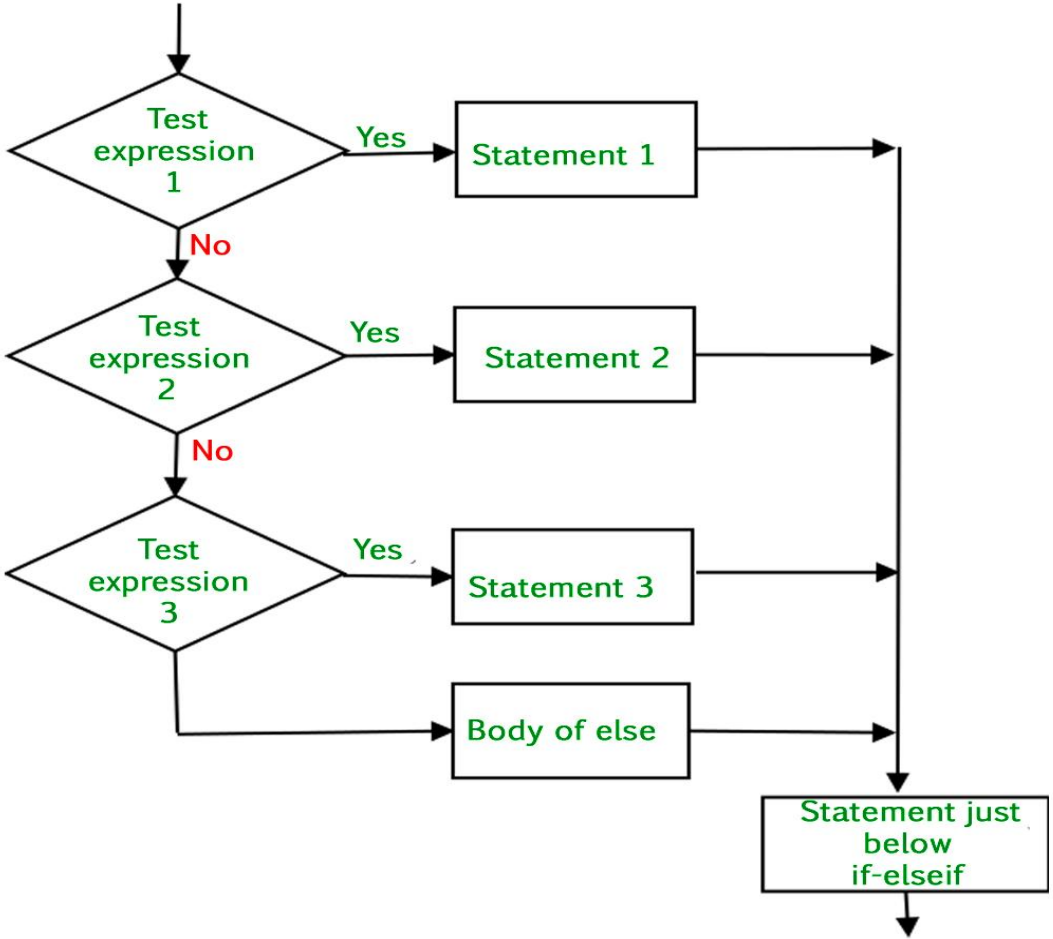
- The if-elif statement is shortcut of if..else chain.
- While using if-elif statement at the end else block is added which is performed if none of the above if-elif statement is true.

IF-ELIF-ELSE STATEMENT

Syntax:

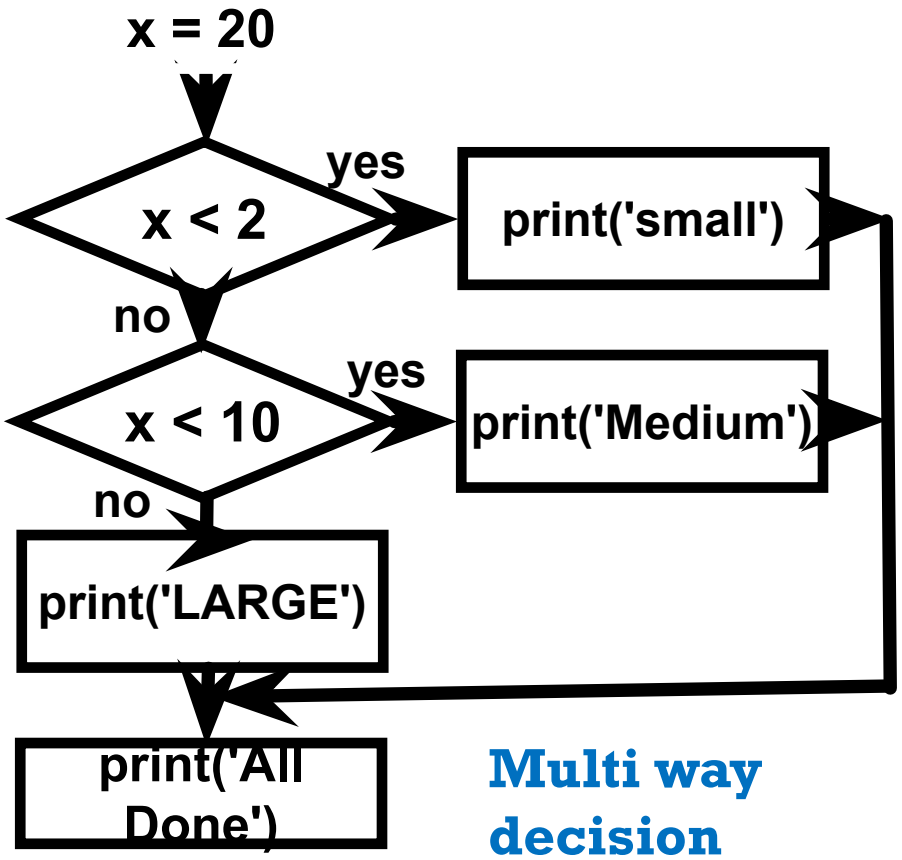
```
if test_expression:
    statement 1
elif test_expression :
    statement 2
elif test_expression :
    statement 3
    . .
    . .
else:
    body of else
```

IF-ELIF-ELSE STATEMENT



EXAMPLE 1

```
x = int(input("enter x:"))
if x < 2 :
    print('small')
elif x < 10 :
    print('Medium')
else :
    print('LARGE')
print('All done')
```



Output:
enter x:20
LARGE
All done

Multi way decision

EXAMPLE 2

```
# if elif else statement example
i = 25
if (i == 10):
    print("i is 10")
elif (i == 15):
    print("i is 15")
elif (i == 20):
    print("i is 20")
else:
    print("i is greater than 20")
```

Output:

i is greater than 20

EXAMPLE 3

```
# if elif else statement example
letter = "D"
if letter == "B":
    print("letter is B")
elif letter =="C":
    print("letter is C")
elif letter == "A":
    print("letter is A")
else:
    print("letter is neither A,B or C")
```

Output:

letter is neither A,B or C

MULTIPLE CONDITIONS

- Multiple conditions can be checked in a ‘if’ statement using logical operators ‘and’ and ‘or’.
- Python code to print ‘excellent’ if mark1 and mark2 is greater than or equal to 90, print ‘good’ if mark1 or mark2 is greater than or equal to 90, print ‘need to improve’ if both mark1 and mark2 are lesser than 90

EXAMPLE 1

```
# multiple condition example
mark1 = int(input("Enter mark1:"))
mark2 = int(input("Enter mark2:"))
if mark1>=90 and mark2>=90:
   print("excellent")
if mark1>=75 or mark2 >= 80:
   print("good")
else:
   print("needs to improve")
```

Output:

```
Enter mark1:20

Enter mark1:60
needs to improve
```

TASK

1. Write a program to print the largest of 3 numbers.
2. Write a program to check whether a person is eligible to vote or not. [The person should be above 18]
3. Write a program to find whether a year is a leap year.
4. Write a python program to segregate student based on their CGPA. The details are as follows:

<=9 CGPA <=10 - outstanding
<=8 CGPA <9 - excellent
<=7 CGPA <8 - good
<=6 CGPA <7 - average
<=5 CGPA <6 - better
CGPA<5 - poor

TASK

5. **Write a Python program to create a simple login system.**

   The program should:

   1. Define a predefined username and password.

   2. Prompt the user to enter their username and password.

   3. Check if the entered username and password match the predefined ones.

   4. Display a success message if the login credentials are correct.

   5. Display an error message if the login credentials are incorrect.

CALCULATE THE CURRENT GMT TIME

- import time
- current_time=time.time() #get current time
- tseconds=int(current_time)
- csecond=tseconds%60
- tminutes=tseconds//60
- cminute=tminutes%60
- thours=tminutes//60
- chour=thours%24
- #print("Current time is", chour,":",cminute,":",csecond,"GMT")
- print(f"Current time is {chour:02}:{cminute:02}:{csecond:02} GMT")

NEED OF ITERATIVE CONTROL

Repeated execution of set of statements

- An **iterative control statement** is a control statement providing repeated execution of a set of instructions

- Because of their repeated execution, iterative control structures are commonly referred to as “loops”.

ITERATIVE CONTROL STATEMENTS

- ***while*** statement (indefinite)
  - Repeatedly executes a set of statements based on a condition.
  - Ideal when stop criteria is not explicit

- ***for*** statement (definite)
  - Repeatedly executes a set of statements until the sequence is exhausted

WHILE

Syntax

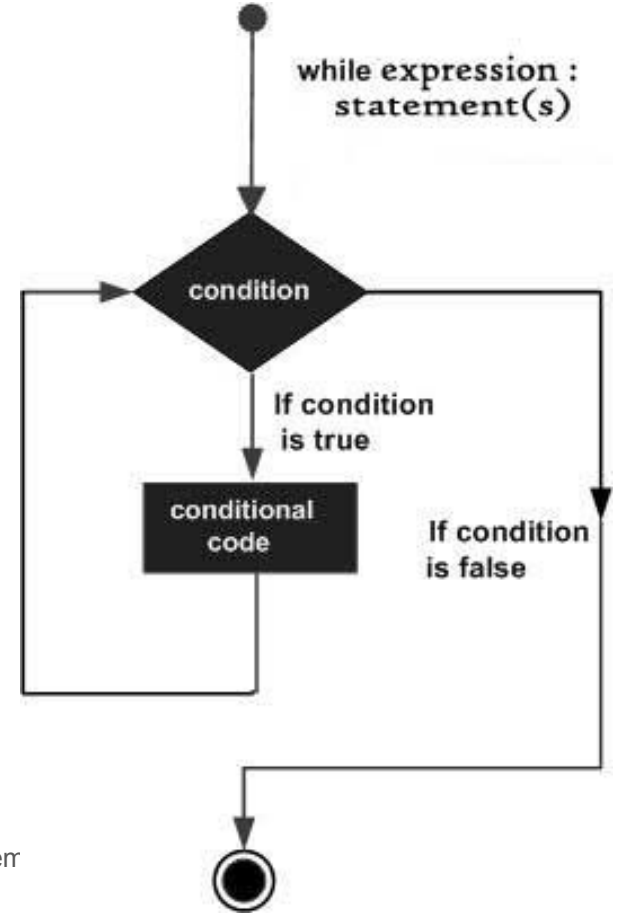
```
while loop-condition:          # Loop test
  statement(s)                 # Loop body
```

- while statement in Python, executes the statements within the while loop as long as the while condition is true.

WHILE LOOP

- Repeat a specific block of code
- Used to iterate over a block of code as long as the test expression (condition) is true
- While loop when we don't know the number of times to iterate beforehand

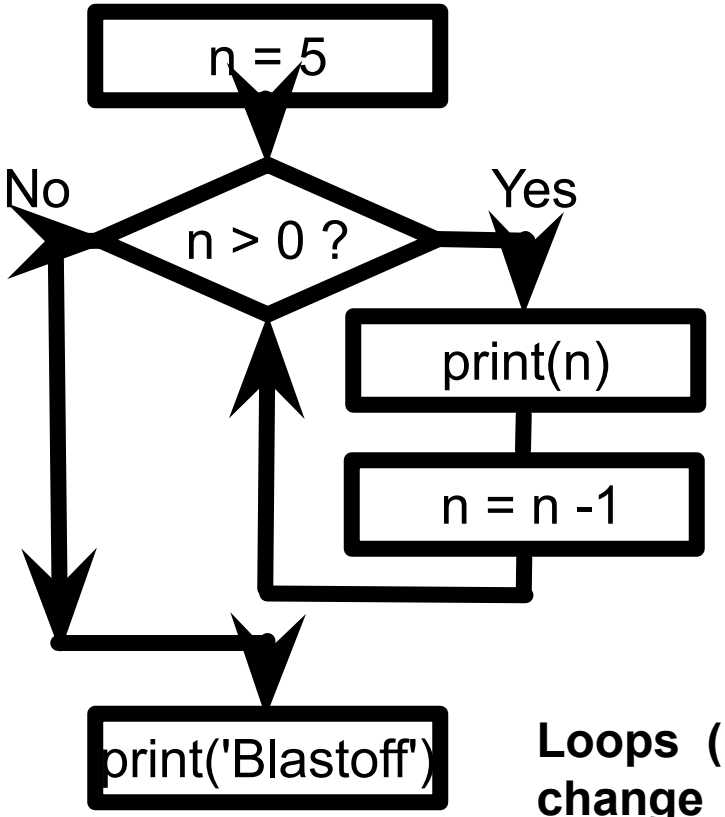
EXAMPLE 1: PRINT VALUES FROM 1 TO N




```
N=int(input())
a =1
while  a  <= N:
  print(a)
  a = a + 1
```

Iteration	a	a<=3	Print a	a=a+1
1	1	True	1	counter=1+1 ->(2)
2	2	True	2	counter=2+1 ->(3)
3	3	True	3	counter=3+1 ->(4)
4	4	False	loop termination	

EXAMPLE 2



Program:

```
n = 5
while n > 0 :
    print(n)
    n = n - 1
print('Blastoff!')
print(n)
```

Output:

```
5
4
3
2
1
Blastoff!
0
```

Loops (repeated steps) have iteration variables that change each time through a loop. Often these iteration variables go through a sequence of numbers.

EXAMPLE 3: PRINT VALUES FROM 90 TO 100 IN A LINE

```
a=90
b=100
while a < b:
        print(a, end=' ')
        a = a+1
```

Output:

90 91 92 93 94 95 96 97 98 99

EXAMPLE 4: IF-ELSE IN WHILE LO

Print even and odd numbers between 1 to the entered number

CODE

```
n = int(input("Enter a number:"))
while n>0:
    if n%2==0:
        print(n, "is even")
    else:
        print(n, "is odd")
    n = n-1
```

OUTPUT

```
Enter a number:5
5 is odd
4 is even
3 is odd
2 is even
1 is odd
```

EXAMPLE 3

- Write a program to reverse the number received from the user
- Input: 14786
- Output:68741

EXAMPLE 3-SOLUTION

```
num1 = int(input("Enter any number : "))
r=0
rnum=0
while(num1!=0):
    r = num1 % 10
    rnum = rnum * 10 + r
    num1 = num1//10
print("Reverse number is : ", rnum)
```

TASKS

- Print a two table
- Sum of first 10 even numbers
- Add the successive cubic roots for an input N….. Ex: 1+8+27=36.(N=3)
- Write a program to display all the numbers which are divisible by 13 but not by 3 between 100 and 500 (exclude 100 and 500)
- Check for palindrome

DEFINITE LOOP

- Quite often we have a list of items in a file - effectively known as finite set of things
- We can write a loop to run the loop once for each of the items in a set using the Python for construct
- These loops are called “definite loops” because they execute an exact number of times
- We say that “definite loops iterate through the members of a set”

FOR LOOP

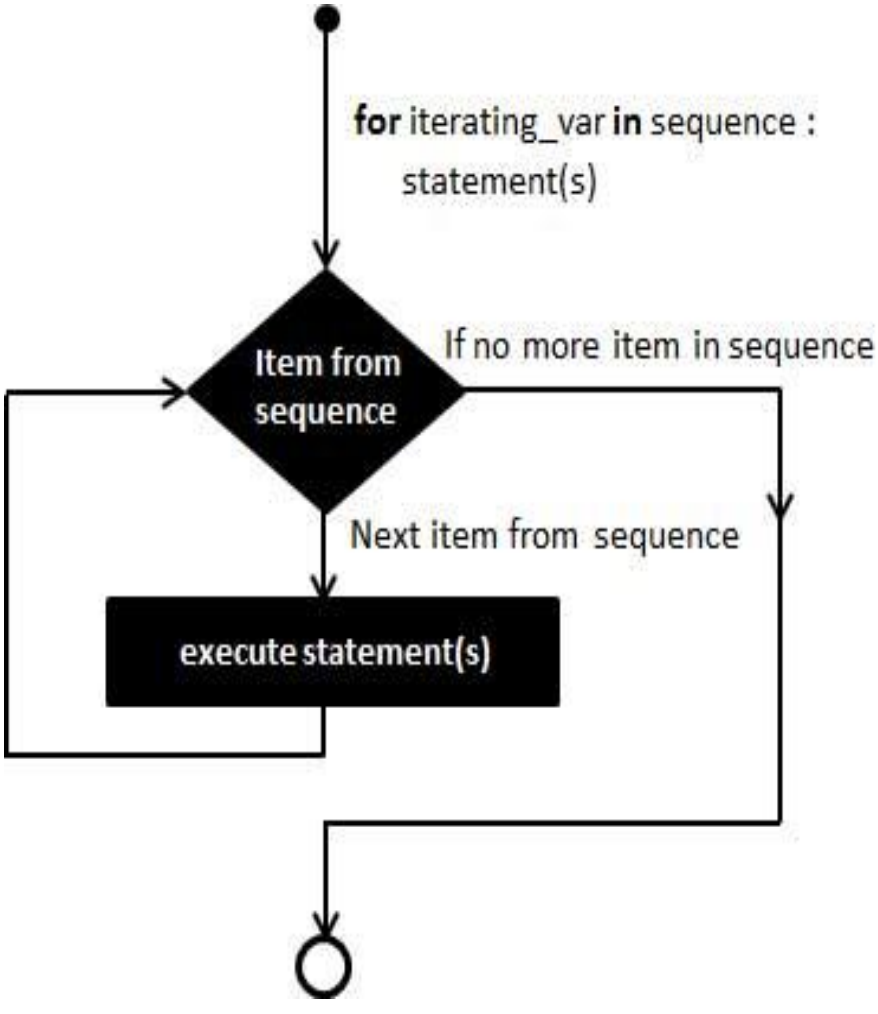
- Repeatedly executes a set of statements until the sequence is exhausted

- Python sequences
- String – a sequence of characters
- range
- List, Tuple

FOR LOOP

- A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).
- With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.
- Syntax:

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



EXAMPLE 1

A SIMPLE FOR LOOP

```
for i in [5, 4, 3, 2, 1] :
    print(i)
print('Blastoff!')
```

OUTPUT

5
4
3
2
1
Blastoff!

EXAMPLE 2

Calculate the square of each number of list

CODE

```
numbers = [1,2,3,4,5]
for i in numbers:
    square = i**2
    print("Square of ", i , "is", square)
```

Output:

```
Square of  1 is 1
Square of  2 is 4
Square of  3 is 9
Square of  4 is 16
Square of  5 is 25
```

EXAMPLE 3

For and Strings

CODE

```
for i in "python":
    print("The letter is", i)
```

OUTPUT

```
The letter is p
The letter is y
The letter is t
The letter is h
The letter is o
The letter is n
```

EXAMPLE 4

CODE

```
names = ['Ram', 'Raj', 'Rak']
for i in names:
    print("Name:", i)
```

OUTPUT

```
Name: Ram
Name: Raj
Name: Rak
```

FOR LOOP

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

FOR LOOP

```
print("String Iteration")
s = "Geeks"
for i in s:
    print(i)
```

FOR AND DICTIONARIES

```
data1 = {0:'hi',1:'a',2:1,3:5.5,4:200.1,5:'do'}
data2 = {'no',.9}
for x in data1.keys():
    print(data1[x])
```

FOR LOOP

- Example:1

```
for i in range(0, 10, 2):
    print(i)
```

  - Example:2

```
for i in range(1, 4):
    for j in range(1, 4):
        print(i, j)
```

  - Example 3:

```
Numbers =[x for x in range(11)]
print(Numbers)
```

- Example 4:

```
data1 = {0:'hi',1:'a',2:1,3:5.5,4:200.1,5:'do'}
data2 = {'no',.9}
for x in data1.keys():
    print(data1[x])
```

- Example 5:

```
cars = ['audi', 'bmw', 'subaru', 'toyota']
for car in cars:
   if car == 'bmw':
      print(car.upper())
   else:
      print(car.title())
```

RANGE

- Syntax - range(begin,end,step) where
- Begin - first value in the range; if omitted, then default value is 0
- end - one past the last value in the range; end value may not be omitted
- Step - amount to increment or decrement; if this parameter is omitted, it defaults to 1 and counts up by ones
- begin, end, and step must all be integer values;
- floating-point values and other types are not allowed

File Edit Format Run Options Window Help

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

File Edit Shell Debug Options Win

```
>>>
================ RESTART:
0
1
2
3
4
>>>
```

File Edit Format Run Options Window Help

```
for num in [0, 1, 2, 3, 4]:
    print(num)
```

File Edit Shell Debug Options Wi

```
================ RESTART:
0
1
2
3
4
>>>
```

class_for_range2.py - C:/Python/Python37/class_for_range2.py (3.7.9)

File Edit Format Run Options Window Help

```
for x in range(5):
    print('Hello world!')
```

File Edit Shell Debug Opt

```
>>>
================ RE
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
>>>
```

IF-ELSE IN FOR LOOP

Example: Print all even and odd numbers

```
for i in range(1, 11):
    if i % 2 == 0:
        print('Even Number:', i)
    else:
        print('Odd Number:', i)
```

Output

```
Odd Number: 1
Even Number: 2
Odd Number: 3
Even Number: 4
Odd Number: 5
Even Number: 6
Odd Number: 7
Even Number: 8
Odd Number: 9
Even Number: 10
```

ADDITIONAL EXAMPLE

- mystr= “Hello World”
- For i in enumerate(mystr):
  - print(i)

  - Note: list(enumerate(mystr1))

USING ELSE CONDITIONAL STATEMENT WITH FOR LOOP

- for i in range(1, 4):
- print(i)
- else: # Executed because no break-in for
- print("No Break")

Note: The else block just after for/while is executed only when the loop is NOT terminated by a break statement.

```
for i in range(1, 4):
  print(i)
  break
else:  # Not Executed as there is a break
print("No Break")
```

NESTED LOOPS

```
x = [1, 2]
y = [4, 5]
i = 0
while i < len(x) :
  j = 0
  while j < len(y) :
    print(x[i] , y[j])
    j = j + 1
  i = i + 1
```

```
x = [1, 2]
y = [4, 5]

for i in x:
  for j in y:
    print(i, j)
```

PRINTING MULTIPLICATION TABLE USING NESTED FOR LOOPS

- Print 2 and 3 tables in the range of 1 to 11

PRINTING MULTIPLICATION TABLE USING NESTED FOR LOOPS

```
for i in range(2,4):
  for j in range (1,12):
    print(i,"*",j,"=",i*j)
  print()
```

TASK

Checkpoint

5.8 Rewrite the following code so it calls the `range` function instead of using the list `[0, 1, 2, 3, 4, 5]`.

```
for x in [0, 1, 2, 3, 4, 5]:
    print('I love to program!')
```

5.9 What will the following code display?

```
for number in range(6):
    print(number)
```

5.10 What will the following code display?

```
for number in range(2, 6):
    print(number)
```

5.11 What will the following code display?

```
for number in range(0, 501, 100):
    print(number)
```

5.12 What will the following code display?

```
for number in range(10, 5, -1):
    print(number)
```

File Edit Format Run Options Window Help

```python
"""for x in  range(5):
    print('I love to program!')
for x in  range(1,6,1):
    print('I love to program!')
for number in range(6):
    print(number)
for number in range(2, 6):
    print(number)
for number in range(0, 501, 100):
    print(number)"""
for number in range(10, 5, -1):
    print(number)
```

File Edit Shell Debug Options Window Help

```
>>>
===================== RESTART: C:/Python/Pythor
I love to program!
I love to program!
I love to program!
I love to program!
I love to program!
>>>
===================== RESTART: C:/Python/Pythor
0
1
2
3
4
5
>>>
===================== RESTART: C:/Python/Pythor
2
3
4
5
>>>
===================== RESTART: C:/Python/Pythor
0
100
200
300
400
500
>>>
===================== RESTART: C:/Python/Pythor
10
9
8
7
6
>>>
```

LOOP CONTROL STATEMENT

- Loop control statements change the execution of the loop from its normal sequence.
- Python follows
  - Break statement: used to exit the loop
  - Continue statement: used to skip once, i.e one iteration
  - pass statement: to execute once

```
fruits=["apple","mango","banana","cherry","orange"]
for items in fruits:
  if items=="banana":
    break
  else:
    print(items)
```

```
fruits=["apple","mango","banana","cherry","orange"]
for items in fruits:
  if items=="banana":
    continue
  else:
    print(items)
```

```
fruits=["apple","mango","banana","cherry","orange"]
for items in fruits:
  if items=="banana":
    pass
  else:
    print(items)
```

```
for num in range(0,10):
    if num == 5:
        continue
    print(f'Iteration: {num}')
```

```
for num in range(0,10):
    if num == 5:
        pass
    print(f'Iteration: {num}')
```

BREAK STATEMENT

- **The break statement is used inside the loop to exit out of the loop**.
- In Python, when a break statement is encountered inside a loop, the loop is immediately terminated, and the program control transfer to the next statement following the loop.
- **For example**, you are searching a specific email inside a file. You started reading a file line by line using a loop. When you find an email, you can stop the loop using the break statement.
- We can use Python break statements in both for loop and while loop.
- It reduces execution time.

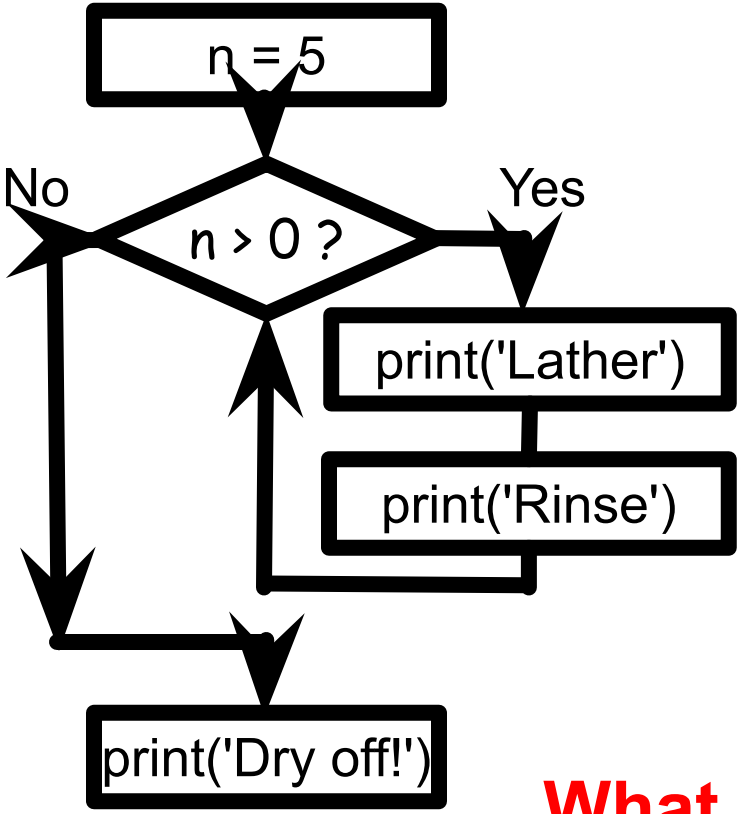
CONTINUE STATEMENT

- Continue statement forces to execute the next iteration of the loop while skipping the rest of the code inside the loop for the current iteration only.
- It can be used in both while and for loops.

PASS STATEMENT

- **Pass** is used to execute nothing; it means when we don't want to execute code, the pass can be used to execute empty
- It just makes the control pass by without executing any code.
- If we want to bypass any code, a pass statement can be used.
- The difference between the comments and pass is that comments are entirely ignored by the Python interpreter, whereas the pass statement is not ignored

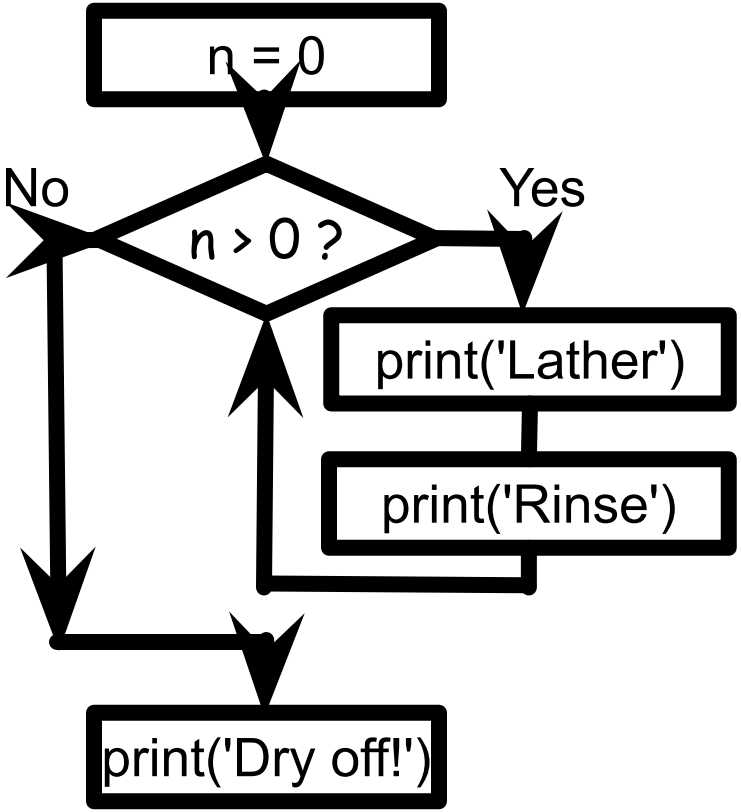
EXAMPLE 4: AN INFINITE LOOP




```
n = 5
while n > 0 :
print('Lather')
  print('Rinse')
print('Dry off!')
```

What is wrong with this loop?

EXAMPLE 5




```
n = 0
while n > 0 :
    print('Lather')
    print('Rinse')
print('Dry off!')
```

What is this loop doing?

SUMMARY

- ***break***
  - Jumps out of the closest enclosing loop
- ***continue***
  - Jumps to the top of the closest enclosing loop
- ***pass***
  - Does nothing at all: it’s an empty statement placeholder

EXAMPLE

- List of months: January, February, March, April, May, June, July, August, September, October, November, December
- Input the name of Month: April
- No. of days: 30 days

```
# Display a list of months to the user

print("List of months: January, February, March, April, May, June, July, August, September, October,
November, December")

# Request input from the user to enter the name of a month and assign it to the variable 'month_name’

month_name = input("Input the name of Month: ")

# Check the input 'month_name' and provide the number of days based on the entered month

if month_name == "February":
    print("No. of days: 28/29 days")  # Display the number of days in February (28 or 29 days for leap
years)
elif month_name in ("April", "June", "September", "November"):
    print("No. of days: 30 days")  # Display the number of days for months having 30 days
elif month_name in ("January", "March", "May", "July", "August", "October", "December"):
    print("No. of days: 31 days")  # Display the number of days for months having 31 days
else:
    print("Wrong month name")  # If the entered month name doesn't match any of the above
conditions, display an error message
```

FIBONACCI SERIES

```
n = 10
num1 = 0
num2 = 1
next_number = num2
count = 1

while count <= n:
    print(next_number, end=" ")
    count += 1
    num1, num2 = num2, next_number
    next_number = num1 + num2
print()
```

PYTHON PROGRAM TO FIND THE FACTORIAL OF A GIVEN NUMBER.

```
# given number
given_number= 5
  # since 1 is a factor of all number set the factorial to 1
factorial = 1
 # iterate till the given number
for i in range(1, given_number + 1):
    factorial = factorial * i
 print("The factorial of ", given_number, " is ", factorial)
```

Write a Python code to check if the given mobile number is valid or not. The conditions to be satisfied for a mobile number are:

a) Number of characters must be 10

b) All characters must be digits and must not begin with a ‘0’

Validity of Mobile Number

Input	Processing	Output
A string representing a mobile number	Take character by character and check if it valid	Print valid or invalid

Test Case 1

- abc8967891
- Invalid
- Alphabets are not allowed

Test Case

- 440446845
- Invalid
- Only 9 digits

Test Case 3

- 0440446845

- Invalid
- Should not begin with a zero

Test Case 4

- 8440446845

- Valid
- All conditions statisfied

```
# Prompt the user to input a mobile number
mobile_number = input("Enter your mobile number: ")

# Initialize a flag to check validity
is_valid = True

# Check if the length is exactly 10
if len(mobile_number) != 10:
    is_valid = False

# Check if the first character is '0'
if mobile_number[0] == '0':
    is_valid = False
```

```
# Check if all characters are digits
for char in mobile_number:
    if char < '0' or char > '9':  # This checks if the character is
not a digit
        is_valid = False
        break

# Output whether the mobile number is valid or not
if is_valid:
    print("The mobile number is valid.")
else:
    print("The mobile number is not valid.")
```

While loop-guess the output

```
x=5
while(x<15):
  print(x**2)
  x+=3
----------------------------------------
a=7
b=5
while(a<9):
  print(a+b)
  a+=1
```

While loop-guess the output

```
b=5
while(b<9):
  print("H")
  b+=1
-----------------------------
i=0
while i<3:
  print(i)
  i=i+1
  print(0)
```

While loop-guess the output

```
i=100
while i<57:
  print(i)
  i+=5
-----------------------
b=15
while(b>9):
  print("Hello")
  b=b-2
```

While loop-guess the output

```
x=15
while(x==15):
  print("Hello")
  x=x-3
-------------------
i=9
while True:
  if i%3==0:
    break
  print("A")
```

While loop-guess the output

```
a=6
while(a<=10):
  print("a")
  a+=1
----------------------
i=0
while i<3:
  print(i)
  i=i+1
else:
  print(7)
```

While loop-guess the output

```
c = -9
while c < 20:
  c += 3
  print(c)
---------------
a=5
while a>0:
   print(a)
   a=a-1
```

Convert the following loop into for loop :

```
x = 4
while(x<=8):
    print(x*10)
    x+=2
```