UNIT:3

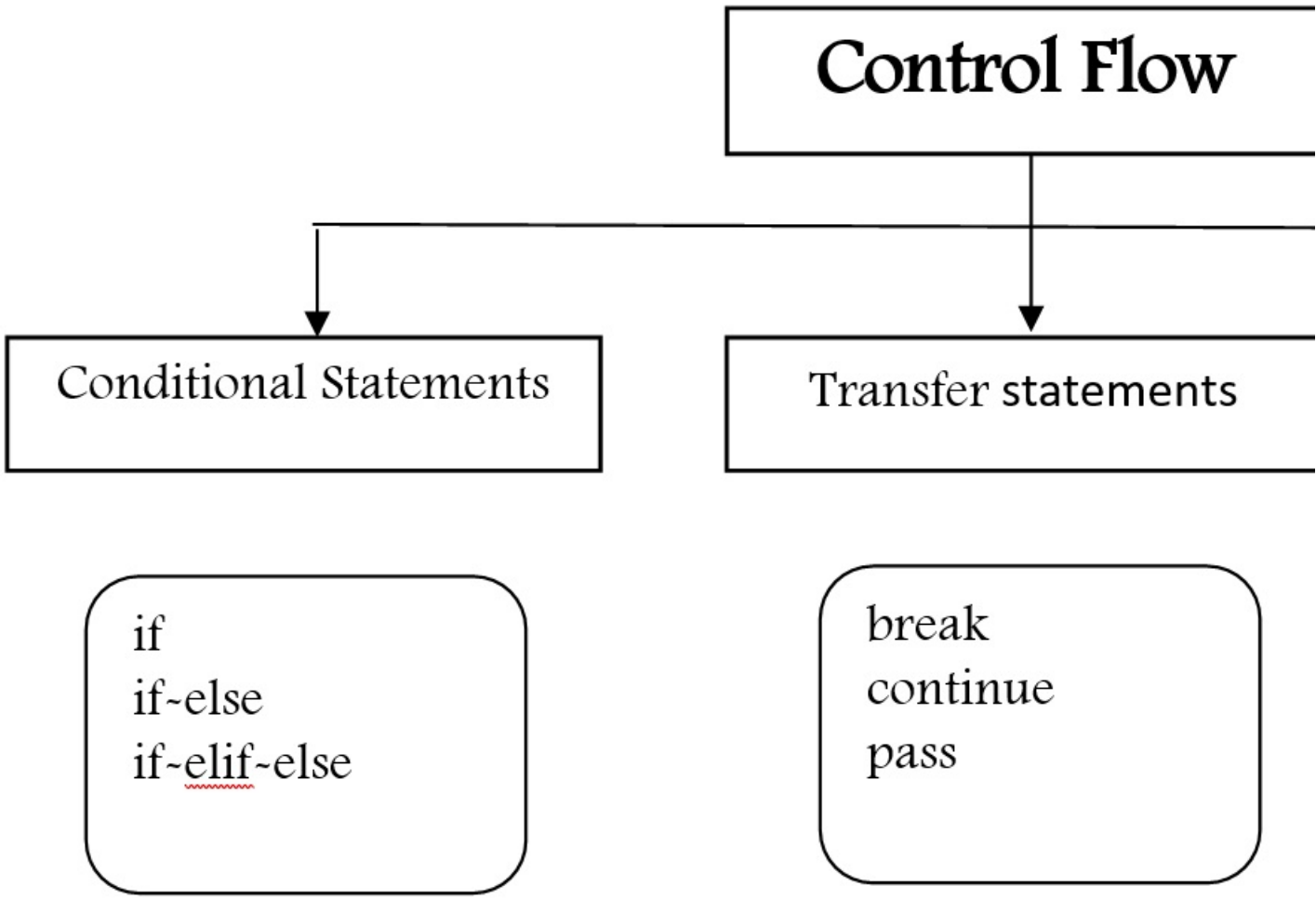
Python statements and Loops:

if, if-else, While, for loops, break, continue, pass, Python Function;

Files I/O.

Functions: Definition, call, positional and keyword parameter. Default parameters, variable number of arguments. Modules - import mechanisms. Functional programming - map, filter, reduce, max, min. lambda function - list comprehension.

Flow control describes the order in which statements will be executed at runtime.



1. **Conditional Statements**
2. **if**

if condition: statement (or)

if condition:

statement-1

statement-2

statement-3

If condition is true then statements will be executed.

E.g.

name=input ("Enter Name:")

if name== “Shivani"

print ("Hello Shivani Good Morning")

print ("How are you!!!")

2) **if-else**:

if condition:

Action-1

else:

Action-2

If condition is true then Action-1 will be executed otherwise Action-2 will be executed.

E.g.

name=input ("Enter Name:")

if name== “Shivani":

print ("Hello Shivani Good Morning")

else:

print ("Hello Guest Good Moring")

3) **if-elif-else:**

Syntax:

if condition1:

Action-1

elif condition2:

Action-2

elif condition3:

Action-3

elif condition4:

Action-4

…….

else: Default Action

Based on condition the corresponding action will be executed.

1. **Iterative Statements**

If we want to execute a group of statements multiple times then we should go for Iterative statements.

Python supports 2 types of iterative statements.

1. for loop

2. while loop

1) **for loop:** If we want to execute some action for every element present in some sequence (it may be string or collection) then we should go for ‘for’ loop.

Syntax:

for x in sequence:

body

where sequence can be string or any collection. Body will be executed for every element present in the sequence.

E.g. To print characters, present in the given string

string = ‘name’

for i in string:

print(i)

for loop with range () function:

range () – The range () function in python generates a sequence of numbers within a specified range.

syntax – range (start, stop, step)

start: optional. The starting value of the sequence (default is 0).

stop: Required. The ending value of the sequence (exclusive).

step: optional. The increment between each number in the sequence (default is1).

E.g. for i in range (10):

print("hello")

E.g. To display odd numbers from 0 to 20 1)

for x in range (21):

if (x%2! = 0):

print(x)

1. **while loop:** If we want to execute a group of statements iteratively until some condition false, then we should go for while loop.

Syntax:

while condition:

body

E.g.: To print numbers from 1 to 10 by using while loop.

x=1

while x <=10

print(x)

x=x+1

E.g.: To display the sum of first n numbers

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

sum=0

i=1

while i<=n:

sum=sum + i

i=i+1

print ("The sum of first", n, "numbers is:", sum)

Nested Loops: Sometimes we can take a loop inside another loop, which are also known as nested loops.

1. **Transfer Statements**
2. **break:** We can use break statement inside loops to break loop execution based on some condition.

E.g.:

cart = [10,20,600,60,70]

for item in cart:

if item>500:

print ("To place this order insurance must be required")

break

```
print(item)
```

1. **continue:** We can use continue statement to skip current iteration and continue next iteration.

E.g. To print odd numbers in the range 0 to 9.

```
for i in range (10):
if i%2==0:
continue
print(i)
```

1. **pass:** pass is a keyword in Python. In our programming syntactically if block is required which won't do anything then we can define that empty block with pass keyword.

E.g.

```
for i in range (100):
if i%9==0:
print(i)
else: pass
```