Unit 2

Decision control statement :-

Control statement that determines the control flow of set of instruction i.e decides the sequence of instruction to be executed three fundamental flow in programming language are sequential sel ↑, selection & iterative control

Sequential control statement :- python program executed sequentially from first line to last line i.e second statement is executed after first third after second so on & so forth this is called sequencial control statement

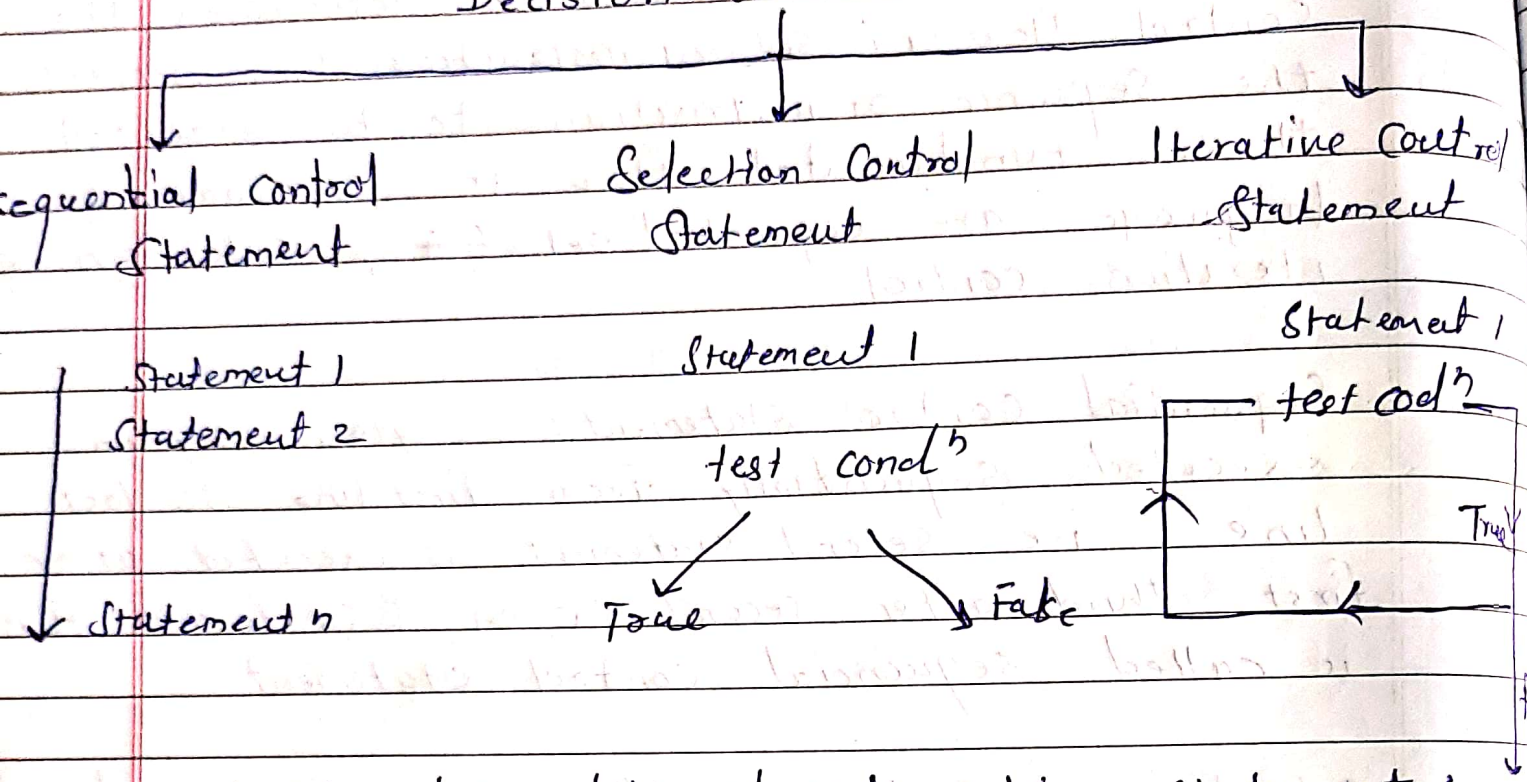
Selection control statement :- It is executed only celected set of statement repeatedly

iterative Contool statement :- Set of statement repeatedy executed is called iterative Control Statement

decision control statement allow the programer to build program that determine which statements of the code should be executed which should be ignored

Fig :- Show categorization of decision control statement

Decision control statements



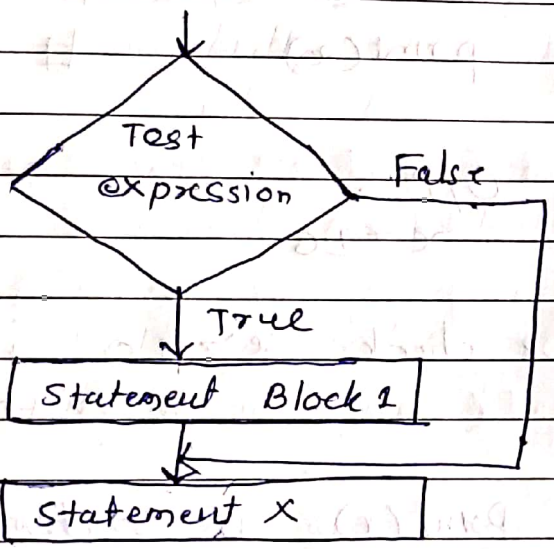
Selection / conditional branching statement :-
Conditional branching statement jumps from one part to another depending on particular condition is satisfied or not.

* python support different type of conditional branching statement
  1. if statement
  2. if else statement
  3. Nested if statement
  4. if-elif-else statement

1. if statement :- it is freqently used for decision making if statement based on value of given Boolean expression.

Syntax of if Statement

```
if test_expression :
    Statement 1
    ...
    statement n
Statement x
```



[properly indent the statement depending on previous statement]

if statement include statement 1 to Statement n inclosed within if block
first test expression evaluated
if true then Statement 1 to N executed
Otherwise (false) these statement 1 to N skiped directly statement x is exicuted

header in python keyword ~~if~~ followed by colon
" if test_expression: "

header

↑ for header

group of statement is called suit

After header all instn are intended at same level called suite

header & suit are together known as clause

Ex -

```
x = 10                  # initialize x
if (x>0):               # test value x
    x = x+1             # increment x
print(x)                # print value x
```

O/P

x = 10

here we check x > 10 if true then

x = x+1 i.e x = 10+1

= 11

Print(x) if statement is false

python uses indentation to block of code where C & C++ used curly brackes

Ex -

```
age = int (input (" Enter the age: "));
if ( age >= 18):
    print (" your are eligible to vote")
```

O/P

Enter the age : 35

your are eligible to vote

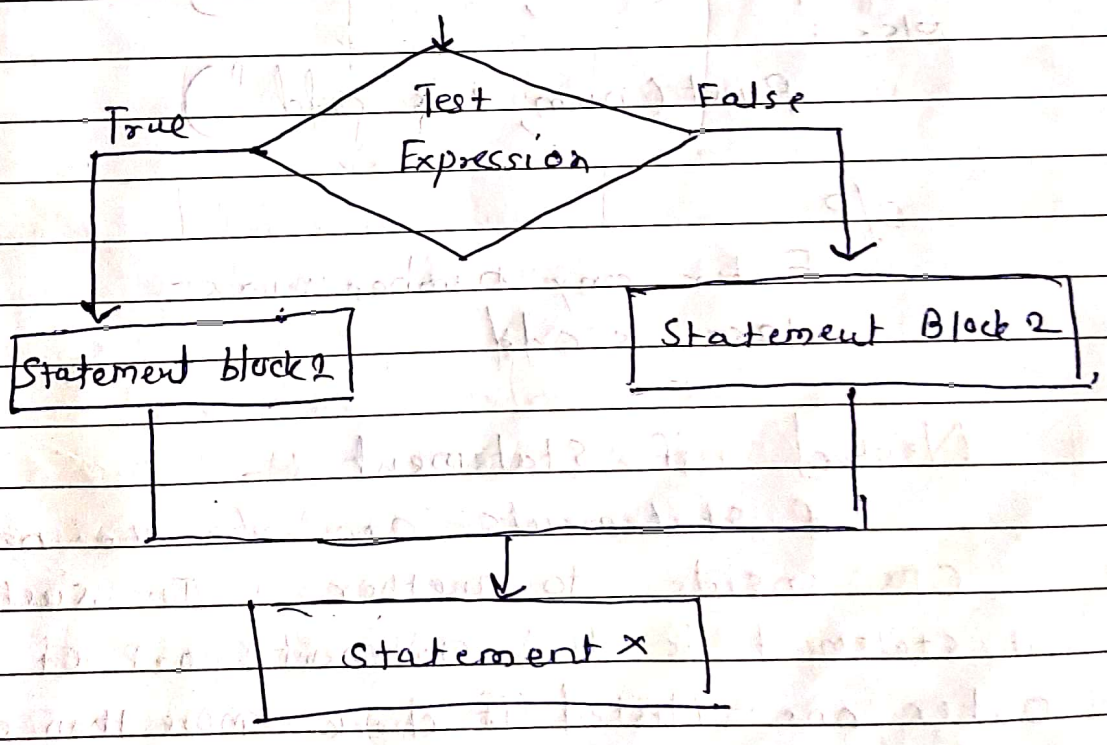
if - else statement :

The test expression is evaluated if statement is true

else is excuted when expression is false

Syntax of if else statement

```
if (test expression):
        Statement block 2
else :
        Statement block 2
Statement x
```



if the expression is true
statement block 1 executed & block2 skip
Otherwise if the expression false
statement block 2 executed & block 1 skip
then control will pass statement X in
every case

Write a program to find even or odd no

```
num = int (input ("Enter any number: "))
if (num % 2 == 0):
    print(num, " is even")
else:
    Print(num, " is odd")
```

O/P

Enter any number: 125
125 is odd

Nested if statement :-

if statements can be nested
One inside to another In such case
if statement is the statement part of the
outer one. Nested if check more than one cond^n is
satisfied [i.e multiway selection]

```
num = int(input("Enter no from 0-30"))
if (num>=0 and num<10):
    print("It is range 0-10")
if (num>=10 and num<20):
```

```
    print("It is in range 10-20")
if (num >= 20 and num < 30):
    print("It is in range 20-30")
```

O/P

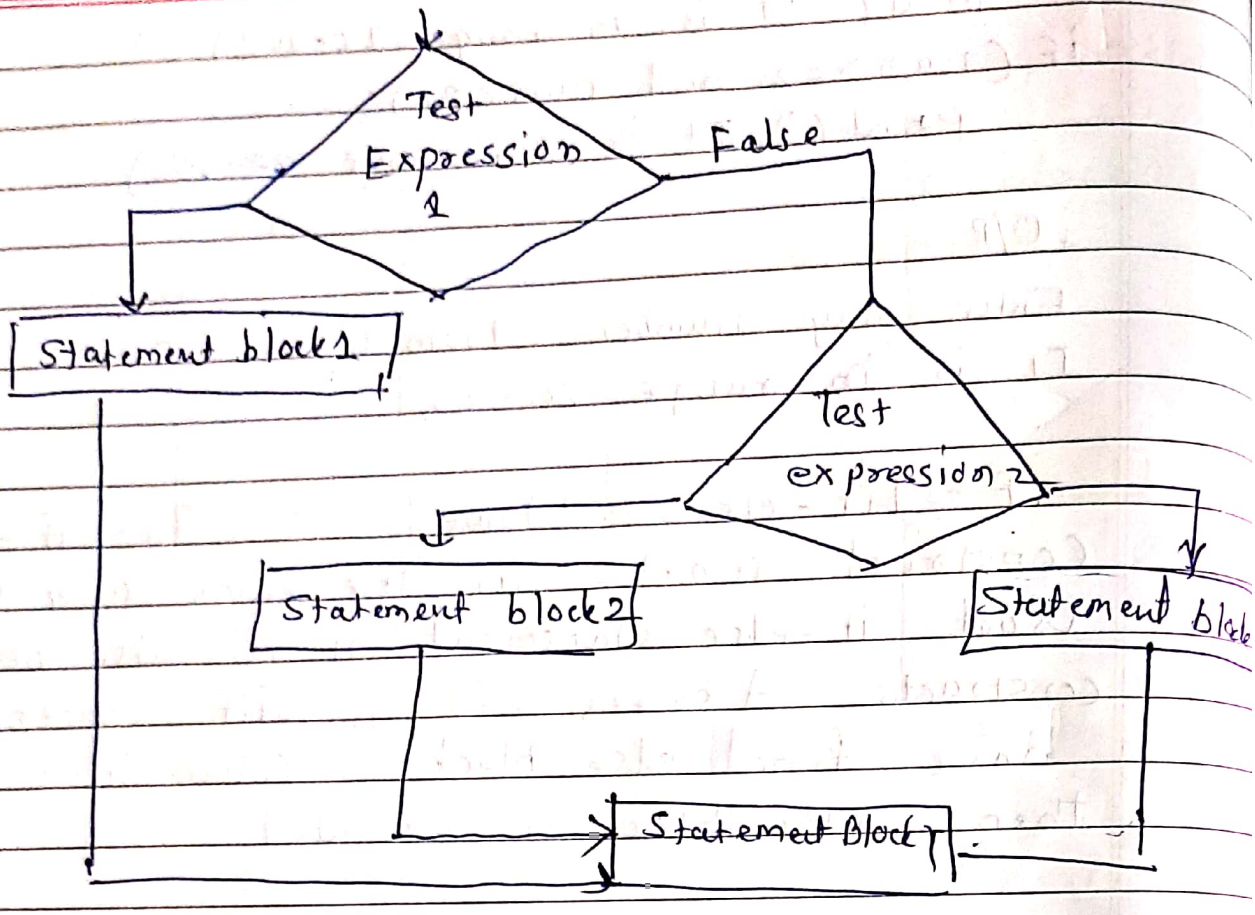
Enter any number from 0-30 : 25
It is in range 20-30

if-elif-else statement :- The if-elif-else Construct works in the same way as usual if-else statement it is also nested if constructs A series of if-elif statements can have final else block When if-elif is false then else block is executed.

Syntax of if-elif-else statement

```
if ( test expression 1)
    statement block 1
elif ( test expression 2)
    statement block 2
elif ( test expression N)
    statement block N
else
    statement block x
Statement
```

flowchart



Ex —

```
num = int(input("Enter ther no"));
if (num == 0):
    print("The Value is equal to zero")
elif (num > 0):
    print("no is positive")
else:
    print("The no is negative")
```

O/p

Enter any number : -10
The number is negative

The program whether no is positive or negative if ~~it~~ the first expression is true then rest statement is ignored & ~~rest~~ next to check true same way

Basic loop structures / iterative statements

Iterative statements are decision control statement that used to repeat execution of list of statement Python support two type of iterative statements ① while loop ② for loop

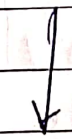
While loop :-

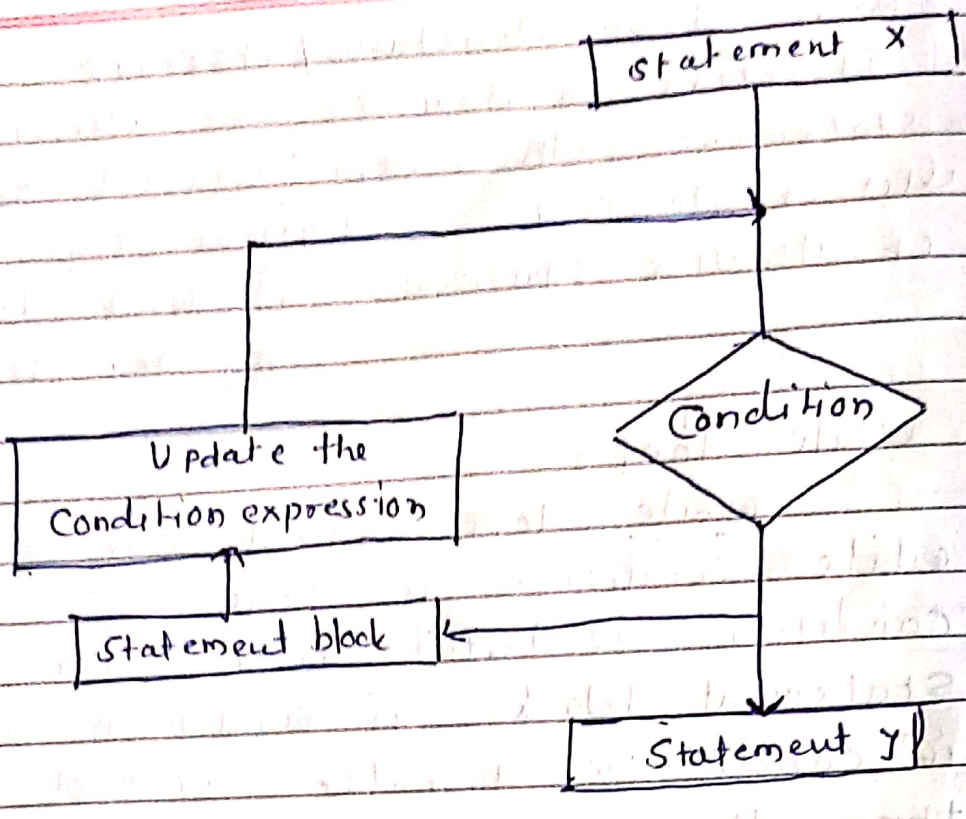
while loop repeat one or more statement while condition is true In while loop condition is tested first then exicuted the statement block if condition is true. if condition is false control jump statement that is the immediate outside the while loop block

Syntax of while loop

```
Statement x
while (condition) :
    statement block
statement y
```

flow chart





Ex

```
i = 0
while (i<=20):
    print(i, end=" ")
    i = i+2
```

[end=" " → space betn 2 value]

O/P

0 1 2 3 4 5 6 7 8 9 10

for printing values same line use end with a separator (you can specify any separator like tab (\t), space, comma etc with end

```
i = 0
while (i<=10):
    print(i, end='\t')
    i+=2
```

O/P

0 1 2 3 4 5 6 7 8 9 10

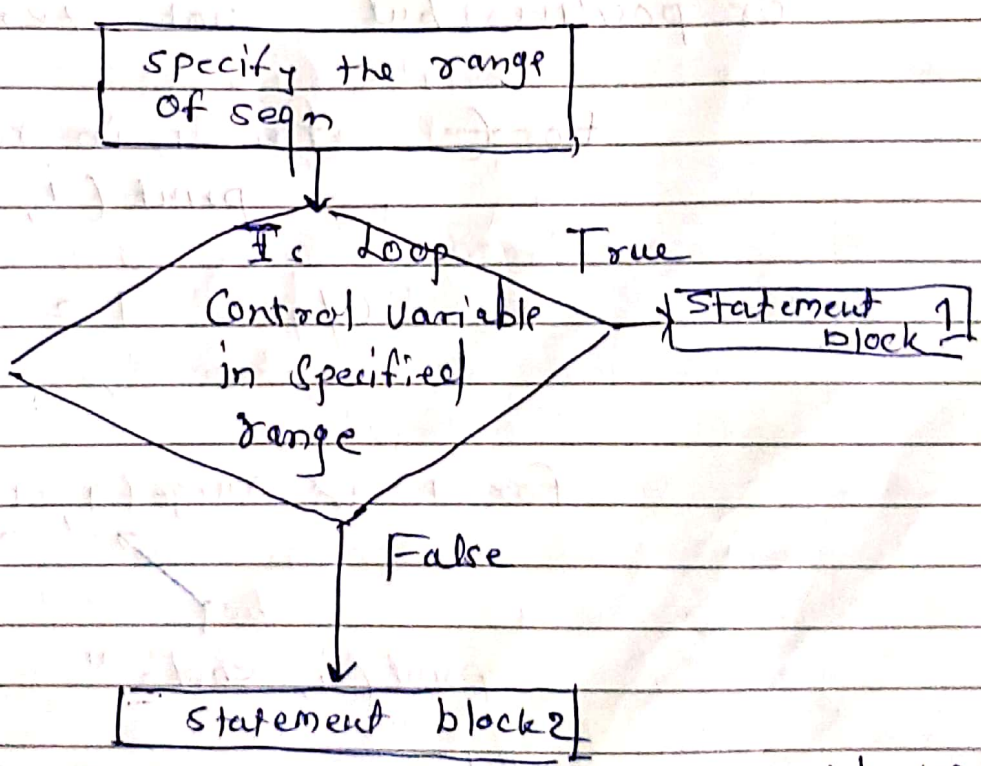
for loop

for loop Provide mechanism to repeat task until particular cond^n is true python syntax uses the range function which makes loop simpler

for ... in statement in-Python use in python flow of statement given in fig [no of time execution determined mathematically by logic of for loop]

syntax for loop

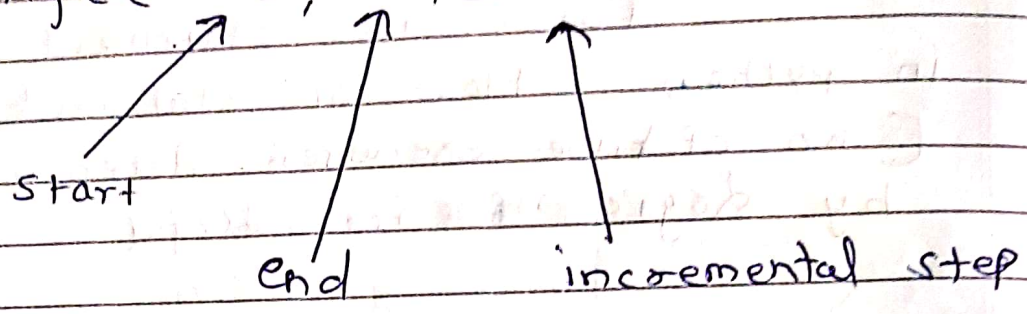
```
for loop_control_var in sequence:
    Statement block
```



for loop executed each item in sequence identify control veriable Assign value in range else ~~statem~~ statement block of loop is executed ~~when~~ after for loop body

The range() function:-
The range() is build in function iterating over a sequence of numbers

range(beg, end, [step])



here
beg is starting number
ending by ending number
Step argument is optional by default no is incremented by one it can be negative or positive but not zero

~~for (~~ for i in range(1, 5):
print(i, end=" ")

O/p 1 2 3 4

for i in range(1, 10, 2)
step
beg
end
print(i, end=" ")

O/p 1 3 5 7 9

```
for i in range(10)
    print(i, end=' ')
```

O/P

0 1 2 3 4 5 6 7 8 9

Selecting appropriate loop

Loop can be different type such as entry - controlled, exit controlled counter controlled condition controlled loops

pre-test and past-test loop

while is entry control loop cond^n is tested befor the loop start

exit control loop cond^n tested test condition after loop is executed

if requirement is pre-test we chose loop for and while loop

Cond^n control & Counter control loop

When we know in advanced no of time loop should be executed we use Counter control loop. counter is variable that must be initalize tested updated in loop opration counter control loop in which counter is intalize as constant value is know as definite repeatation loop

when we do not know no of time loop will executed it is called Conditiony control loop Special Value is called sentinel value

When counter control loop is use we chose for loop

In counter control loop we use while loop

counter- control loop

```
i = 0
while (i <= 10):
print (i, end = " ")
i += 1
```

```
i = 1
while (i>0):
print (i, end = " ")
i += 1
if (i == 10):
    break
```

Nested Loops

Python allow Nested loops i.e loop that can placed within other loops this will work with any loop like while loop as well as for loop it is most commonly used with For loop

Write a program for following Pattern

```
*
* *
* * *
* * * *
* * * * *
```

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

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

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

The break statement

The break statement used terminate the execution of nearest enclosing loop

It is used with for loop & while loop when compiler read break statement controller terminat the loop go to next to loop

Ex ——

Note:- break statement terminate and transfers execution to the statement immediately following the loop

```
while ....
  ...
  if cond^n :
      break
  ...
....
```

Transfers control out of the while loop

```
for ....
  ...
  if condition :
      break
  ...
  ...
```

Transfer control out of the for loop

```
for ....
  ...
  for ...
    ...
    if condition
        break
    ...
  ...
```

Transfer control out of inner for loop

Ex —

```
i=1
while i<=20:
    print(i, end=" ")
    if i==5:
        break
    i = i+1
print("\n Done")
```

O/P

1 2 3 4 5 —

The continue Statement

Like break the continue statement can only appear in body of a loop when compiler incounter continue statement then rest of statement is skiped and again continue of loop for perticular cond^{n} in Loop

```
for i in range (1, 11):
    if (i == 5):
        continue
    print (i, end = " ")
print ("\n Done")
```

O/P

1 2 3 4 6 7 8 9 10
Done

```
while (...)
...
if cond$^{n}$:
    continue
...
...
```

Transfer control cond^{n} expression while loop

```
for (...)
.....
if condition:
    continue
.....
.....
```

Transfer control cond^{n} exp^{n} for loop

```
for (...)
...
for (...)
.....
if cond$^{n}$:
    continue
.....
```

Transfer control cond^{n} exp^{n} to inner for loop

[Continue STOP current iteration of loop and go to next One]

The **Pass statement**

Nothing happed when pass statement is executed It use for when statement is required syntatically

```
for letter in "HELLO":
    pass # The statement is doing nothing

    print ("pass:", letter)
print ("Done")
```

```
pass : H
pass : E
pass : L
pass : L
pass : O

Done
```

pass statement is written as we cannot have empty body in loop

The else Statement used with loop

python you can have else statement associated with loop statement

if the else statement is used with For loop else is exicuted when the loop has completed

when used with while Loop else is executed when condition become false

Ex
```
for letter in "HELLO":
    print(letter, end=" ")
else:
    print("\n Done")
```

O/p

H E L L O

Done

Ex
```
i = 1
while (i<0):
    print(i)
    i = i-1
else:
    print(i," is not negative so loop did
          not executed ")
```

O/p

1 is not negative so loop did not executed

other data types

Tuples: tuple is similar to list values are separed by commas inclosed within parenthesi difference betn clist & tuple that you can change the values in list but not tuple, to edit the edit data in tuple then error will be generated

```
Tup = ('a', 'bc', 78, 1.23)
Tup = ('d', 78)
print (Tup)
print (Tup[0]) # print first element
print(Tup[1:3]) # print element 2nd till 3rd
print(Tup[2:]) # print starting 3rd
print(Tup*2) # repeat tuple
print(Tup + Tup2) # Concatenate 2 tuples
```

O/P

```
('a', 'bc', 78, 1.23)
a
('bc', 78)
(78, 1.23)
('a', 'bc', 78, 1.23, 'a', 'bc', 78, 1.23)
('a', 'bc', 78, 1.23, 'd', 78)
```

List: list is consist item by commas and inclose within squar bracket ([])
list is value belonging at different types
value stored list using indexes index of first element is 0 to n-1 for n total number value in list

```
list = ['a', 'bc', 78, 1.23]
list2 = ['d', 78]

print (list)
print (list[0]) # print first element is list
print (list[1:3]) # starting 2nd till 3rd
print (list[2:]) # starting 3rd ele
print (list * 2) # repeate list
print (list + list2) # combination list
```

o/p

same (only square bracket)

[,]

a

[. . .]

[ ]

[ ]

Dictionary :- python dictionaries store data key value pairs key is string and value is any data type key value pairs enclose in curly braces ({ }) & separed by colon (:) to access ~~braces~~ value in dictionary you just specify key in square braces ([ ])

```
Dict = { "Item" : "chocolate", "price" : 100 }
print ( Dict [ "Item" ] )
print ( Dict [ "price" ] )
```

O/p

chocolate

100