* Comparison Operator.

* Chained Comparison Operator.

* Python Statements

1) if, elif and else statements.

- if case 1:   # syntax
        perform action 1
  elif case 2:
        perform action 2.
  else:
        perform action 3.

  hence else behaves as a default.

2) For loop.

- for item in object: # syntax
        statements to do stuff.

3) While loop

- while test: # syntax
        code statement
  else:
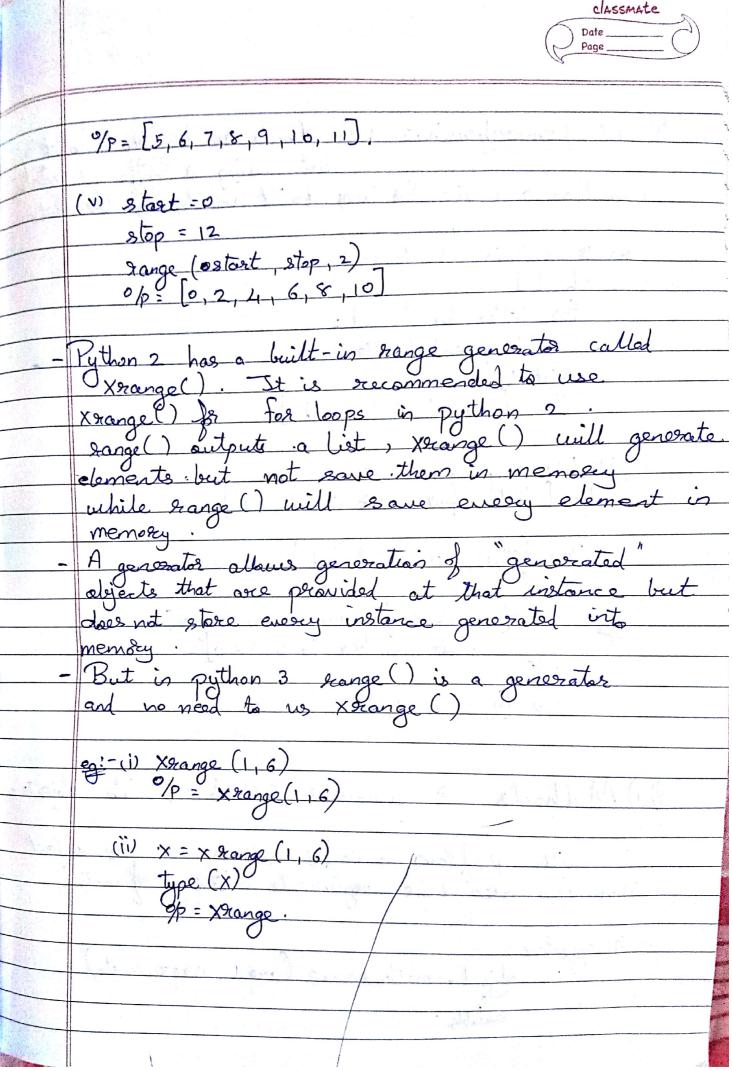   final code statement.

- Break - breaks out of the current closest enclosing loop.

Continue - goes to the top of the closest enclosing loop.

Pass - does nothing at all.

- while test:
    code statement
        if test:
            break.
        if test:
            continue
  else:

1) range() ⟹ allows you to create a list of nos. ranging from the starting point upto the ending point and can also specify the step size

(i) range (0, 10)
    o/p= [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

(ii) range (5)
    o/p: [0, 1, 2, 3, 4].

(iii) x = range (0, 10)
    type (x)
    o/p= list.

(iv) start = 5
    stop = 12
    range (start, stop)

$o/p = [5, 6, 7, 8, 9, 10, 11]$.

(v) start $= 0$
     stop $= 12$
     range (start, stop, 2)
     $o/p = [0, 2, 4, 6, 8, 10]$

- Python 2 has a built-in range generator called xrange(). It is recommended to use xrange() for for loops in python 2. range() outputs a list, xrange() will generate elements but not save them in memory while range() will save every element in memory.

- A generator allows generation of "generated" objects that are provided at that instance but does not store every instance generated into memory.

- But in python 3 range() is a generator and no need to us xrange()

eg:- (i) xrange (1, 6)
         $o/p = xrange(1, 6)$

     (ii) x = x range (1, 6)
           type (x)
           $o/p = xrange$.

5) **List Comprehensions.** ⟹ allows us to build out lists using a different notation such as → 1 line for loop built inside brackets.

eg:- # traditional for loop.
```
·L = []
for letter in `word`:
    L.append (letter)
print L.
%p = .[`w', `o', `r', `d']
```

```
# list comprehension .
L = [ letter for letter in `word`]
L
%p = [`w', `o', `r', `d']
```

eg:- `lst = [number for number in range (11) if number % 2 == 0]`
```
lst
%p = [0, 2, 4, 6, 8, 10]
```