The range() Function

```
range(10)
-----
list(range(10))
-----
range(4,10)
-----
list(range(4,10))
-----
#range(begin,end, step)
list(range(4,50,5))
```

The built-in function range() is the right function to iterate over a sequence of numbers. It generates an iterator of arithmetic progressions:

The "for" loop

```
primes = [2, 3, 5, 7]
for prime in primes:
    print(prime)
-------------------------------------
```

For loops can iterate over a sequence of numbers using the "range" and "xrange" functions. The difference between range and xrange is that the range function returns a new list with numbers of that specified range, whereas xrange returns an iterator, which is more efficient. (Python 3 uses the range function, which acts like xrange). Note that the range function is zero based.

loop1.py

```
# Prints out the numbers 0,1,2,3,4
for x in range(5):
    print(x)
# Prints out 3,4,5
for x in range(3, 6):
    print(x)
# Prints out 3,5,7
for x in range(3, 8, 2):
    print(x)
```

"while" loops

```
# Prints out 0,1,2,3,4
count = 0
while count < 5:
    print(count)
    count += 1 # This is the same as count = count
+ 1
```

While loops repeat as long as a certain boolean condition is met.

"break" and "continue" statements

```
# Prints out 0,1,2,3,4
count = 0
while True:
    print(count)
    count += 1
    if count >= 5:
        break
# Prints out only odd numbers - 1,3,5,7,9
for x in range(10):
    # Check if x is even
    if x % 2 == 0:
        continue
    print(x)
```

break is used to exit a for loop or a while loop, whereas continue is used to skip the current block, and return to the "for" or "while" statement.

can we use "else" clause for loops

```
# Prints out 0,1,2,3,4 and then it prints "count
value reached 5"
count=0
while(count<5):
    print(count)
    count +=1
else:
    print("count value reached %d" %(count))
# Prints out 1,2,3,4
for i in range(1, 10):
    if(i%5==0):
        break
    print(i)
else:
    print("this is not printed because for loop is
terminated because of break but not due to fail in
condition")
```

We can use else for loops. When the loop condition of "for" or "while" statement fails then code part in "else" is executed. If break statement is executed inside for loop then the "else" part is skipped. Note that "else" part is executed even if there is a continue statement.