10.1 Ranges:

Kotlin range is defined as an interval from start value to the end value.

Range expressions are created with operator (..) which is complemented by in and !in.

The value which is equal or greater than start value and smaller or equal to end value comes inside the defined range.

https://www.javatpoint.com/kotlin-integer-range

```
val aToZ = 'a'..'z'
val oneToNine = 1..9
```

Example:

```
val aToZ = 'a'..'z'
val oneToNine = 1..9
for(i in aToZ)
     println(i)
```

```
:Output:
a
b
c
d
e
f
:
:
z
```

What happened when we try to iterate a r range in decreasing order using .. operator ? This will print nothing.

```
for (a in 5..1){
     print(a)// print nothing
}

for (a in 5 downTo 1){
   print(a)// 54321
}
```

until range

The until() function or until keyword in range is used to exclude the last element. It iterates range from start to 1 less than end.

```
for (a in 1 until 5){
   print(a)// 12345
}
```

Other examples of Range

```
for (x in 1.rangeTo(5))
            print(x)
println()
/*-------------------------*/
for (x in 5.downTo(1))
            print(x)
println()
```

Kotlin range of characters

```
fun main(args: Array<String>) {
     for(x in 'a'..'e')
          print("$x ")
     println()
     for (x in 'e' downTo 'a')
          print("$x ")
}
```

```
:Output:
a b c d e

e d c b a
```

https://www.javatpoint.com/kotlin-integer-range

Kotlin range step

```
fun main(args: Array<String>) {
     for (x in 1..10 step 2)
          print("$x ")
     println()
     for (x in 10 downTo 1 step 3)
          print("$x ")
}
```

```
:Output:
1 3 5 7 9

10 7 4 1
```

Kotlin range iterator

An **iterator()** method is also be used to iterate the range value.
It uses ***hasNext()*** method which checks the next element in the range and ***next()*** method returns the next element of the range.

```
fun main(args: Array<String>) {
     val chars = ('a'..'e')
     val it = chars.iterator()
         while (it.hasNext()) {
               val x = it.next()
               print("$x ")
         }
}
```

Working of ranges

- Ranges implement ClosedRange<**T**> a common interface in the library.
- It represents a closed mathematical interval defined for comparable types.
- It contains two endpoints as ***start*** and ***end*** (endInclusive)points.
- The operation performed in range is to check whether the element is contained in it or not.
- This is done by using in or !in operators.
- An arithmetic progression is represented by integral type progressions such as CharProgression, IntProgression, Long Progression.
- Progressions represent the first element, the last element and the step which is non-zero.
- The first element is first, sub-sequent elements represent previous element plus step and the last element is the last element unless progression is completed.

- Progression refers to subtype of Iterable<N>, where N is Char, Int or Long.

- As progression is Iterable<N> type it can be used in for-loop and function such as filter, map etc.

- The .. operator creates an object for integral type which implements both ClosedRange<T>and Progression.

- For example, a range type LongRange implements ClosedRange<Int> and extends Long Progression, it means all the operation which are defined for LongProgression is also available for LongRange.

- The output generated by ***downTo()*** and ***step()*** functions is always a Progression.

- The last element of the Progression is largest value not greater than the end value for positive step. The minimum value of progression is not less than the end value for negative step.
- The last value is checked by using (last-first) %step == 0.

Kotlin Utility Functions

Kotlin range utility functions have several standard library functions which are used in Kotlin ranges.

These utility functions are as follow:

- rangeTo()
- downTo()
- reversed()
- step()

https://www.javatpoint.com/kotlin-utility-functions