INTRODUCTION: PART - 4

For Loop

Control Flow: For Loop

The **for** loop in Python is used to iterate over a sequence or other iterable objects.

Iterating over a sequence is called *traversal*.

for syntax:

```
for element in sequence:          # don't forget to add colon here
    statement1                    # code
    statement2
    ...
    statement=n
```

Here, element is the variable that takes the value of the item inside the sequence on each iteration.

Loop continues until we reach the last item in the sequence.

Fig: operation of for loop

In [1]:
```
# Example: for loop

lst = [10,20,'A','B','iota',100]   # list

for element in lst:
    print(element)                 # element is taking the value of each item in the list
```

```
10
20
A
B
iota
100
```

In [2]:
```
string = "I am learning Python"

print(len(string))          # use len function to find length (number of items) in an iterable.

for i in string:
    print(i)
    print("loop running")

print("Loop Completed")   # this is outside for loop
```

```
20
I
loop running

loop running
a
loop running
m
loop running

loop running
l
loop running
e
loop running
a
loop running
r
loop running
n
loop running
i
loop running
n
loop running
g
loop running

loop running
P
loop running
y
loop running
t
loop running
h
loop running
o
loop running
n
loop running
Loop Completed
```

range() function

We can generate a sequence of numbers using range() function.

- Doesn't store all values in the memory, it would be inefficient. So it remembers the start stop and step size and generates the next number on the go.
- *range(10)*: it creates an object which has numbers from 0 to 9. To print that we need to pass them in a list or iterate with for loop.

range(stop) -> range object
range(start, stop[, step]) -> range object

> - Return an object that produces a sequence of integers from start (inclusive)
> - to stop (exclusive) by step. range(i, j) produces i, i+1, i+2, ..., j-1.
> - start defaults to 0, and stop is omitted! range(4) produces 0, 1, 2, 3.
> - These are exactly the valid indices for a list of 4 elements.
> - When step is given, it specifies the increment (or decrement).

In [3]:
```
# Example
a = range(10)
print(a)

print(list(a))
```

```
range(0, 10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [4]:
```
# Example

for i in range(1,5):
    print(i)
    print(f"Square of {i} is :{i**2}")

print("Loop has ended")
```

```
1
Square of 1 is 1
2
Square of 2 is 4
3
Square of 3 is 9
4
Square of 4 is 16
Loop has ended
```

Control Flow: Break & Continue statements

In Python, break and continue can alter the flow of a normal loop.

> Loops iterate over a block of code until criteria is False, but sometimes we wish to terminate the current iteration or even the whole loop without checking the criteria/condition.

The break and continue statements are used in these cases.

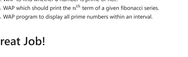
break statement

syntax:

```
for var in sequence:
    #codes inside for loop
    if condition:
        break
    #codes inside for loop

#codes outside for loop


while criteria:
    #codes inside while loop
    if condition:
        break
    #codes inside while loop
#codes outside while loop
```




```
for var in sequence:
    # codes inside for loop
    if  condition:
        break
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if  condition:
        break
    # codes inside while loop

# codes outside while loop
```

In [5]:
```
# Example

numbers = range(1,5)

for num in numbers:
    if num==3:
        break
    print(num)

print("Outside for loop")
```

```
1
2
Outside for loop
```

continue statement

syntax continue:

```
for var in sequence:
    #codes inside for loop
    if condition:
        continue
    #codes inside for loop

#codes outside for loop


while criteria:
    #codes inside while loop
    if condition:
        continue
    #codes inside while loop
#codes outside while loop
```



```
for var in sequence:
    # codes inside for loop
    if  condition:
        continue
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if  condition:
        continue
        # codes  inside while loop

    # codes outside while loop
```

In [6]:
```
# Example: print odd numbers only from the list

list2 = [1,2,3,4,5,6]

for num in list2:
    if num%2==0:          # checking for even numbers
        continue          # if num is even continue will send it for next iteration without executing below-
    print(num)            # lines of code
```

```
1
3
5
```

Exercise: 5

1. WAP (Write a program) to print the even numbers between 1 to 100.
2. WAP to calculate factorial of any given number.
3. WAP to find whether a number is prime or not?
4. WAP which should print the n^{th} term of a given fibonacci series.
5. WAP program to display all prime numbers within an interval.

Great Job!