## Python – For Loop

- A `for` loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

- With the `for` loop we can execute a set of statements, once for each item in a list, tuple, set etc.

## Looping Through a String

Even strings are iterable objects, they contain a sequence of characters:

**Example**

Loop through the letters in the word "banana":

```
for x in "banana":
  print(x)
```

**Output:**

```
>>> for x in "banana":
...     print(x)
...
b
a
n
a
n
a
>>>
```

**Example**

Print each fruit in a fruit list:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

**Output:**

```
apple
banana
cherry
```

## The continue Statement

With the `continue` statement we can stop the current iteration of the loop, and continue with the next:

Example

Do not print banana:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

**Output:**

```
apple
cherry
```

## The break Statement

With the `break` statement we can stop the loop before it has looped through all the items:

**Example**

Exit the loop when `x` is "banana":

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

**Output:**

```
apple
banana
```

## The range() Function

To loop through a set of code a specified number of times, we can use the `range()` function,

The `range()` function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

**Example**

Using the range() function:

```
for x in range(6):
  print(x)
```

**Output:**

```
0
1
2
3
4
5
```

Note that `range(6)` is not the values of 0 to 6, but the values 0 to 5.

The `range()` function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: `range(2, 6)`, which means values from 2 to 6 (but not including 6):

**Example**

Using the start parameter:

```
for x in range(2, 6):
  print(x)
```

**Output**

```
2
3
4
5
```

## Step Value for range function

- The `range()` function defaults to increment the sequence by 1.
- However it is possible to specify the increment value by adding a third parameter: `range(2, 30, 3)`:

Example

Increment the sequence with 3 (default is 1):

```
for x in range(2, 30, 3):
  print(x)
```

**Output**

```
2
5
8
11
14
17
20
23
26
29
```

## Else in For Loop

## Nested Loops

A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop":

Example

Print each adjective for every fruit:

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

**Output:**

```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

## The pass Statement

`for` loops cannot be empty, but if you for some reason have a `for` loop with no content, put in the `pass` statement to avoid getting an error.

Example

```
for x in [0, 1, 2]:
  pass
```

**Output:**

## Activities using

- for loop, nested for loops (including break, continue, range(), else)

1. Write python programs to print the following series.
   (a) 0, 1, 2, 3, 4, 5, ….
   (b) 3, 5, 7, 9, 11, …
   (c) 2, 4, 8, 16, …
2. Write a program to print Fibonacci series.
3. Write python programs to print the following patterns.
   a. A
      B B
      C C C
      D D D D
      …

b. A
   A B
   A B C
   A B C D
   ....

c. A
   2 2
   C C C
   4 4 4 4
   E E E E E
   ......

4. **To find the factorial of a given number ‘n’ using while loop.**

   **<u>Sample Input/output:</u>**

   Input n: 5
   Output: 5 x 4 x 3 x 2 x 1 = 120

   Input n =3
   Output: 3 x 2 x 1 = 6