

# for loops

(Iterative code)

1

This video will discuss compound statements and for loops in Python.

# Loops

- A **loop** is a set of statements that repeats until some condition is met.
- Loops repeat until...
  - all items in a list or dictionary have been run through the loop (for loop)
  - a certain condition is met (while loop)
- Loops are compound statements...

2

**Loops** allow certain lines of a script to be repeated until a specified condition is met.

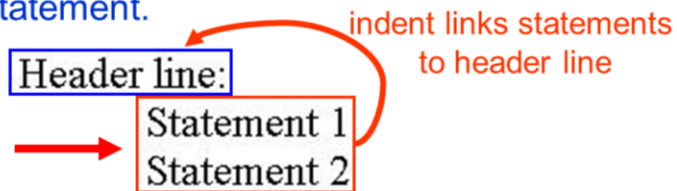
A **for loop** will repeat until it has processed all items in a Python data collection – these data collections include lists and dictionaries as well as arrays and sets (which will be discussed later in the course).

A **while loop** will repeat until a specific condition is met.

Loops are **compound statements**.

## Compound statements

- Statements that consist of multiple lines of code are **compound statements**.
- The 1<sup>st</sup> line is the **header line** which ends with a colon.
- The lines following the header line are indented to indicate they are associated with the header line.
- Consistent indentation is very important within a compound statement.



3

Compound statements contain multiple lines of code that are linked to a single operation.

Compound statements begin with a header line – the header line ends with a colon.

The lines below the header line are indented to indicate their association with the header line.

All lines associated with the header line must be indented by the same amount.

## for loops

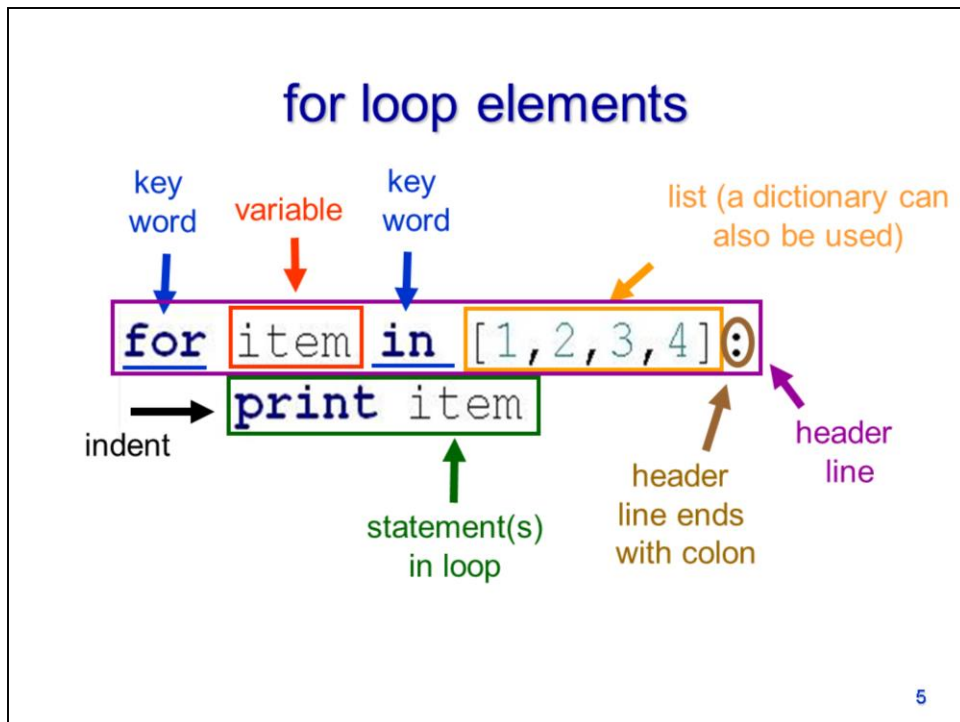
- for loops work with either a list or a dictionary.
- The loop runs one time for each item in the list / dictionary
- In each iteration...
  - an item from the list / dictionary is assigned to a variable.
  - the indented statements (which are part of the loop) run.

4

A for loop requires a list, dictionary, or other Python data collection as its input.

The for loop will run one time for each item in the data collection.

In each iteration, an item from the data collection is assigned to a variable and the indented lines below the for loop header are run.



Here is an example of a for loop.

The loop starts with a header line

Which ends in a colon.

The header line begins with **for**

Followed by a variable.


The variable is followed by **in**

Which is followed by the data collection (typically a list or dictionary).

Below the header will be one or more indented lines that are inside the for loop.

## for loop: example

```
for item in [1, 2, 3, 4]:  
    print item
```



Value of item	Output
item = 1	1
item = 2	2
item = 3	3
item = 4	4

6

Let's look at an example of a for loop in operation.

In the 1<sup>st</sup> iteration of the loop, the integer 1 is assigned to the variable *item*. The print statement prints the value 1.

In the 2<sup>nd</sup> iteration of the loop, the integer 2 is assigned to the variable *item*. The print statement prints the value 2.

The process repeats for the 3<sup>rd</sup> and 4<sup>th</sup> iterations until all items in the list have been processed.

## Creating sequential lists – the range function

- `range` creates a sequential list that contains as many numbers as specified...

```
range(5) → [0, 1, 2, 3, 4]
```

- The starting value can be set...

```
range(5, 10) → [5, 6, 7, 8, 9]
```

- The increment can be set...

```
range(0, 10, 2) → [0, 2, 4, 6, 8]
```

```
range(5, 0, -1) → [5, 4, 3, 2, 1]
```

7

Sequential lists can often be useful when using for loops. The **range** function can be used to easily create sequential lists.

For the range function, you can specify a single integer value to create a list in which the sequence starts with zero and stopping at the specified value (note that the ending value is not included in the result).

To start the list with a non-zero value, specify both a starting and ending value for the list.

To create a sequence with an increment other than 1, specify a third integer for the increment amount.

Note that sequential lists can count down from the start value to the end value if you specify a negative increment.

## Using sequential lists with for loops

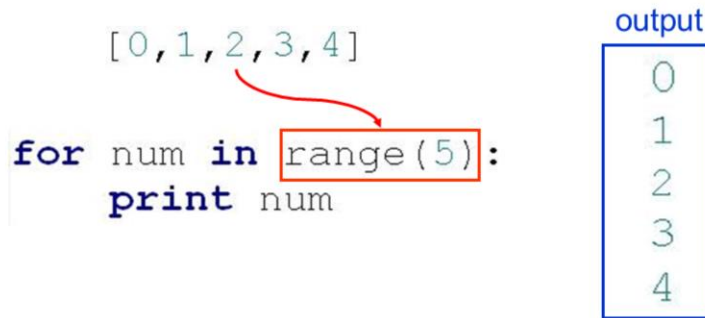
- The range function makes it easy to set the number of times a for loop will run.

```
[0, 1, 2, 3, 4]
```

```
for num in range(5):  
    print num
```

output

```
0  
1  
2  
3  
4
```

A diagram illustrating the use of the range function in a for loop. At the top left, a list [0, 1, 2, 3, 4] is shown. A red arrow points from the number 5 in the range(5) function of the code below to the list. The code is: for num in range(5): print num. To the right of the code, the word 'output' is written above a vertical list of numbers: 0, 1, 2, 3, 4. The entire diagram is enclosed in a thin black border.

8

A sequential list created by the range function can be convenient if you want to create a for loop that runs a specific number of times.

The range function is equivalent to a list and can be included directly in the for loop header line.

The range function will create the list before the for loop runs.

The loop in this example will simply print the values in the sequential list which range from 0 to 5.