Python-2

April 12, 2019

1 if Statement

Decision making is required when we want to execute a code only if a certain condition is satisfied.

The Structure will be like this:

if test_expression:

```
code 1
code 2
  .
  .
  .
code n
```

The if..else statement evaluates test expression and will execute body of if only when test condition is True.

If the condition is False, body of else is executed. Indentation is used to separate the blocks.

The structure will be like this:

if test_expression:

```
body of if
```

else:

```
body of else
```

The elif is short for else if. It allows us to check for multiple expressions.

If the condition for if is False, it checks the condition of the next elif block and so on.

If all the conditions are False, body of else is executed.

Only one block among the several if. . . elif. . . else blocks is executed according to the condition.

The if block can have only one else block. But it can have multiple elif blocks.

The structure will be like this:

if test_expression:

```
body of if
```

elif test_expression:

```
body of elif
```

else:

```
body of else
```

We can have a if...elif...else statement inside another if...elif...else statement. This is called nesting in computer
programming.
Any number of these statements can be nested inside one another. Indentation is the only way to figure out the level of
nesting. This can get confusing, so must be avoided if we can.

2 while Loop

The while loop in Python is used to iterate over a block of code as long as the test expression (condition) is true.
We generally use this loop when we don't know beforehand, the number of times to iterate.
The structure will be like this:
while test_expression:

```
body of while
```

Python interprets any non-zero value as True. None and 0 are interpreted as False.
The else part is executed if the condition in the while loop evaluates to False.
The while loop can be terminated with a break statement. In such case, the else part is ignored. Hence, a while loop's else
part runs if no break occurs and the condition is false.
Here is an example to illustrate this.
The structure will be like this:
while test_expression:

```
body of while
```

else:

```
body of else
```

3 for loop

The for loop in Python is used to iterate over a sequence (list, tuple, string) or other iterable objects. Iterating over a
sequence is called traversal.
The structure will be like this:
for var in iterable:

```
body of for
```

Here, val 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. The body of for loop is separated from the rest of the code using

indentation.

A for loop can have an optional else block as well. The else part is executed if the items in the sequence used in for loop

exhausts.

break statement can be used to stop a for loop. In such case, the else part is ignored.

Hence, a for loop's else part runs if no break occurs.

The structure will be like this:

for var in iterable:

```
body of for
```

else:

```
body of else
```

3.1 Looping Techniques

1. Range Function
2. Enumerate Function
3. Reversed Function
4. Zip Function

4 Control Flow Tools

1. Break
2. Continue
3. pass