Study Guide: for Loops

This study guide provides a summary of what you learned in this segment and serves as a guide for the upcoming practice quiz.

In the **for** Loops segment, you learned about the logical structure and syntax of **for** loops. You took a closer look at the **range()** function. You learned about nested **for** loops and complex nested **for** loops with **if** statements. You also learned how to fix common errors in **for** loops.

for Loops vs. **while** Loops

for loops and **while** loops share several characteristics. Both loops can be used with a variety of data types, both can be nested, and both can be used with the keywords **break** and **continue**. However, there are important differences between the two types of loops:

- **while** loops are used when a segment of code needs to execute repeatedly **while** a condition is true
- **for** loops iterate over a sequence of elements, executing the body of the loop **for** each element in the sequence

Syntax

The syntax of a **for** loop with the **in** keyword:

```
for variable in sequence:
    body of loop
```

Common **for** Loop Structures

for Loop with **range()**

The **in** keyword with the **range()** function generates a sequence of integer numbers, which can be used with a **for** loop to configure the iterations of the code. The range of numbers [0, 1, 2] correlates to ordinal index positions (1st, 2nd, 3rd), rather than the cardinal (quantity) values of the numbers 0, 1, and 2. For example, range(5) means the five index positions in the range [0, 1, 2, 3, 4].

The **range()** function can take up to three parameters. The roles of the three possible **range(x,y,z)** parameters are:

- **x = Start** - Starting index position of the range
  - Default index position is 0.
  - The starting index position is included in the range.
  - Example syntax: range(**2**, y, z) or range(**x+3**, y, z)
- **y = Stop** - Ending index position of range
  - No default index position. Must include the ending index position in the range() parameters.
    - Example syntax: range(**y**)
  - The value of the ending index position is excluded from the range.
  - To include the ending index number, use the expression: **y+1** (index + 1)
    - Example syntax: range(x, **y+1**, z)
    - Alternatively, if **y** = 10, you can write: range(x, **11**, z)
- **z = Step** - Incremental value
  - Default increment is +1.
  - The default value can be overridden with any valid increment.
  - The incremental value will end the for loop before it reaches the end of range index position (end of range index minus 1).

Example of a **for** loop with the **in** keyword and the **range()** function:

```
# This loop iterates on the value of the "number" variable in a range
# of 1 to 6+1 (the upper range limit of 6 is excluded, so +1 has
# been added to it to include 6 in the range). The incremental value
# for the loop is 2 (number+2). The print() function will output the
# resulting value of "number" multiplied by 3.


for number in range(1,6+1,2):
    print(number*3)


# The loop should print 3, 9, 15
```

Run

Reset

Common pitfalls when using the range() function:

- Forgetting that **the upper limit of a range() isn't included** in the range.
- **Iterating over non-sequences.** For example, strings are iterable letter by letter, but not word by word.

```
# This loop iterates on the value of the "number" variable in a range
# of 2 to 7 (the upper range limit of 8 is excluded). The print()
# function will output the resulting value of "number" squared.


for number in range(2,8):
    print(number**2)


# The loop should print 4, 9, 16, 25, 36, 49
```

Run

Reset

Nested **for** Loops

The syntax of nested **for** loops:

```
for x in sequence:
    # start of the outer loop body
    for y in sequence:
        # start of the inner loop body

        # end of of the inner loop body
    # continue body of the outer loop
    # end of the outer loop body

```

Example of nested **for** loops:

```
# This code demonstrates the outer and inner loop iterations of a pair
# of nested for loops. Click "Run" to see the results. The outer loop
# will run twice for the range pointer positions [0, 1] in range(2).
# The inner loop will run 4 times for the range pointer positions
# [0, 1, 2, 3] in range(3+1) or range(4) each time the outer loop runs.
# So, the inner loop will execute 8 times in total.


for x in range(2):
    print("This is the outer loop iteration number " + str(x))
    for y in range(3+1):
        print("Inner loop iteration number " + str(y))
    print("Exit inner loop")
```

Run

Reset

for Loop with nested **if** Statement

The syntax of a **for** Loop with nested **if** Statement:

```
for x in sequence:
    # start of body of for loop
    if condition is true:
        # start of body of if-statement

        # end of body of if-statement
    # continue body of for loop
    # end of body of for loop
```

Example of a **for** Loop with Nested **if** Statement:

```
# This for loop iterates through the numbers 0 to 6. The if statement
# uses the modulo operator to test if the "x" variable is divisible by
# 2. If True, the if statement will print the value of "x" and exit
# back into the for loop for the next iteration of "x". Since no
# incremental value is specified in the range() parameters, the default
# increment is +1.


for x in range(7):
    if x % 2 == 0:
        print(x)


# The loop should print 0, 2, 4, 6
```

Run

Reset

Python practice information

For additional Python practice, the following links will take you to several popular online interpreters and codepads:

- Welcome to Python
- Online Python Interpreter
- Create a new Repl
- Online Python-3 Compiler (Interpreter)
- Compile Python 3 Online
- Your Python Trinket

Mark as completed

Like Dislike Report an issue