Task: Beginner Control Structures - For Loop

www.hyperiondev.com

Introduction

Welcome to The Beginner Control Structures - For Loop Task!

In this task you will be sequentially exposed to loop statements in order to understand how they can be utilised in reducing lengthy code, preventing coding errors, as well as paving the way towards code reusability. The next statement you'll be exposed to is the for loop, which is essentially a different variation of the while loop.

Connect with your mentor

CONNECT

Remember that with our courses - you're not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to login to www.hyperiondev.com/support to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

A note from the Hyperion Team...

Python is a high-level language meaning it is closer to human languages than machine languages and is therefore easier to understand and write. It is also more or less independent of a particular type of computer. But what was the first high-level language? The answer to that question is Fortran.

Fortran was invented in 1954 by IBM's John Backus. The name Fortran is derived from "Formula Translation" and the language is best suited to numeric computation and scientific computing. Fortran is still used in computationally intensive areas such as numeric weather prediction, finite element analysis, computational fluid dynamics, computational physics, crystallography and computational chemistry.

- The Hyperion Team

What is a for Loop?

The for statement is an alternative loop structure for a while statement. This means that a for loop is equivalent to a while loop and one can apply a for loop where a while loop could be applied and vice-versa. The main difference between a while loop and a for loop is the syntax. A for loop allows for counter-controlled repetition to written more compactly and clearer therefore making it easier to read.

In the for loop below, the condition is that when the variable i (which is an integer) is in the range of 1 to 10 (i.e. either 1, 2, 3 ,4 ,5 ,6 ... or 9), the indented code in the body of the for loop will execute. range(1, 10) specifies that i = 1 in the first iteration of the loop. So 1 will be printed in the first iteration of this code. Then the code will run again, this time with i=2, and 2 will be printed out...etc. until i=10. Now i is not in the range (1,10) so the code will stop executing.

For the function range(start index: end index), start index IS included and end index IS NOT included. i is known as the index variable as it can tell you what 'iteration' or repetition the loop is on. In each iteration of the for loop, the code indented inside the for loop is repeated.

```
for i in range(1, 10):
        print(i)
```

This for loop prints the numbers 1 to 9. Again, note the indentation and the colon, just like in the if statement.

You can use an if statement within a for loop!

```
for i in range (1,10):
        if i > 5:
                print(i)
```

This will only print the numbers 6, 7, 8 and 9 because numbers less than or equal to 5 are filtered out.

In order for a for loop to function properly, the loop needs to be initialized and a loop test must be performed.

Initialise Loop

The loop needs to use a variable as it's counter variable. This variable will tell the computer how many times to execute the loop.

Loop Test

The loop test is a boolean expression, that is, the loop test is a Python expression such that when it is evaluated, the value of the expression is a boolean. The loop test expression is evaluated prior to any iteration of the for loop. If the condition is true then the program control is passed to the loop body; if false, control passes to the first statement after the loop body.

Update Statement

Update statements assign new values to the loop control variables. The statements typically use the increment i+=1 to update the control variable. An update statement is always only executed after the body has been executed. After the update statement has been executed, control passes to the loop test to mark the beginning of the next iteration.

Break Statement

Within a loop body, a break statement causes an immediate exit from the loop to the first statement after the loop body. The break allows for an exit at any intermediate statement in the loop.

```
break
```

Using a break statement to exit a loop has limited, but important applications. Let us describe one of these situations. A program may use a loop to input data from a file. The number of iterations depends on the amount of data in the file. The task of reading from the file is part of the loop body which thus becomes the place where the program discovers that data is exhausted. When the end-of-file condition becomes true, a break statement exits the loop.

Infinite Loops

In selecting a loop construct (either while loop or for loop) to read from a file, we recognise that the test for end-of-file occurs within the loop body. The loop statement has the form of an infinite loop: one that runs forever. The assumption is that we do not know how much data is in the file. Versions of the for loop and the while loop permit a programmer to create an infinite loop. In the for loop each field of the loop is empty. There are no control variables and no loop test. The equivalent while loop uses the constant true as the logical expression.

The syntax of infinite for and while loops are as follows:

```
for(range):
        loop block

while(true):
        loop block
```

Instructions

Before you get started we strongly suggest you start using Notepad++ or IDLE to open all text files (.txt) and python files (.py). Do not use the normal Windows notepad as it will be much harder to read.

First read example.py, open it using Notepad++ (Right click the file and select 'Edit with Notepad++') or IDLE.

- example.py should help you understand some simple Python. Every task will have example code to help you get started. Make sure you read all of example.py and try your best to understand.

- You may run example.py to see the output. Feel free to write and run your own example code before doing the Task to become more comfortable with Python.

- You are not required to read the entirety of Additional Reading.pdf, it is purely for extra reference.

Compulsory Task 1

Follow these steps:

- Save your program as tables.py.
- This program needs to display the timetables for any number.
- For example, say the user enters 6 the program must print:

```
The 6 times table is:
        6 x 1 = 6
        6 x 2 = 12
        ...
        6 x 12 = 72
```

- Compile, save and run your file.

Compulsory Task 2

Follow these steps:

A simple rule to determine a whether a year is a leap year is to test whether it is a multiple of 4.

- Write a program to input a year and a number of years.
- Then determine and display which of those years were or will be leap years.

```
What year do you want to start with?      1994
How many years do you want to check?     8

1994 isn't a leap year
1995 isn't a leap year
1996 is a leap year
1997 isn't a leap year
1998 isn't a leap year
1999 isn't a leap year
2000 is a leap year
2001 isn't a leap year
```

- Compile, save and run your file.

Things to look out for:

1. Make sure that you have installed and setup all programs correctly. You have setup **Dropbox** correctly if you are reading this, but **Python or Notepad++** may not be installed correctly.
2. If you are not using Windows, please ask your mentor for alternative instructions.

Give your thoughts..

RATE

Hyperion strives to provide internationally-excellent course content that helps you achieve your learning outcomes. Think the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.