Hyperiondev

TASK:

Capstone Project I: Nested Loops

Visit our website

Introduction

WELCOME TO *THE FIRST CAPSTONE* PROJECT!

Now that you have experience using loops, we are going to consolidate that knowledge by introducing you to nested loops.

A nested loop, simply put, is a loop inside a loop. This task will help you understand what nested loops are and how they are implemented.

Get in touch

Connect for support

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/portal** 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!

NESTED LOOPS

As previously stated, a nested loop is simply a loop within a loop. Each time the outer loop is executed, the inner loop is executed right from the start. That is, all the iterations of the inner loop are executed with each iteration of the outer loop.

Take a look at the syntax for nested loops below.

The syntax for a nested for loop in another for loop is as follows:

```
for iterating_var in sequence:
   for iterating_var in sequence:
      statements(s)
   statements(s)
```

The syntax for a nested while loop in another while loop is as follows:

```
while expression:
   while expression:
    statement(s)
   statement(s)
```

The loops that make up a nested loop don't have to be the same; you can put any type of loop inside of any other type of loop. For example, a for loop can be inside a while loop or vice versa.

The following program shows the potential of a nested loop:

```
for x in range(1, 6):
    for y in range(1, 6):
        print(str(x) + " * " + str(y) + " = " + str(x*y))
    print("")
```

When the above code is executed, it produces following result :

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5

2 * 1 = 2
2 * 2 = 4
```

Note a few things in the example program above:

1. In the displayed output, x and y take on values from 1 to 5 while the range is defined from (1, 6). This is because *range(start, end)* does not include *end* as programmers prefer 0-based indexing, which means calling range(0, 10) returns 10 elements starting from 0, namely: 0, 1, 2, 3, 4, 5, 6, 7, 8 ,9.

2. Try running the above program without the final **`print("")`**.

- Do you notice the difference **`print("")`** creates in the output?
- What happens if you provide an extra indentation before **`print("")`**?

3. You can also get the same output using the formatting operator **`%`**. Try running the following:

```
for x in range(1, 6):
    for y in range(1, 6):
        print ('%d * %d = %d' % (x, y, x*y))
    print ("")
```

NESTED IF STATEMENTS

You can also nest **`if`** statements either within another **`if`** statement or within a loop.

Consider the example below:

```
print ("Example of a nested if statement:")
name = "B"
if name == "A":
        if name =="B":
                print ("It isn't possible for this code to execute, how can
the variable name be two things at once?")
else:
        print ("Your name isn't A but I can't automatically assume from that
that it is B.")
```

Note the indentation carefully. Here you have an **`if`** within an **`if`** and then an **`else`**.

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.

Compulsory Task 1

Follow these steps:

- Create a new Python file in this folder called **tritable.py**.
- Write a program that uses nested for loops to create the following number pyramid.

```
1
2   4
3   6    9
4   8    12   16
5   10   15   20   25
6   12   18   24   30   36
7   14   21   28   35   42   49
8   16   24   32   40   48   56   64
9   18   27   36   45   54   63   72   81
```

Compulsory Task 2

Follow these steps:

- Create a new Python file in this folder called **prime.py**
- Create a program to check if a number inputted by the user is prime.
- A prime number is a positive integer greater than 1, whose only factors are 1 and the number itself.
- Examples of prime numbers are: 2, 3, 5, 7, etc.
- Ask the user to enter an integer.
- First check if the number is greater than 1.
- If it is greater than 1, check to see if it has any factors besides one and itself. (That is, if there are any numbers between 2 and the number itself that can divide the number without any remainders.)
- If the number is a prime number, print out the number and ' is a prime number!'
- If the number is not a prime number, print out the number followed by ' is not a prime number'.

Completed the task(s)?

Ask your mentor to review your work!

Review work

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that 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.