Python do...while Loop Statement Emulation

Summary: in this tutorial, you'll learn how to emulate the `do...while` loop statement in Python

Introduction to the do...while loop statement

If you have come from other programming languages such as JavaScript (https://www.javascripttutorial.net/javascript-do-while/), Java, or C#, you're already familiar with the `do...while` loop statement.

Unlike the `while` (https://www.pythontutorial.net/python-basics/python-while/) loop, the `do...while` loop statement executes at least one iteration. It checks the `condition` at the end of each iteration and executes a code block until the `condition` is `False`.

The following shows the pseudocode for the `do...while` loop in Python:

```
do
    # code block
while condition
```

Unfortunately, Python doesn't support the `do...while` loop. However, you can use the `while` loop and a `break` (https://www.pythontutorial.net/python-basics/python-break/) statement to emulate the

`do...while` loop statement.

First, specify the `condition` as `True` in the `while` loop like this:

```
while True:
    # code block
```

This allows the code block to execute for the first time. However, since the condition is always `True`, it creates an indefinite loop. This is not what we expected.

Second, place a condition to break out of the `while` loop:

```
while True:
    # code block

    # break out of the loop
    if condition
        break
```

In this syntax, the code block always executes at least one for the first time and the condition is checked at the end of each iteration.

Python do...while loop emulation example

Suppose that you need to develop a number guessing game with the following logic:

- First, generate a random number within a range e.g., 0 to 10.
- Then, repeatedly prompt users for entering a number. If the entered number is lower or higher than the random number, give users a hint. If the entered number equals the random number, the loop stops.

The following program uses a `while` loop to develop the number guessing game:

```
from random import randint
```

```
# determine the range
MIN = 0
MAX = 10

# generate a secret number
secret_number = randint(MIN, MAX)

# initialize the attempt
attempt = 0

# The first attempt
input_number = int(input(f'Enter a number between {MIN} and {MAX}:'))
attempt += 1

if input_number > secret_number:
    print('It should be smaller.')
elif input_number < secret_number:
    print('It should be bigger.')
else:
    print(f'Bingo! {attempt} attempt(s)')

# From the second attempt
while input_number != secret_number:

    input_number = int(input(f'Enter a number between {MIN} and {MAX}:'))
    attempt += 1

    if input_number > secret_number:
        print('It should be smaller.')
    elif input_number < secret_number:
        print('It should be bigger.')
    else:
        print(f'Bingo! {attempt} attempt(s)')
```

The following shows a sample run:

```
Enter a number between 0 and 10:5
It should be bigger.
Enter a number between 0 and 10:7
It should be bigger.
Enter a number between 0 and 10:8
Bingo! 3 attempt(s)
```

Since the `while` loop checks for the condition at the beginning of each iteration, it's necessary to repeat the code that prompts for user input and checking the number twice, one before the loop and one inside the loop.

To avoid this duplicate code, you can use a `while` loop to emulate `do while` loop as follows:

```
from random import randint

# determine the range
MIN = 0
MAX = 10

# generate a secret number
secret_number = randint(MIN, MAX)

# initialize the attempt
attempt = 0

while True:
    attempt += 1

    input_number = int(input(f'Enter a number between {MIN} and {MAX}:'))

    if input_number > secret_number:
        print('It should be smaller.')
```

```
    elif input_number < secret_number:
        print('It should be bigger.')
    else:
        print(f'Bingo! {attempt} attempt(s)')
        break
```

How it works.

- First, remove the code before the `while` loop.
- Second, add the condition to stop the loop if the entered number equals the random number by using the `break` statement.

Summary

- Python doesn't support the do-while loop statement.
- Use a `while` loop and the `break` statements to emulate a `do...while` loop in Python