Python Floor Division

Summary: in this tutorial, you'll learn about Python floor division operator (//) or mod.

Introduction to Python floor division

Suppose you have a division of two integers:

```
101 / 4
```

In this division, 101 is called a numerator (`N`) and 4 is called a denominator (`D`).

The integer division 101 / 4 returns 25 with the remainder 1. In other words:

```
101 / 4 = 25 with remainder 1
```

Or put it in another way:

```
101 = 4 * 25 + 1
```

Python uses two operators `//` and `%` that returns the result of the division:

```
101 // 4 = 25
101 % 4 = 1
```

The `//` is called the **floor division** operator or div. And the `%` is called the **modulo** operator or mod.

This tutorial focuses on the floor division operator. You'll learn about the modulo operator in the following tutorial (https://www.pythontutorial.net/advanced-python/python-modulo/) .

Both floor division and modulo operators satisfy the following equation:

```
101 = 4 * (101 // 4) + (101 % 4)
101 = 4 * 25         + 1
```

Generally, if `N` is the numerator and `D` is the denominator, then the floor division and modulo operators always satisfy the following equation:

```
N = D * ( N // D) + (N % D)
```

The floor division in Python

To understand the floor division, you first need to understand the floor of a real number.

The floor of a real number is the largest integer less than or equal to the number. In other words:

```
floor(r) = n, n is an integr and n <= r
```

For example, the floor of 3.4 is 3 because 3 is the largest integer less than or equal to 3.4. The floor of 3.9 is also 3. And the floor of 3 is 3 obviously:

```
floor(3.4) = 4
floor(3.9) = 3
floor(3) = 3
```

For the positive numbers, it would be easy to understand the definition. However, you should pay attention when it comes to negative numbers.

For example, the floor of `-3.4` returns `-4`, not `-3` based on the floor definition. Similarly, the floor of `-3.9` also returns `-4`.

```
floor(-3.4) = -4
floor(-3.9) = -4
floor(-3) = -3
```

The floor division can be defined as:

```
n // d = floor(n/d)
```

Notice that the floor division of a number is not always the same as truncation. The floor division is the same as truncation only when the numbers are positive.

Python floor division operator examples

The following example uses the floor division operators with positive and negative integers:

```
a = 10
b = 3

print(a//b)  # 3

a = -10
b = -3

print(a//b)  # 3

a = 10
b = -3
print(a//b)  # -4
```

```
a = -10
b = 3
print(a//b)  # -4
```

Output:

```
3
3
-4
-4
```

The following table illustrates the floor division of two integers `a` and `b` :

a	b	a // b
10	3	3
-10	-3	3
10	-3	-4
-10	3	-3

Python math.floor() function

The `floor()` function of the `math` module returns the floor division of two integers. For example:

```
from math import floor

a = 10
b = 3
```

```
print(a//b)  # 3
print(floor(a/b))  # 3
```

Output:

```
3
3
```

As you can see clearly from the output, the `floor()` function returns the same result as the floor division operator (`//`). It's also true for the negative numbers:

```
from math import floor

a = 10
b = -3

print(a//b)  # -4
print(floor(a/b))  # -4
```

Output:

```
-4
-4
```

Summary

- Python uses // as the floor division operator and `%` as the modulo operator.
- If the numerator is N and the denominator D, then this equation `N = D * ( N // D) + (N % D)` is always satisfied.
- Use floor division operator `//` or the `floor()` function of the `math` module to get the floor division of two integers.