03 | Link : https://edabit.com/challenge/noqONSr5o9qzvXWzL

โจทย์

Right Shift by Division

Published by Deep Xavier in Python

complete bit_operations numbers

The **right shift** operation is similar to **floor division by powers of two.**

Sample calculation using the right shift operator (`>>`):

```
80 >> 3 = floor(80/2^3) = floor(80/8) = 10
-24 >> 2 = floor(-24/2^2) = floor(-24/4) = -6
-5 >> 1 = floor(-5/2^1) = floor(-5/2) = -3
```

Write a function that **mimics** (without the use of >>) the right shift operator and returns the result from the two given integers.

Examples

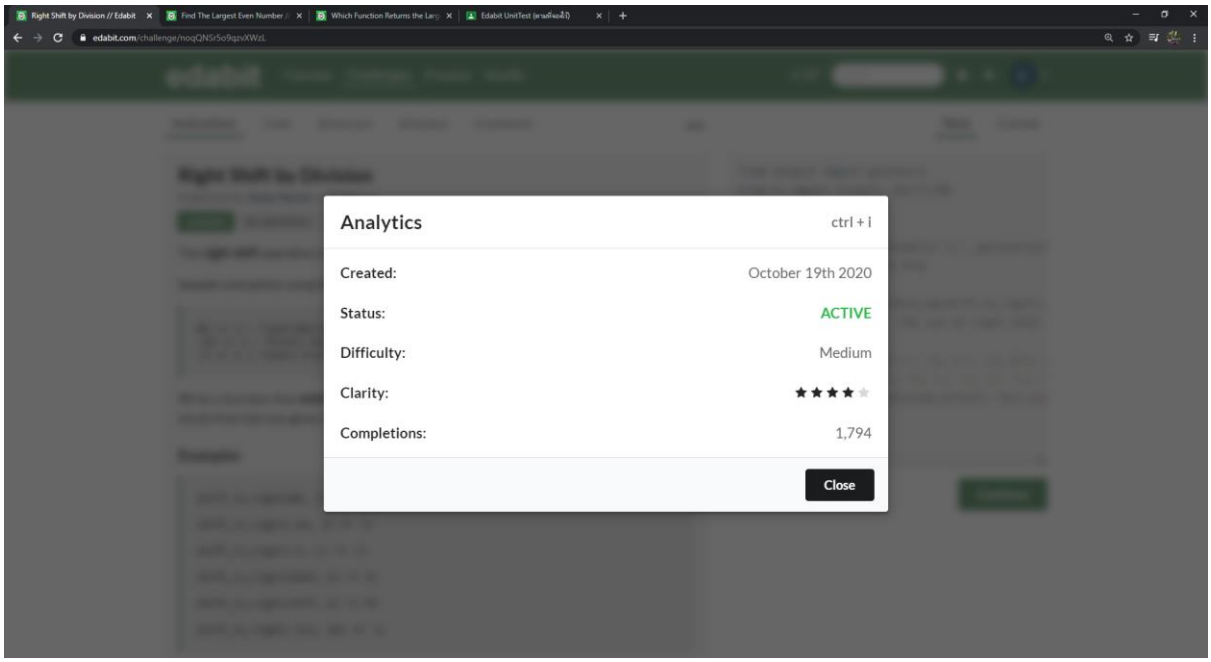
```
shift_to_right(80, 3) → 10

shift_to_right(-24, 2) → -6

shift_to_right(-5, 1) → -3

shift_to_right(4666, 6) → 72

shift_to_right(3777, 6) → 59

shift_to_right(-512, 10) → -1
```

Notes

- There will be no negative values for the second parameter `y`.
- This challenge is more like recreating of the **right shift** operation, thus, **the use of the operator directly** is **prohibited.**
- Alternatively, you can solve this challenge via recursion.
- A recursive version of this challenge can be found via this link.

SUGGEST EDIT

ระดับความยาก



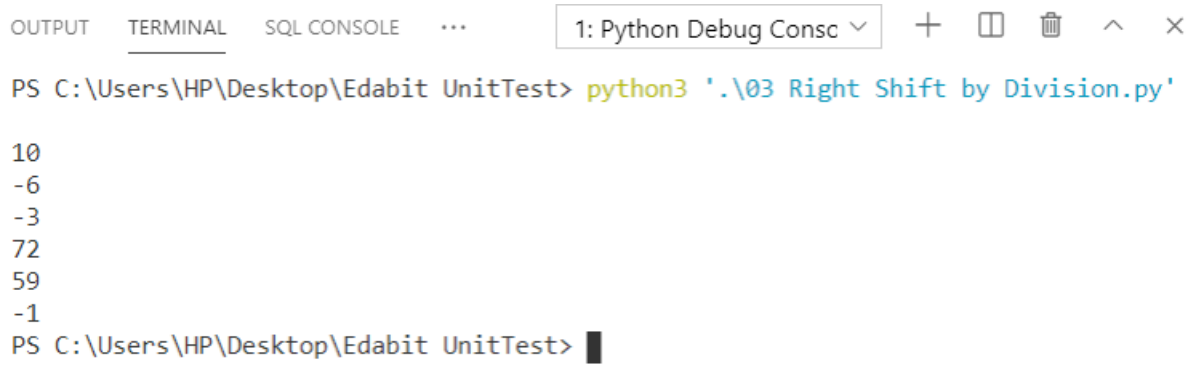
อธิบายโจทย์

การคำนวณค่าของตัวดำเนินการแบบบิต โดยใช้เครื่องหมาย >> หมายถึงการเลื่อนบิตไปทางขวามือ (Right Shift) ซึ่งมีค่าเท่ากับ 2^(shift) โดยเขียนฟังก์ชันเลียนแบบโดยไม่ใช้ >> โดยส่งผลลัพธ์จากจำนวนเต็มสองจำนวนที่กำหนดไว้

อธิบายการทำงานของโปรแกรม

ฟังก์ชัน shift_to_right คือฟังก์ชัน ตัวดำเนินการระดับบิต (Bitwise Operator) การเลื่อนบิตไปทางขวามือ (Right Shift) โดยรับค่าเข้ามาผ่านตัวแปร x และ y โดยให้ตัวแปร x เก็บค่าที่รับเข้ามาตัวตั้ง และให้ตัวแปร y เป็น Shift หมายถึงตัวเลขจำนวนที่ต้องเลื่อนบิตไปทางขวา โดย return กลับไปที่ฟังก์ชัน shift_to_right โดยเริ่มทำจากในวงเล็บก่อน โดยที่เลข 2 เป็นค่าที่โจทย์กำหนดไว้ นำไป ** หรือ ยกกำลังกับตัวแปร y ที่รับเข้ามา เมื่อได้ค่ามาแล้วนำค่าไปคำนวณกับตัวแปร x โดย // (floor division) หรือ การหารไม่เอาเศษ

หน้าจอของผล run โปรแกรม



หน้าจอของผล run. Unit Test

Code Program

```
def shift_to_right(x, y):
    return (x // (2 ** y))


print(shift_to_right(80,3))
print(shift_to_right(-24,2))
print(shift_to_right(-5,1))
print(shift_to_right(4666,6))
print(shift_to_right(3777,6))
print(shift_to_right(-512,10))
```

Code Unit Test

```
import unittest
import RightShiftbyDivision03

class Test(unittest.TestCase):
    def shift_to_right (self):
        self.assertEqual(RightShiftbyDivision03.shift_to_right(80,3))
        self.assertEqual(RightShiftbyDivision03.shift_to_right(-24,2))
        self.assertEqual(RightShiftbyDivision03.shift_to_right(-5,1))
        self.assertEqual(RightShiftbyDivision03.shift_to_right(4666,6))
        self.assertEqual(RightShiftbyDivision03.shift_to_right(3777,6))
        self.assertEqual(RightShiftbyDivision03.shift_to_right(-512,10))

if __name__ == '__main__':
    unittest.main()
```