

# Solution Review: Convert Decimal Integer to Binary

This lesson contains the solution review for the challenge of converting integer values to their binary equivalents.

## WE'LL COVER THE FOLLOWING ^

- Implementation
- Explanation

Let's analyze the solution to the exercise in the previous lesson.

## Implementation #

```
def convert_int_to_bin(dec_num):  
    s = Stack()  
  
    while dec_num > 0:  
        remainder = dec_num % 2  
        s.push(remainder)  
        dec_num = dec_num // 2  
  
    bin_num = ""  
    while not s.is_empty():  
        bin_num += str(s.pop())  
  
    return bin_num  
  
print(convert_int_to_bin(56))  
print(convert_int_to_bin(2))  
print(convert_int_to_bin(32))  
print(convert_int_to_bin(10))  
  
print(int(convert_int_to_bin(56),2)==56)
```



## Explanation #

On **line 2**, we declare a stack and proceed to a **while** loop on **line 4** which executes if **dec\_num** is greater than 0.

As stated in the **division by 2** method, we calculate the remainder of the division of `dec_num` by 2 and push it onto the stack (**lines 5-6**). Then we divide `dec_num` by 2 using the `//` operator to `dec_num` which floors the answer of the division, and we update `dec_num` with the answer (**line 7**). We keep executing the code on **lines 5-7** as long as `dec_num` is greater than 0. As soon as `dec_num` becomes equal to or less than 0, the `while` loop terminates.

On **line 9**, `bin_num` is declared as an empty string. The `while` loop on the very next line executes if the stack `s` is *not* empty. If `s` is not empty, we pop a value from `s` and append it to the `bin_num` string on **line 11**. We keep popping elements from `s` until it becomes empty and the `while` loop is terminated.

The `bin_num` is returned from the function on **line 13**.

The following code helps us to evaluate whether our implementation is correct or not:

```
print(int(convert_int_to_bin(56),2)==56)
```

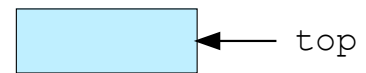
The above statement will print `True` if `convert_int_to_bin(56)` returns the correct binary equivalent for `56`. We convert the returned value from `convert_int_to_bin(56)` to an integer value by specifying base `2` of the returned value. It will convert to `56` if it's equal to `56` in binary format. Otherwise, the statement will print `False` if we get some number other than `56`.

In this problem, the *First-In, Last-Out* property of the stack has enabled us to store the binary bits from the *MSB* (Most Significant Bit) to the *LSB* (Least Significant Bit), although we get the values in reverse order by the *division by 2* method.

Below are some slides that will help you understand the code even better:

→

```
def convert_int_to_bin(dec_num):  
    s = Stack()  
  
    while dec_num > 0:  
        remainder = dec_num % 2  
        s.push(remainder)  
        dec_num = dec_num // 2  
  
    bin_num = ""  
    while not s.is_empty():  
        bin_num += str(s.pop())  
  
    return bin_num
```

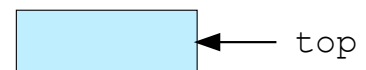


dec\_num = 10

1 of 29

→


```
def convert_int_to_bin(dec_num):  
    s = Stack()  
  
    while dec_num > 0:  
        remainder = dec_num % 2  
        s.push(remainder)  
        dec_num = dec_num // 2  
  
    bin_num = ""  
    while not s.is_empty():  
        bin_num += str(s.pop())  
  
    return bin_num
```



dec\_num = 10

2 of 29

```
def convert_int_to_bin(dec_num):  
    s = Stack()
```



```
    while dec_num > 0:  
        remainder = dec_num % 2  
        s.push(remainder)  
        dec_num = dec_num // 2
```

```
    bin_num = ""  
    while not s.is_empty():  
        bin_num += str(s.pop())
```


```
    return bin_num
```



```
dec_num = 10  
remainder = 0
```

3 of 29

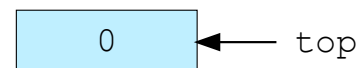
```
def convert_int_to_bin(dec_num):  
    s = Stack()
```



```
    while dec_num > 0:  
        remainder = dec_num % 2  
        s.push(remainder)  
        dec_num = dec_num // 2
```

```
    bin_num = ""  
    while not s.is_empty():  
        bin_num += str(s.pop())
```

```
    return bin_num
```



```
dec_num = 10  
remainder = 0
```

4 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



```
dec_num = 5
remainder = 0
```

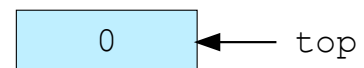
5 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



```
dec_num = 5
remainder = 0
```

6 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



dec\_num = 5  
remainder = 1

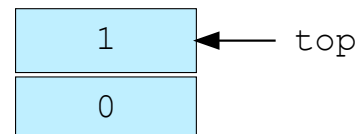
7 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



dec\_num = 5  
remainder = 1

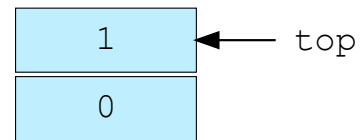
8 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



```
dec_num = 2
remainder = 1
```

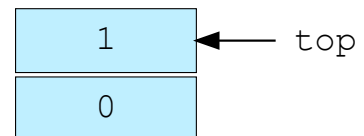
9 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



```
dec_num = 2
remainder = 1
```

10 of 29

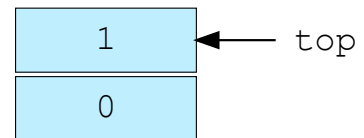
```
def convert_int_to_bin(dec_num):  
    s = Stack()
```



```
    while dec_num > 0:  
        remainder = dec_num % 2  
        s.push(remainder)  
        dec_num = dec_num // 2
```

```
    bin_num = ""  
    while not s.is_empty():  
        bin_num += str(s.pop())
```

```
    return bin_num
```



```
dec_num = 2  
remainder = 0
```

11 of 29

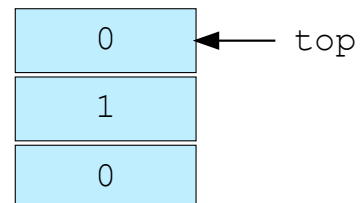
```
def convert_int_to_bin(dec_num):  
    s = Stack()
```



```
    while dec_num > 0:  
        remainder = dec_num % 2  
        s.push(remainder)  
        dec_num = dec_num // 2
```

```
    bin_num = ""  
    while not s.is_empty():  
        bin_num += str(s.pop())
```

```
    return bin_num
```



```
dec_num = 2  
remainder = 0
```

12 of 29

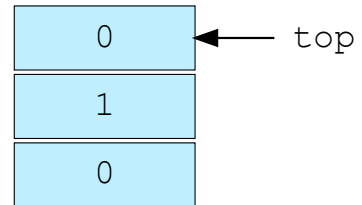


```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



dec\_num = 1  
remainder = 0

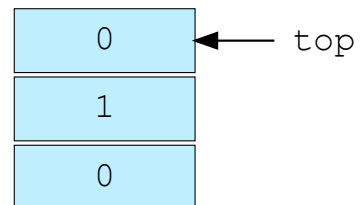
13 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



dec\_num = 1  
remainder = 0

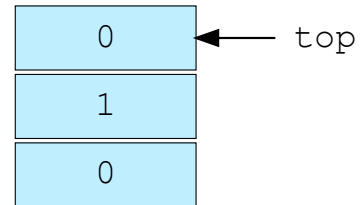
14 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



```
dec_num = 1
remainder = 1
```

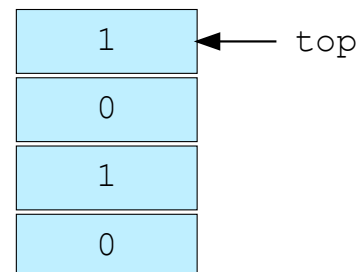
15 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



```
dec_num = 1
remainder = 1
```

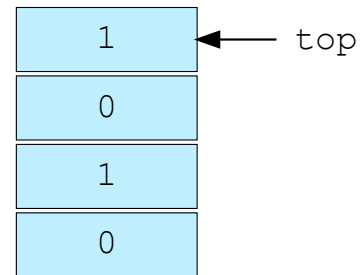
16 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



dec\_num = 0  
remainder = 1

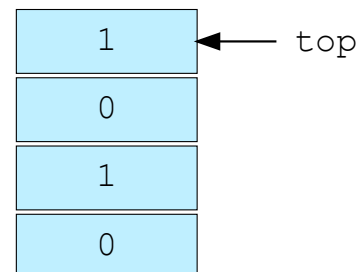
17 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



dec\_num = 0  
remainder = 1

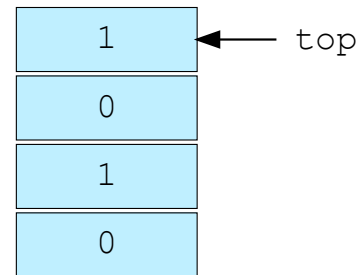
18 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



```
dec_num = 0
remainder = 1
bin_num = ""
```

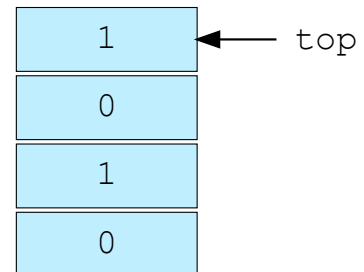
19 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



```
dec_num = 0
remainder = 1
bin_num = ""
```

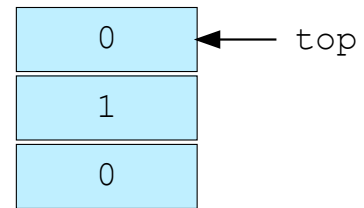
20 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



```
dec_num = 0
remainder = 1
bin_num = "1"
```

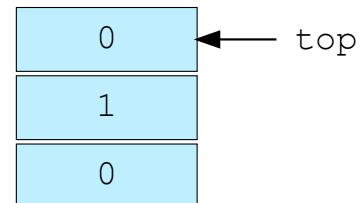
21 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



```
dec_num = 0
remainder = 1
bin_num = "1"
```

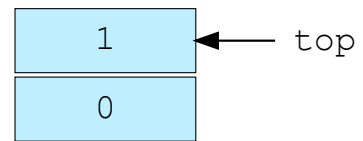
22 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



```
dec_num = 0
remainder = 1
bin_num = "10"
```

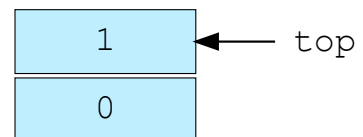
23 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



```
dec_num = 0
remainder = 1
bin_num = "10"
```

24 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



```
dec_num = 0
remainder = 1
bin_num = "101"
```

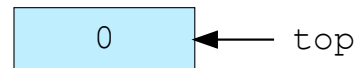
25 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



```
dec_num = 0
remainder = 1
bin_num = "101"
```

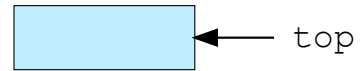
26 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



```
dec_num = 0
remainder = 1
bin_num = "1010"
```

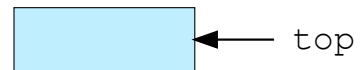
27 of 29

```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



```
dec_num = 0
remainder = 1
bin_num = "1010"
```

28 of 29

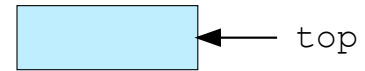


```
def convert_int_to_bin(dec_num):
    s = Stack()

    while dec_num > 0:
        remainder = dec_num % 2
        s.push(remainder)
        dec_num = dec_num // 2

    bin_num = ""
    while not s.is_empty():
        bin_num += str(s.pop())

    return bin_num
```



```
dec_num = 0
remainder = 1
bin_num = "1010"
```

29 of 29

—

[ ]

Hope everything's clear up until now! We'll now move on to the next chapter which is all about Singly Linked Lists.