Determine whether an integer is a palindrome
An integer is a palindrome when it reads the same backward as forward

Example 1:	Example 2:	Example 3:
Input: 121	Input: -121	Input: 10
Output: True	Output: False	Output: False

Could you solve it without converting the integer to a string?

Converting the input to string:

```
def is_Palindrome(input):
    return (str(x) == str(x)[::-1])
```

original input

original put casted to a string and reversed

```
def is_Palindrome(input):
    number, reversed_number = input, 0

    while number > 0:
        last_digit = number % 10

        reversed_number = (reversed_number * 10) + last_digit

        number = int(number / 10)

    return x == reversed_number
```

→ Chop off the last number of the input #

↳ start creating the reversed number

↳ update the input number

↳ return this boolean

Breakdown with input: 121 (should return True)

1st iteration

input: 121
number = 121
reversed_number = 0

while number > 0:

last_number = number % 10 → 121 % 10 = 1

$$10 \overline{)121} \quad 12 \text{ r } 1$$

* last_number of the input number

that's what the modulo operator does... (kind of)

reversed_number = int(0 * 10) + 1 = 1

↳ creating the reversed number from the numbers we chop off

number = int(number/10) = int(121/10) = 12

* We already chopped off the last_digit so now we need to move forward with the input number

2nd iteration

number = 12
reversed_number = 1

while 12 > 0:

last_digit = 12 % 10 → $10 \overline{)12} \quad 1 \text{ r } 2$ = 2

reversed_number = (1 * 10) + last_digit = 10 + 2 = 12

number = int(number/10) = int(12/10) = 1

3rd iteration

number = 1
reversed_number = 12

while 1 > 0 :

last_digit = 1 % 10 = 1

reversed_number = (12 * 10) + 1 = 120 + 1

number = int(1/10) = 0

4th iteration

number = 0
reversed_number = 121

while 0 > 0 ⟶ false

return input == reversed_number

↳ 121 == 121

↳ True

Time Complexity = O(n) → we go through the input in 1 pass

Space Complexity = O(1) → we store our result in a variable

The general gist: we chop off the last number of the input number AND store it

Then we create the reversed_number starting with last_digit

↳ to prepare for the next number we multiply it by 10

↳ 10 b/c it adds a zero

i.e) 1 x 10 = 10 + 2 = 12 ... etc

Then we update our input number so we can iterate all the way through the input...