**Problem #8: String to Integer (atoi). (Medium)**

<https://leetcode.com/problems/string-to-integer-atoi/description/>

**My Solution:**

1. Igrore any leading whitespace by using lstrip function on s.
2. For Signedness, set positive and negative to False.

If s is not an empty string and if the character at index 0 is ‘+’, then set positive to True, and if the character at index 0 is ‘-‘ set negative to True. Also omit index 0 and make string s start from index 1.

1. For leading zeros, left strip ‘0’ from s using lstrip function.
2. Set num to be
3. Iterate through the characters in s from beginning to end.

If the character at index I is a a digit, then multiply num by 1 and then the difference between ordinal of the character at index I and ordinal of “0”.

If the character is not a digit, then break from any further processing.

1. Apply the sign -- if negative is True, then multiply num by (-1).
2. If num is greater than (2\*\*31 ) – 1, then return (2 \*\*31) – 1.
3. If num is less than -(2\*\*31 ), then return -(2 \*\*31).
4. Return num .

class Solution:

def myAtoi(self, s: str) -> int:

# ignore any leading whitespace

s = s.lstrip()

# Signedness

positive = False

negative = False

if s:

if s[0] == '+':

positive = True

s = s[1:]

elif s[0] == '-':

negative = True

s = s[1:]

# Leading zeros.

s = s.lstrip("0")

num = 0

for i in range(len(s)):

if s[i].isdigit():

num \*= 10

num += ord(s[i]) - ord("0")

else:

break

# Apply the sign

if negative:

num \*= (-1)

if num > (2 \*\* 31) - 1:

return (2 \*\* 31) - 1

elif num < (-1) \* (2 \*\* 31):

return (-1) \*(2 \*\* 31)

return num