<https://leetcode.com/problems/valid-number/description/>

A **valid number** can be split up into these components (in order):

A **decimal number** or an **integer**.

(Optional) An 'e' or 'E', followed by an **integer**.

A **decimal number** can be split up into these components (in order):

(Optional) A sign character (either '+' or '-').

One of the following formats:

One or more digits, followed by a dot '.'.

One or more digits, followed by a dot '.', followed by one or more digits.

A dot '.', followed by one or more digits.

An **integer** can be split up into these components (in order):

(Optional) A sign character (either '+' or '-').

One or more digits.

For example, all the following are valid numbers: ["2", "0089", "-0.1", "+3.14", "4.", "-.9", "2e10", "-90E3", "3e+7", "+6e-1", "53.5e93", "-123.456e789"], while the following are not valid numbers: ["abc", "1a", "1e", "e3", "99e2.5", "--6", "-+3", "95a54e53"].

Given a string s, return true *if* s *is a* ***valid number***.

**Example 1:**

Input: s = "0"

Output: true

**Example 2:**

Input: s = "e"

Output: false

**Example 3:**

Input: s = "."

Output: false

**Constraints:**

* 1 <= s.length <= 20
* s consists of only English letters (both uppercase and lowercase), digits (0-9), plus '+', minus '-', or dot '.'.

**Attempt 1: 2023-08-14**

**Solution 1: String handling in intuitive way (120 min)**

class Solution {

// 1. No matter a decimal number or an integer, must include at least one digit

// 2. We should only have one '.'

// 3. If '.' presents it means a potential valid decimal present, it must before 'e' or 'E', otherwise violate 'e' or 'E' followed by an integer rule (see Condition 5)

// 4. We should only have one 'e' or 'E'

// 5. If 'e' or 'E' presents must before an integer, cannot be followed by decimal

// 6. If 'e' or 'E' presents must not be first poistion, either a decimal or integer must be a prefix

// 7. If 'e' or 'E' presents must follow by an integer, it cannot be the last position

// 8. If '+' or '-' presents must at first position or immediately follow with 'e' or 'E'

// 9. Any other character break the validation

public boolean isNumber(String s) {

boolean dotSeen = false;

boolean numSeen = false;

boolean eSeen = false;

for(int i = 0; i < s.length(); i++) {

char c = s.charAt(i);

if(c >= '0' && c <= '9') {

numSeen = true;

} else if(c == '.') {

// Condition 2 & 3

if(dotSeen || eSeen) {

return false;

}

dotSeen = true;

} else if(c == 'e' || c == 'E') {

// Condition 4 & 5 & 6

if(eSeen || !numSeen) {

return false;

}

eSeen = true;

// Reset based on Condition 7, e.g "1e" not valid

numSeen = false;

} else if(c == '+' || c == '-') {

// Condition 8, e.g "3E+7" expect true, "005047e+6" expect true

if(i != 0 && s.charAt(i - 1) != 'e' && s.charAt(i - 1) != 'E') {

return false;

}

} else {

// Condition 9

return false;

}

}

// Don't directly return true, based on Condition 1, e.g "." expect false

return numSeen;

}

}

**Refer to**

<https://leetcode.com/problems/valid-number/solutions/23738/clear-java-solution-with-ifs/>

public boolean isNumber(String s) {

s = s.trim();

boolean pointSeen = false;

boolean eSeen = false;

boolean numberSeen = false;

boolean numberAfterE = true;

for(int i=0; i<s.length(); i++) {

if('0' <= s.charAt(i) && s.charAt(i) <= '9') {

numberSeen = true;

numberAfterE = true;

} else if(s.charAt(i) == '.') {

if(eSeen || pointSeen) {

return false;

}

pointSeen = true;

} else if(s.charAt(i) == 'e') {

if(eSeen || !numberSeen) {

return false;

}

numberAfterE = false;

eSeen = true;

} else if(s.charAt(i) == '-' || s.charAt(i) == '+') {

if(i != 0 && s.charAt(i-1) != 'e') {

return false;

}

} else {

return false;

}

}

return numberSeen && numberAfterE;

}

We start with trimming.

* If we see [0-9] we reset the number flags.
* We can only see . if we didn't see e or ..
* We can only see e if we didn't see e but we did see a number. We reset numberAfterE flag.
* We can only see + and - in the beginning and after an e
* any other character break the validation.

At the and it is only valid if there was at least 1 number and if we did see an e then a number after it as well.

So basically the number should match this regular expression:

[-+]?(([0-9]+(.[0-9]\*)?)|.[0-9]+)(e[-+]?[0-9]+)?

**Refer to**

<https://leetcode.com/problems/valid-number/solutions/23738/clear-java-solution-with-ifs/comments/22978>

We can improve a little bit. The 'numberAfterE' is unnecessary.

class Solution {

public boolean isNumber(String s) {

s = s.trim();

boolean pointSeen = false;

boolean eSeen = false;

boolean numberSeen = false;

for(int i=0; i<s.length(); i++) {

if('0' <= s.charAt(i) && s.charAt(i) <= '9') {

numberSeen = true;

} else if(s.charAt(i) == '.') {

if(eSeen || pointSeen)

return false;

pointSeen = true;

} else if(s.charAt(i) == 'e') {

if(eSeen || !numberSeen)

return false;

numberSeen = false;

eSeen = true;

} else if(s.charAt(i) == '-' || s.charAt(i) == '+') {

if(i != 0 && s.charAt(i-1) != 'e')

return false;

} else

return false;

}

return numberSeen;

}

}