8.10. The Accumulator Pattern with Conditionals

Sometimes when we're accumulating, we don't want to add to our accumulator every time we iterate. Consider, for example, the following program which counts the number of letters in a phrase.

Save & Run 5/9/2021, 12:07:39 PM - 2 of 2 Show in CodeLens

```
1 phrase = "What a wonderful day to program"
2 tot = 0
3 for char in phrase:
4     if char != " ":
5         tot = tot + 1
6 print(tot)
7
```

```
26
```

Activity: 1 -- ActiveCode (ac7_10_1)

Here, we **initialize** the accumulator variable to be zero on line two.

We **iterate** through the sequence (line 3).

The **update** step happens in two parts. First, we check to see if the value of `char` is not a space. If it is not a space, then we update the value of our accumulator variable `tot` (on line 6) by adding one to it. If that conditional proves to be False, which means that char *is* a space, then we don't update `tot` and continue the for loop. We could have written `tot = tot + 1` or `tot += 1`, either is fine.

At the end, we have accumulated a the total number of letters in the phrase. Without using the conditional, we would have only been able to count how many characters there are in the string and not been able to differentiate between spaces and non-spaces.

We can use conditionals to also count if particular items are in a string or list. The following code finds all occurrences of vowels in the following string.

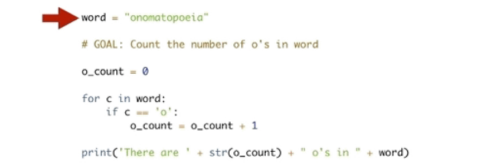
Save & Run 5/9/2021, 12:07:42 PM - 2 of 2 Show in CodeLens

```
1 s = "what if we went to the zoo"
2 x = 0
3 for i in s:
4     if i in ['a', 'e', 'i', 'o', 'u']:
5         x += 1
6 print(x)
7
```

```
8
```

Activity: 2 -- ActiveCode (ac7_10_2)

We can also use `==` to execute a similar operation. Here, we'll check to see if the character we are iterating over is an "o". If it is an "o" then we will update our counter.



```
word = "onomatopoeia"

# GOAL: Count the number of o's in word

o_count = 0

for c in word:
    if c == 'o':
        o_count = o_count + 1

print('There are ' + str(o_count) + " o's in " + word)
```

8.10.1. Accumulating the Max Value

We can also use the accumulation pattern with conditionals to find the maximum or minimum value. Instead of continuing to build up the accumulator value like we have when counting or finding a sum, we can reassign the accumulator variable to a different value.

The following example shows how we can get the maximum value from a list of integers.

Save & Run 5/9/2021, 12:07:44 PM - 2 of 2 Show in CodeLens

```
1 nums = [9, 3, 8, 11, 5, 29, 2]
2 best_num = 0
3 for n in nums:
4     if n > best_num:
5         best_num = n
6 print(best_num)
7
```

```
29
```

Activity: 3 -- ActiveCode (ac7_10_3)

Here, we initialize best_num to zero, assuming that there are no negative numbers in the list.

In the for loop, we check to see if the current value of n is greater than the current value of `best_num`. If it is, then we want to **update** `best_num` so that it now is assigned the higher number. Otherwise, we do nothing and continue the for loop.

You may notice that the current structure could be a problem. If the numbers were all negative what would happen to our code? What if we were looking for the smallest number but we initialized `best_num` with zero? To get around this issue, we can initialize the accumulator variable using one of the numbers in the list.

Save & Run 5/9/2021, 12:07:49 PM - 2 of 2 Show in CodeLens

```
1 nums = [9, 3, 8, 11, 5, 29, 2]
2 best_num = nums[0]
3 for n in nums:
4     if n > best_num:
5         best_num = n
6 print(best_num)
7
```

```
29
```

Activity: 4 -- ActiveCode (ac7_10_4)

The only thing we changed was the value of `best_num` on line 2 so that the value of `best_num` is the first element in `nums`, but the result is still the same!

Check your understanding

condition-10-1: What is printed by the following statements?

```
s = "We are learning!"
x = 0
for i in s:
    if i in ['a', 'b', 'c', 'd', 'e']:
        x += 1
print(x)
```

- A. 2
- B. 5
- C. 0
- D. There is an error in the code so it cannot run.

Check me Compare me

✔ Yes, we add to x each time we come across a letter in the list.

Activity: 5 -- Multiple Choice (question7_10_1)

condition-10-2: What is printed by the following statements?

```
list= [5, 2, 1, 4, 9, 10]
min_value = 0
for item in list:
   if item < min_value:
       min_value = item
print(min_value)
```

- A. 10
- B. 1
- C. 0
- D. There is an error in the code so it cannot run.

Check me | Compare me

✔ Yes, min_value was set to a number that was smaller than any of the numbers in the list, so it was never updated in the for loop.

Activity: 6 -- Multiple Choice (question7_10_2)

For each string in the list `words`, find the number of characters in the string. If the number of characters in the string is greater than 3, add 1 to the variable `num_words` so that `num_words` should end up with the total number of words with more than 3 characters.

Save & Run | 5/13/2021, 5:54:05 PM - 5 of 5 | Show in CodeLens

```
1 words = ["water", "chair", "pen", "basket", "hi", "car"]
2 num_words = 0
3 for i in words:
4     if len(i) > 3:
5         num_words += 1
6
```

Activity: 7 -- ActiveCode (ac7_10_5)

Result	Actual Value	Expected Value	Notes
Pass	3	3	Testing that num_words has the correct value.

You passed: 100.0% of the tests

Challenge For each word in `words`, add 'd' to the end of the word if the word ends in "e" to make it past tense. Otherwise, add 'ed' to make it past tense. Save these past tense words to a list called `past_tense`.

Save & Run | 5/9/2021, 12:15:53 PM - 2 of 2 | Show in CodeLens

```
1 words = ["adopt", "bake", "beam", "confide", "grill", "plant", "time", "wave", "wis
2 past_tense = []
3 for i in words:
4     if(i[len(i)-1] == 'e'):
5         i += 'd'
6     else:
7         i += 'ed'
8     past_tense.append(i)
9
```

Activity: 8 -- ActiveCode (ac7_10_7)

Result	Actual Value	Expected Value	Notes	
Pass	['ado...hed']	['ado...hed']	Testing that the past_tense list is correct.	Expand Differences
Pass	'else'	'words...d(i)\n'	Testing output (Don't worry about actual and expected values).	Expand Differences
Pass	'for'	'words...d(i)\n'	Testing output (Don't worry about actual and expected values).	Expand Differences

You passed: 100.0% of the tests

✔ Completed. Well Done!

| Back to top