9.11. The Accumulator Pattern with Strings

We can also accumulate strings rather than accumulating numbers, as you've seen before. The following program isn't particularly useful for data processing, but we will see more useful things later that accumulate strings.

Save & Run   Original - 1 of 1   Show in CodeLens

```
s = input("Enter some text")
ac = ""
for c in s:
    ac = ac + "-" + c + "-"

print(ac)

```

```
c-c-a-a-t-t-
```

Activity: 1 -- ActiveCode (ac8_10_1)

Look carefully at line 4 in the above program (`ac = ac + c + "-" + c + "-"`). In words, it says that the new value of `ac` will be the old value of `ac` concatenated with the current character, a dash, then the current character and a dash again. We are building the result string character by character.

Take a close look also at the initialization of `ac`. We start with an empty string and then begin adding new characters to the end. Also note that I have given it a different name this time, `ac` instead of `accum`. There's nothing magical about these names. You could use any valid variable and it would work the same (try substituting x for ac everywhere in the above code).

Check your understanding

seqmut-10-1: What is printed by the following statements:

```
s = "ball"
r = ""
for item in s:
   r = item.upper() + r
print(r)
```

- A. Ball
- B. BALL
- [✓ C. LLAB]

Check me   Compare me

✓ Yes, the order is reversed due to the order of the concatenation.

Activity: 2 -- Multiple Choice (question8_10_1)

1. For each character in the string already saved in the variable `str1`, add each character to a list called `chars`.

Save & Run   5/13/2021, 4:39:50 PM - 2 of 2   Show in CodeLens

```
str1 = "I love python"
# HINT: what's the accumulator? That should go here.
chars=[]
for i in str1:
    chars.append(i)

```

Activity: 3 -- ActiveCode (ac8_10_2)

Result	Actual Value	Expected Value	Notes

Pass	['l',... 'n']	['l',... 'n']	Testing that chars is assigned to correct values.	Expand Differences
Pass	'append'	'str1 ...d(i)\n'	Testing your code (Don't worry about actual and expected values).	Expand Differences

You passed: 100.0% of the tests

Assign an empty string to the variable `output`. Using the `range` function, write code to make it so that the variable `output` has 35 `a` s inside it (like `"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"`). Hint: use the accumulation pattern!

Save & Run 5/13/2021, 4:40:25 PM - 2 of 2 Show in CodeLens

```
output = ""

for i in range(0,35):
  output = output + 'a'
print(output)

```

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

Activity: 4 -- ActiveCode (ac6_6_6)

Result	Actual Value	Expected Value	Notes	
Pass	'aaaaa...aaaaa'	'aaaaa...aaaaa'	Testing that output has the correct value.	Expand Differences
Pass	'aaaaa...aaaaa'	'outpu...put)\n'	Testing your code (Don't worry about actual and expected values).	Expand Differences

You passed: 100.0% of the tests

You have attempted 5 of 4 activities on this page

e Accumulator Pattern with Lists"> ✔ Completed. Well D 9.12. Accumulator Pattern Strategies">Next Section - 9.12. Accumulator Pattern Strategies