String Concatenation

- Combines two strings into a new, longer string.
- Uses the same plus sign as addition.

```
s1 = "CS141"
s2 = "rocks!"
bigstring = s1 + s2
print(bigstring)
   # prints CS141rocks!
```

String Concatenation

- Unlike print(), string concatenation does not put spaces between your strings.

```
s1 = "CS141"
s2 = "rocks!"
bigstring = s1 + " " + s2
print(bigstring)
   # prints CS141 rocks!
```

Sample problem

- All professor email addresses at Rhodes are constructed from the professor's last name, followed by the initial letter of their first name.
- We want to design a function that takes a prof's first and last name and returns their email address.

```
def make_prof_email(first, last):
  init = first[0]
  address = last + init + "@rhodes.edu"
  return address

def main():
  firstname = input("First name: ")
  lastname = input("Last name: ")
  addr = make_prof_email(firstname, lastname)
  print("Email:", addr)
```

You try it

- Modify this program so it creates a student email address, not a professor's email.
- You'll need to change the code so it uses first name, last name, middle name, and class year.
- Your new function should take FOUR arguments.

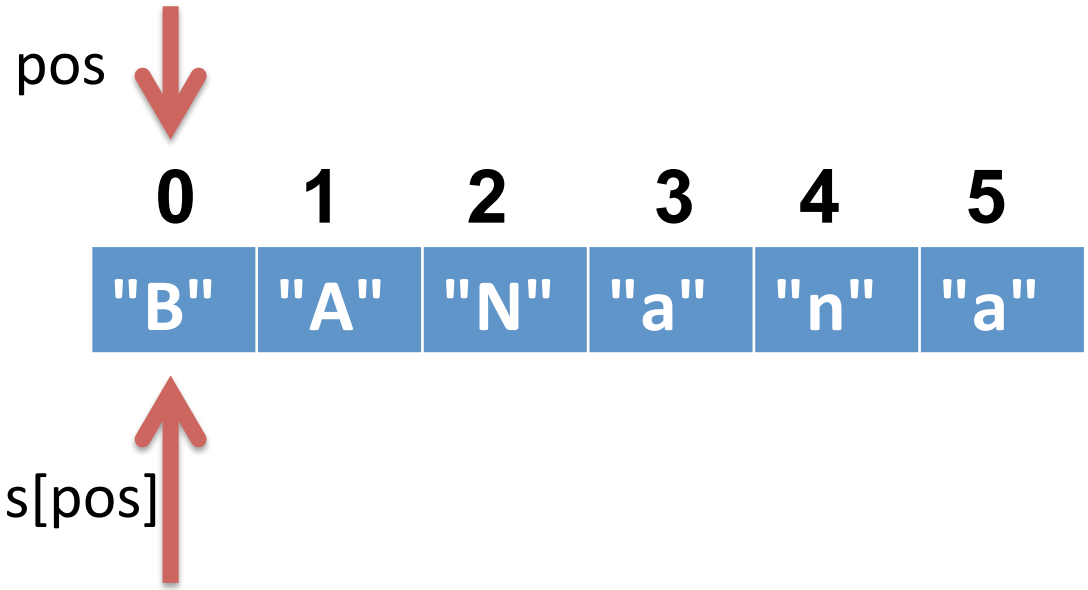
String concatenation in loops

- Basic string for loop can be used with string concatenation, too.
- Use this idea when we need to examine every character of a string and construct a new string based on certain characters.

```
def count_any_a(str):
  counter = 0
  for pos in range(0, len(str)):
    if str[pos] == "a" or str[pos] == "A":
      counter = counter + 1
  return counter

def filter_any_a(str):
  answer = ""
  for pos in range(0, len(str)):
    if str[pos] == "a" or str[pos] == "A":
      answer = answer + str[pos]
  return answer
```

```
str = "BANana"
for pos in range(0, len(str)):
    if str[pos] == "a" or str[pos] == "A":
      answer = answer + str[pos]
```



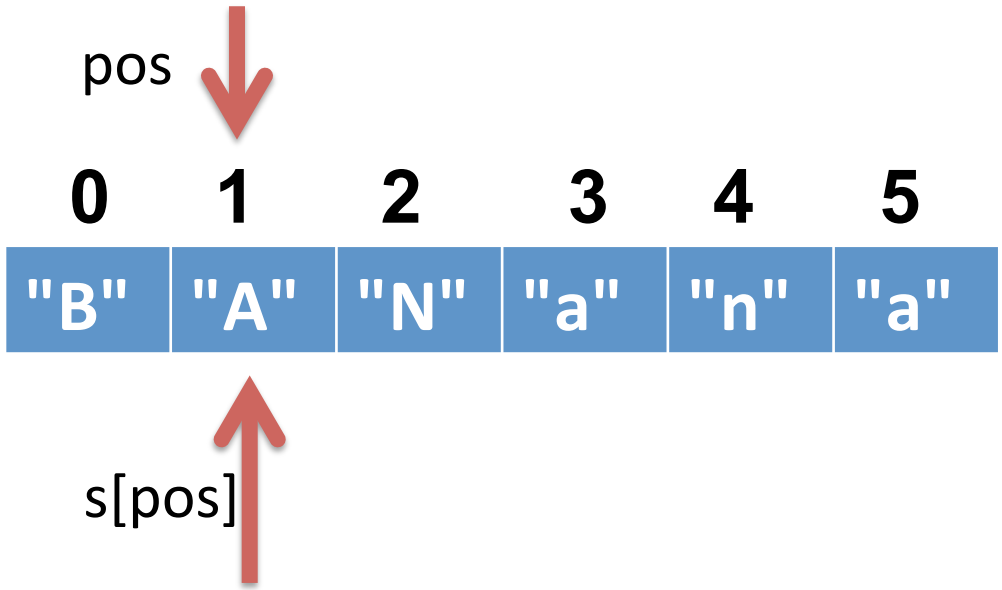
1st iteration

pos: 0

s[pos]: "B"

answer: ""

```
str = "BANana"
for pos in range(0, len(str)):
    if str[pos] == "a" or str[pos] == "A":
      answer = answer + str[pos]
```



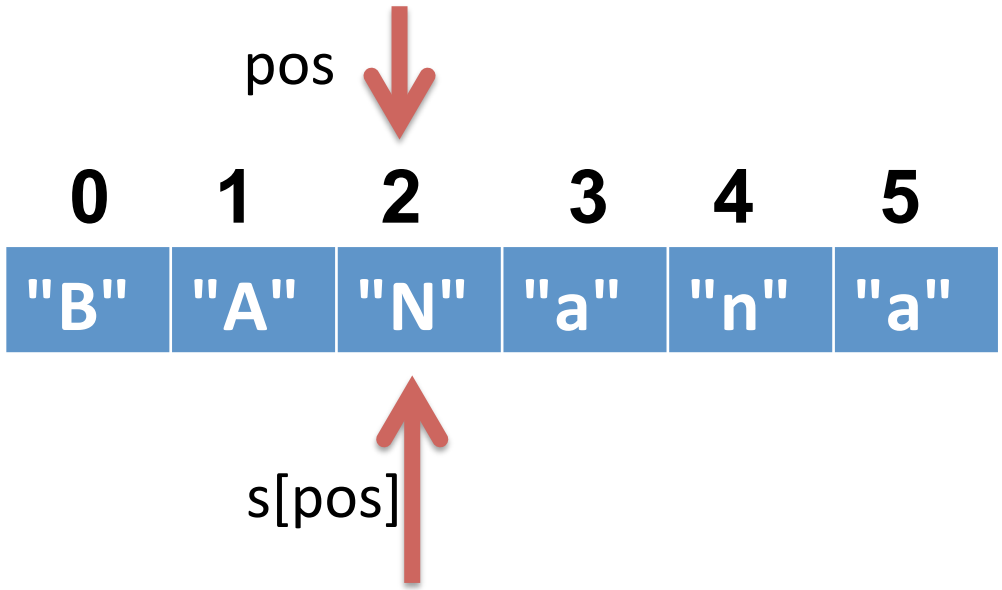
2nd iteration

pos: 1

s[pos]: "A"

answer: "A"

```
str = "BANana"
for pos in range(0, len(str)):
    if str[pos] == "a" or str[pos] == "A":
        answer = answer + str[pos]
```



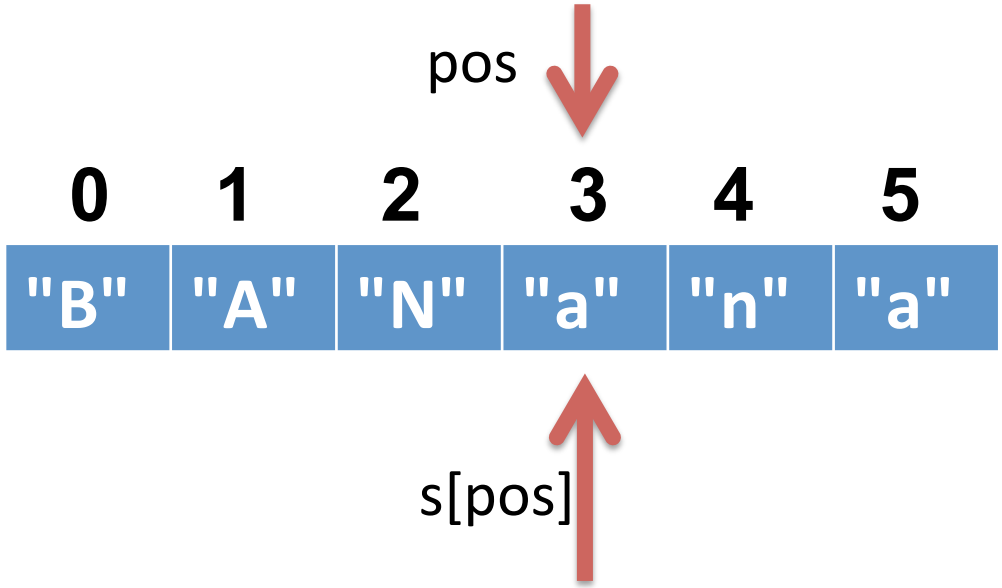
3rd iteration

pos: 2

s[pos]: "N"

answer: "A"

```
str = "BANana"
for pos in range(0, len(str)):
    if str[pos] == "a" or str[pos] == "A":
      answer = answer + str[pos]
```



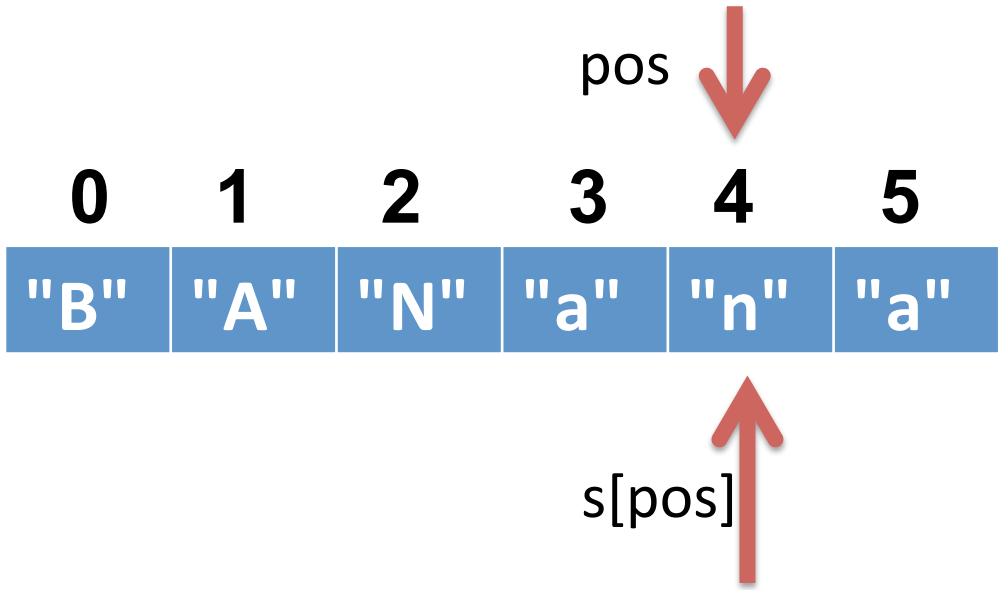
4th iteration

pos: 3

s[pos]: "a"

answer: "Aa"

```
str = "BANana"
for pos in range(0, len(str)):
    if str[pos] == "a" or str[pos] == "A":
        answer = answer + str[pos]
```



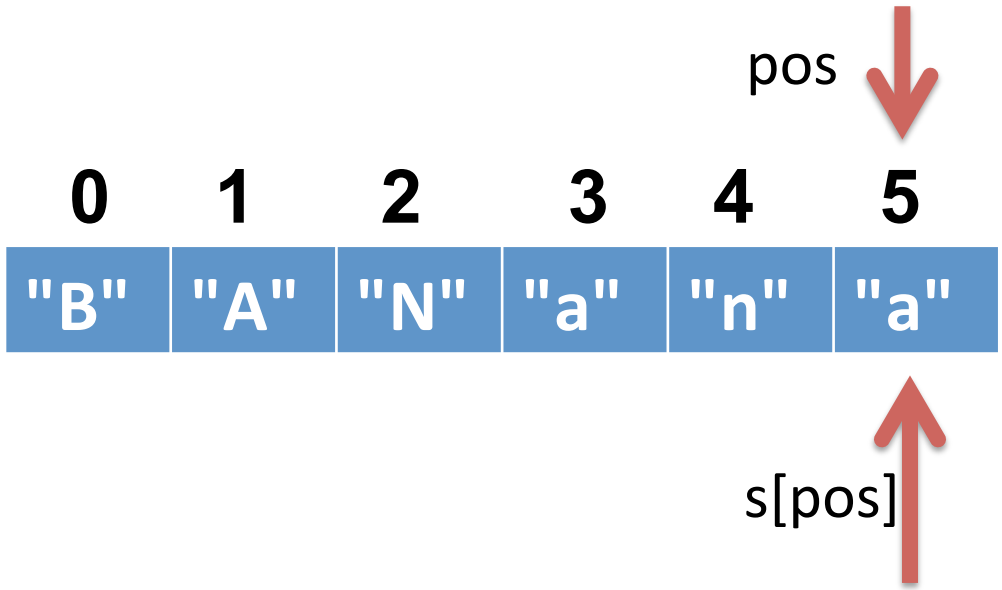
5th iteration

pos: 4

s[pos]: "n"

answer: "Aa"

```
str = "BANana"
for pos in range(0, len(str)):
    if str[pos] == "a" or str[pos] == "A":
      answer = answer + str[pos]
```



6th iteration

pos: 5

s[pos]: "a"

answer: "Aaa"

You try it

- Write a function called reverse that takes a string argument. It returns the string argument with all the characters in reverse order.
  - Example: reverse("abc") returns "cba"
- Write a function called every_other that takes a string argument. It returns a new string built from skipping every other character in the argument.
  - Example: every_other("abcdef") returns "ace"

```
def reverse(s):
  answer = ""
  for pos in range(len(s)-1, -1, -1):
    answer = answer + s[pos]
      counter = counter + 1
  return counter

def every_other(s):
  answer = ""
  for pos in range(0, len(s), 2):
    answer = answer + s[pos]
  return answer
```

Helpful string functions

- Handout has lots of useful string functions.
- Will be given to you on the test.

- Write a function called count_digits that counts the number of digits in a string.
  - Ex: count_digits("43abc8") returns 3.
  - Hint: use isdigit() from the handout.
- Write a function digit_sum that returns the total sum of the digits in a string.
  - Ex: digit_sum("43abc8") returns 15.
- Write a function total_time that takes a string argument with two numbers separated by a colon. The two numbers represent minutes and seconds. The function should return the total number of seconds in the time given. **The two numbers may be any number of digits. Hint: use the find function to find the colon.**
  - Ex: total_time("1:40") returns 100
  - Ex: total_time("10:40") returns 640
  - Ex: total_time("123:456") returns 7836