Strings III

Indexing/Slicing/Length

- If s is a string variable,
- s[p] returns character at index p.
- s[p:q] returns slice from characters p to q-1.
- len(s) returns the length of s (number of characters)

Write a function called total_seconds that takes one string argument. This argument will be a string of the form "M:SS" where M is a number of minutes (a single digit) and SS is a number of seconds (2 digits). This function should calculate the total number of seconds in this amount of time and return it as an integer. (Don't need a for loop!)

```
def total_seconds(time):
```

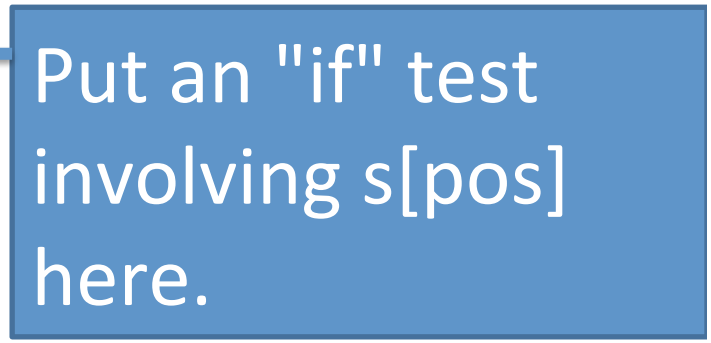
Basic for loop

- To do "something" with every character in a string s:

```
for pos in range(0, len(s), 1):
    # do something with s[pos]
```

Basic counting for loop

```
total = 0
for pos in range(0, len(s), 1):
    if ____________________:
        total = total + 1
```



Count the number of lowercase a's

```
total = 0
for pos in range(0, len(s), 1):
  if s[pos] == "a":
    total = total + 1
```

Count the number of any a's

```
total = 0
for pos in range(0, len(s), 1):
  if s[pos] == "a" or s[pos] == "A":
    total = total + 1
```

`s in t`	True if s is a substring in t
`s not in t`	False if s is a substring in t
`s.isalpha()`	True if s contains only letters
`s.isdigit()`	True if s contains only digits
`s.islower()`	True if s contains only lowercase letters
`s.isupper()`	True if s contains only uppercase letters
`s.isspace()`	True if s contains only whitespace.

Count the letters

```
total = 0
for pos in range(0, len(s), 1):
  if s[pos].isalpha()
    total = total + 1
```

Count the uppercase letters

```
total = 0
for pos in range(0, len(s), 1):
  if s[pos].isupper()
    total = total + 1
```

Count the vowels

```
total = 0
for pos in range(0, len(s), 1):
  if s[pos] in "aeiouAEIOU"
    total = total + 1
```

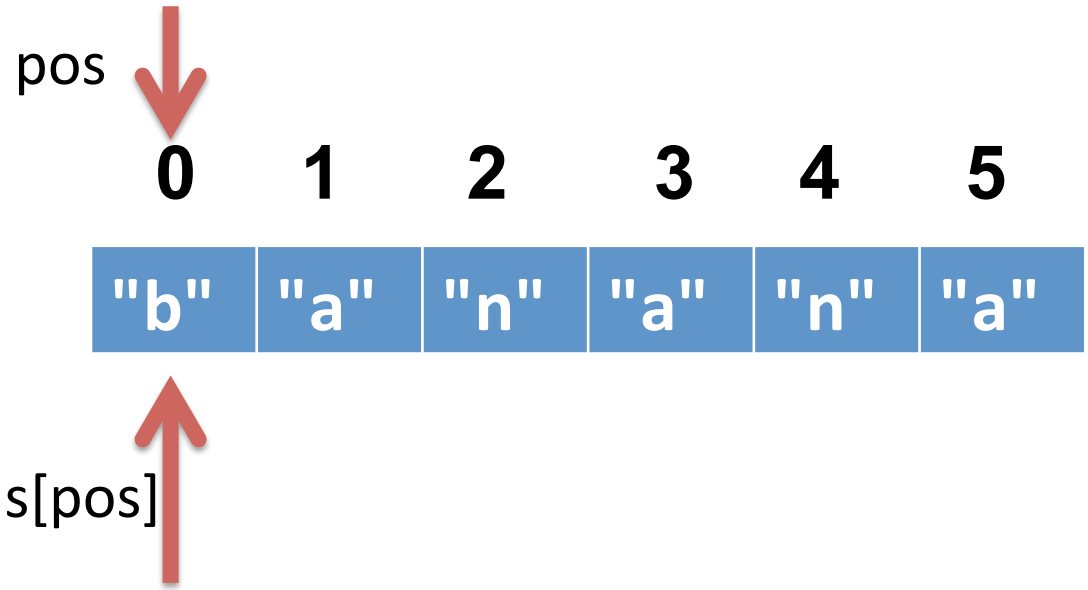
String concatenation

- Have string variables s and t:
- s + t gives you a new string with all the characters of s followed by all the characters of t.

What does this code do?

```
answer = ""
for pos in range(0, len(s), 1):
  answer = answer + s[pos]
```

```
s = "banana"
answer = ""
for pos in range(0, len(s), 1):
  answer = answer + s[pos]
```



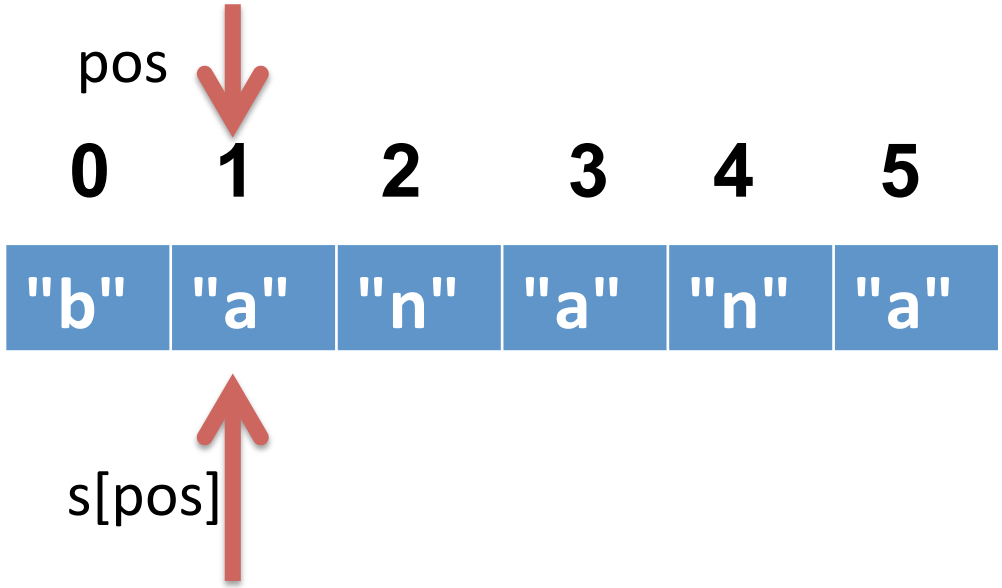
1st iteration

pos: 0

s[pos]: "b"

answer: "b"

```
s = "banana"
answer = ""
for pos in range(0, len(s), 1):
  answer = answer + s[pos]
```



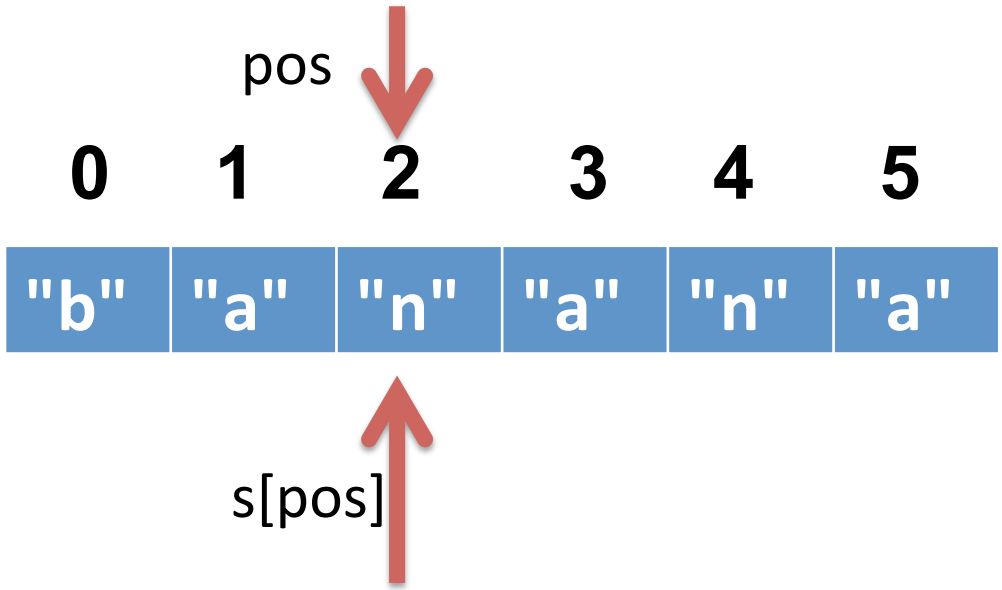
2nd iteration

pos: 1

s[pos]: "a"

answer: "ba"

```
s = "banana"
answer = ""
for pos in range(0, len(s), 1):
  answer = answer + s[pos]
```



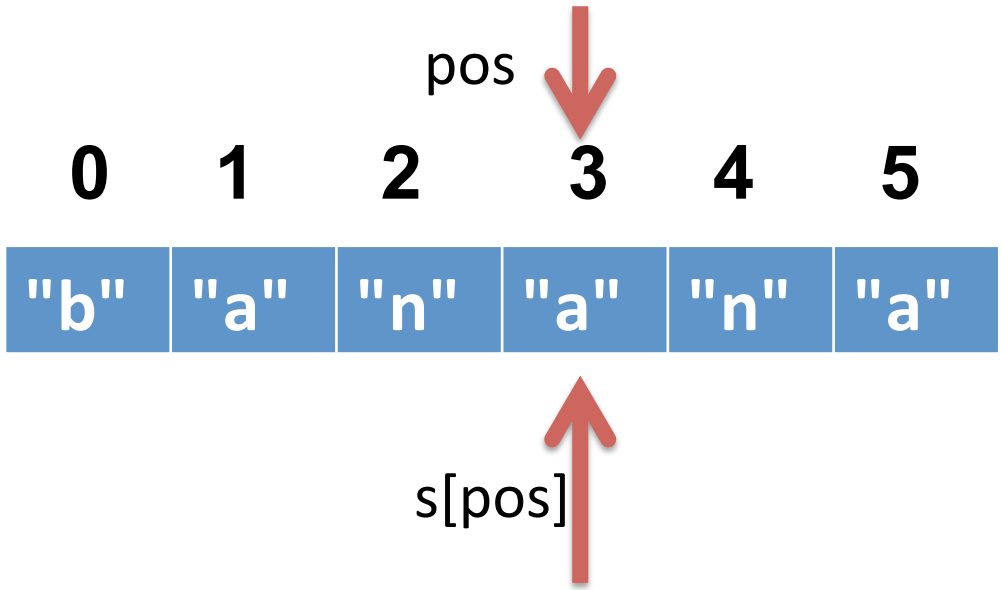
3rd iteration

pos: 2

s[pos]: "n"

answer: "ban"

```
s = "banana"
answer = ""
for pos in range(0, len(s), 1):
  answer = answer + s[pos]
```



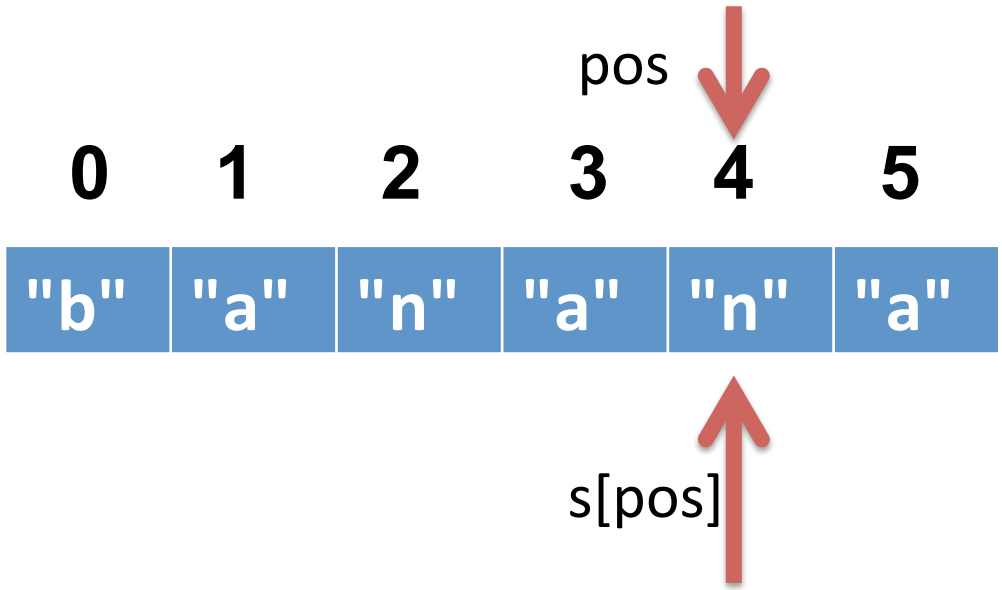
4th iteration

pos: 3

s[pos]: "a"

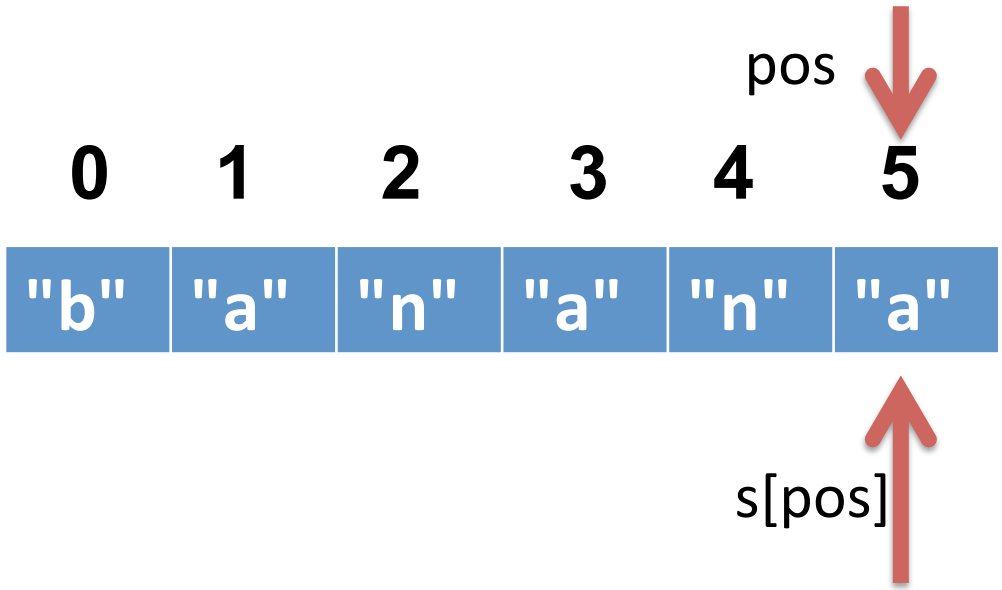
answer: "bana"

```
s = "banana"
answer = ""
for pos in range(0, len(s), 1):
  answer = answer + s[pos]
```



5th iteration
pos: 4
s[pos]: "n"
answer: "banan"

```
s = "banana"
answer = ""
for pos in range(0, len(s), 1):
  answer = answer + s[pos]
```



6th iteration

pos: 5

s[pos]: "a"

answer: "banana"

What does this do?

```
answer = ""
for pos in range(0, len(s), 1):
  if s[pos].isupper()
    answer = answer + s[pos]
```

```
total = 0
for pos in range(0, len(s), 1):
  if <test s[pos] for something>:
    total = total + 1
```

COUNT

```
answer = ""
for pos in range(0, len(s), 1):
  if <test s[pos] for something>
    answer = answer + s[pos]
```

FILTER

```
def some_counting_function(s):
  total = 0
  for pos in range(0, len(s), 1):
    if <test s[pos] for something>:
      total = total + 1
  return total
```

COUNT

```
def some_filtering_function(s):
  answer = ""
  for pos in range(0, len(s), 1):
    if <test s[pos] for something>
      answer = answer + s[pos]
  return answer
```

FILTER

- Write a function called count_digits that returns the number of digits in a string.
  - count_digits("abc123def5") returns 4
- Write a function called filter_digits that returns only the digits from a string.
  - filter_digits("abc123def5") returns "1235"
- Write a function called sum_digits that returns the sum of all the digits in a string.
  - sum_digits("abc123def5") returns 30

- Write a function called count_dups that counts the number of back-to-back duplicated characters in a string.
  - count_dups("balloon") returns 2.
- Write a function called count_unique that counts the number of unique characters in a string.
  - count_unique("abracadabra") returns 5.
- Write a function called reverse that RETURNS (not prints) the reverse of string s.
  - reverse("abc") returns "cba"