

# Strings III

Warmup: 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!)

*Hint: use the `int()` function to convert each component to an integer.*

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

Challenge: Make your function work with strings with a one- or two-digit minute component.

# Review: 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)

- Slices don't need both left and right indices.
- Missing left index:
  - Python assumes you meant 0 [far left of string]
- Missing right index:
  - Python assumes you meant `len(s)` [far right of string]

```
s = "Computer"  
print(s[1:])           # prints omputer  
print(s[:5])         # prints Compu  
print(s[-2:])       # prints er
```

# Indices don't have to be literal numbers

Say we have this code:

```
name = input("type in your name: ")  
x = int(len(name) / 2)  
print(name[0:x])
```

What does this print?

# Basic for loop

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

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

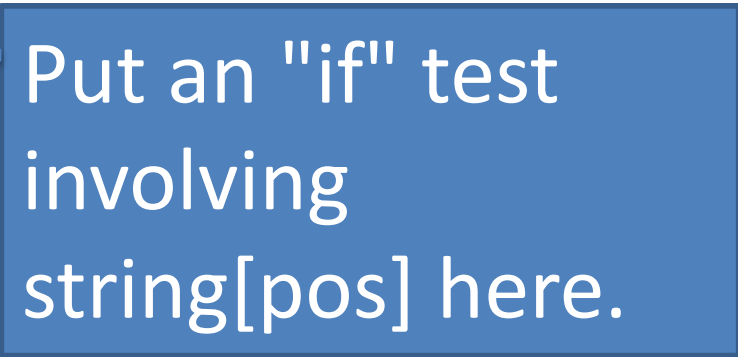
# Basic counting for loop

```
total = 0
```

```
for pos in range(0, len(string)):
```

```
    if _____:
```

```
        total = total + 1
```



Put an "if" test involving string[pos] here.

# Count the number of lowercase a's

```
total = 0
```

```
for pos in range(0, len(string)):
```

```
    if string[pos] == "a":
```

```
        total = total + 1
```



# Count the number of any a's

```
total = 0
```

```
for pos in range(0, len(string)):
```

```
    if string[pos]=="a" or string[pos]=="A":
```

```
        total = total + 1
```

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

# Count the letters

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

# Count the uppercase letters

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

# Count the vowels

```
total = 0
for pos in range(0, len(s)):
    if string[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`.
- `s` and `t` do not change!
  - Just like if you say `x = y + z`, where all your variables are integers, `y` and `z` don't change.

## Pig Latin Translator:

1. Get `piglatin_start.py` and `dickens.txt` from the class website (see "programs written in class")
2. Write `first_vowel`. This function should take a word and return the position/index of the first vowel in that word. **Test your function from the Python shell.**
3. Write `piglatin`. This function should translate a word into pig latin. Test your function by calling `main()`.
4. Fill in the code for `main2()`. This function reads an entire file and translates it into pig latin.
5. Challenge: Change your `piglatin` function so it handles words that end with punctuation.

# What does this code do?

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

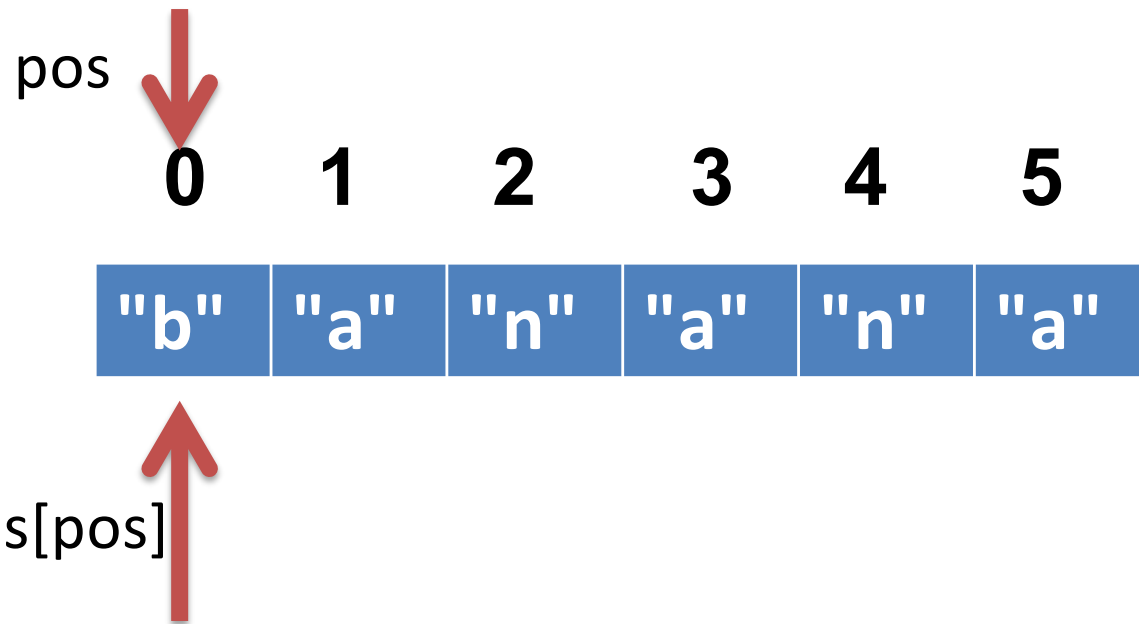


```
s = "banana"
```

```
answer = ""
```

```
for pos in range(0, len(s)):
```

```
    answer = answer + s[pos]
```



**1st iteration**

pos: 0

s[pos]: "b"

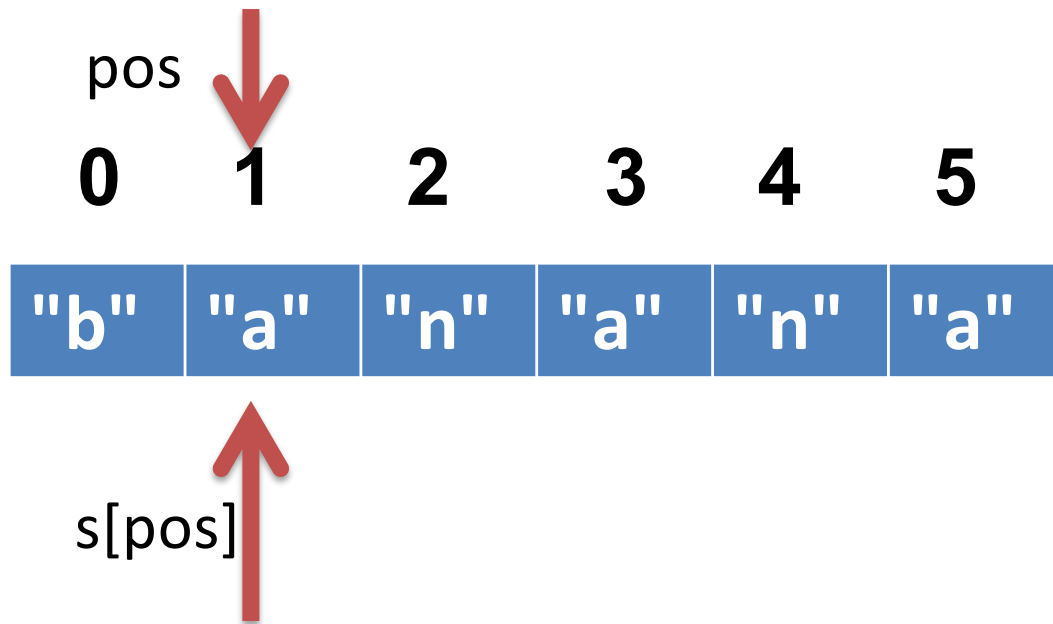
answer: "b"

```
s = "banana"
```

```
answer = ""
```

```
for pos in range(0, len(s)):
```

```
    answer = answer + s[pos]
```



2<sup>nd</sup> iteration

pos: 1

s[pos]: "a"

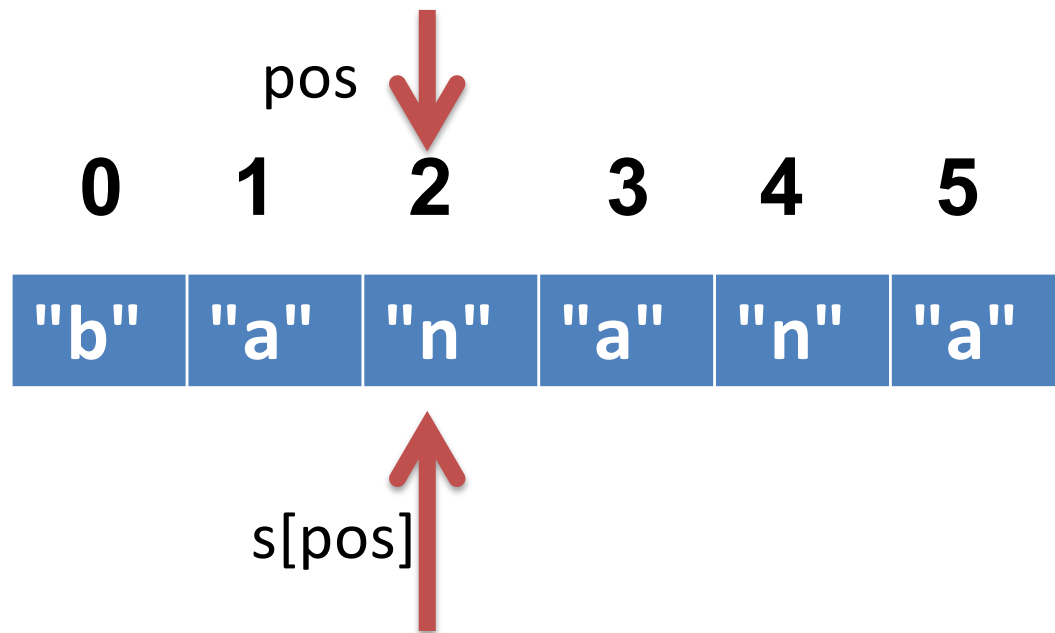
answer: "ba"

```
s = "banana"
```

```
answer = ""
```

```
for pos in range(0, len(s)):
```

```
    answer = answer + s[pos]
```



3<sup>rd</sup> iteration

pos: 2

s[pos]: "n"

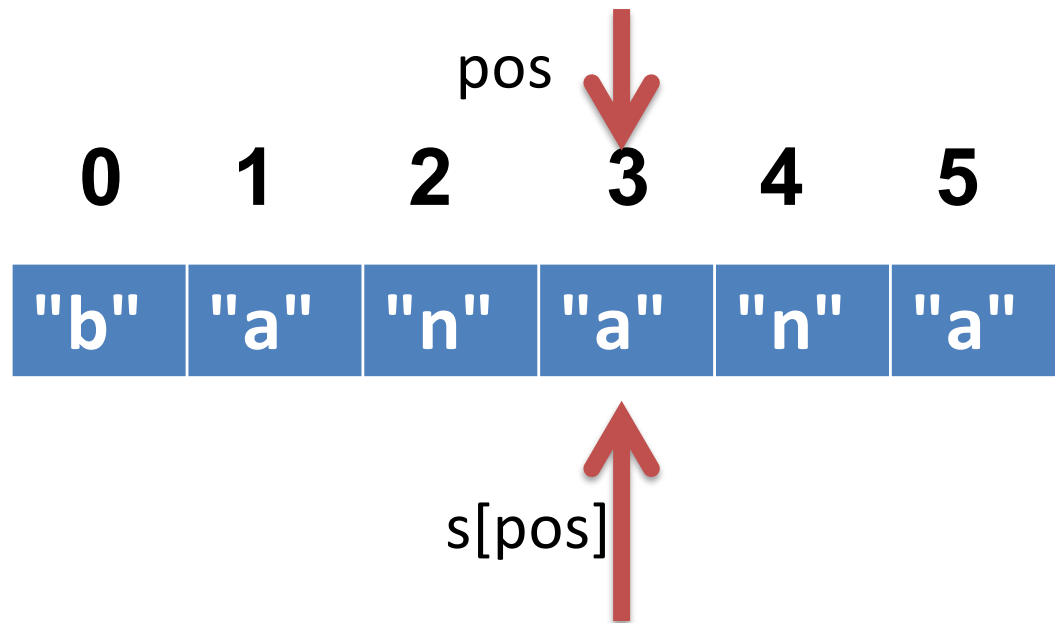
answer: "ban"

```
s = "banana"
```

```
answer = ""
```

```
for pos in range(0, len(s)):
```

```
    answer = answer + s[pos]
```



4<sup>th</sup> iteration

pos: 3

s[pos]: "a"

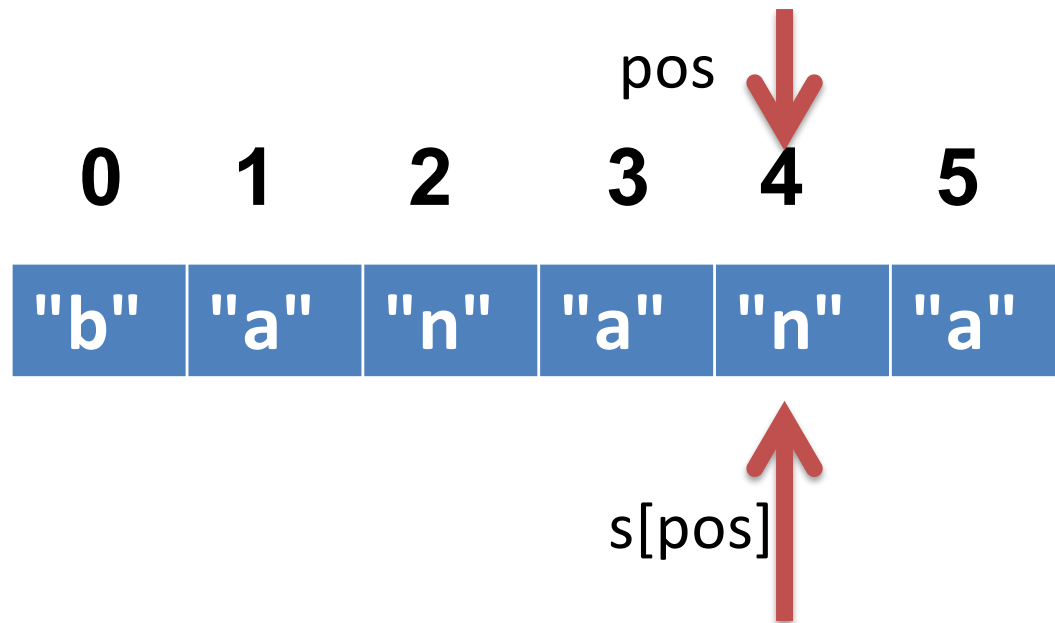
answer: "bana"

```
s = "banana"
```

```
answer = ""
```

```
for pos in range(0, len(s)):
```

```
    answer = answer + s[pos]
```



5<sup>th</sup> iteration

pos: 4

s[pos]: "n"

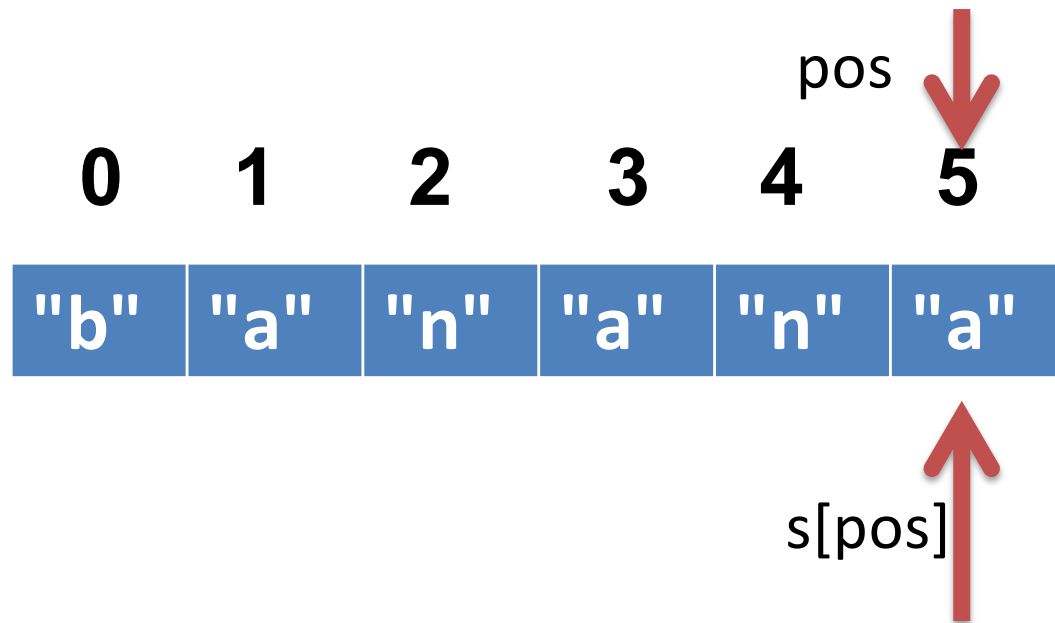
answer: "banan"

```
s = "banana"
```

```
answer = ""
```

```
for pos in range(0, len(s)):
```

```
    answer = answer + s[pos]
```



6<sup>th</sup> iteration

pos: 5

s[pos]: "a"

answer: "banana"

# What does this do?

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

```
total = 0
```

COUNT

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

```
answer = ""
```

FILTER

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



```
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