Strings

Membership

With `in`, we can test whether or not a string is a substring of another:

Membership

With `in`, we can test whether or not a string is a substring of another:

Example.

```
>>> 'i' in 'trinity'
```


Membership

With `in`, we can test whether or not a string is a substring of another:

Example.

```
>>> 'i' in 'trinity'
True
```

Membership

With `in`, we can test whether or not a string is a substring of another:

Example.

```
>>> 'i' in 'trinity'
True
>>> 'k' in 'trinity'
False
```


Membership

With `in`, we can test whether or not a string is a substring of another:

Example.

```
>>> 'i' in 'trinity'
True
>>> 'k' in 'trinity'
```


Membership

With `in`, we can test whether or not a string is a substring of another:

Example.

```
>>> 'init' in 'trinity'
True
```


Membership

With `in`, we can test whether or not a string is a substring of another:

Example.

```
>>> 'init' in 'trinity'
```

Membership

With `in`, we can test whether or not a string is a substring of another:

Example.

```
>>> 'init' in 'trinity'
True
>>> 'int' in 'trinity'
False
```


Membership

With `in`, we can test whether or not a string is a substring of another:

Example.

```
>>> 'init' in 'trinity'
True
>>> 'int' in 'trinity'
```


`for` loop

Example.

```
def in_both(word1, word2):
  for letter in word1
    if letter in word2
      print letter,
```

This function prints all letters that appear in two given words.

`for` loop

Example.

```
def in_both(word1, word2):
  for letter in word1
    if letter in word2
      print letter,
```

for loop

Example.

```
def in_both(word1, word2):
  for letter in word1
    if letter in word2
      print letter,
```

```
>>> in_both('trinity', 'find')
i n i
```


for loop

Example.

```
def in_both(word1, word2):
  for letter in word1
    if letter in word2
      print letter,
```

```
>>> in_both('find', 'trinity')
i n
```


for loop

Example.

```
def in_both(word1, word2):
  for letter in word1
    if letter in word2
      print letter,
```

```
>>> in_both('trinity', 'find')
```


for loop

Example.

```
def in_both(word1, word2):
  for letter in word1
    if letter in word2
      print letter,
```

```
>>> in_both('find', 'trinity')
```

The Search Pattern

Modify find to have 3rd parameter for starting index.

```
def find(word, letter):
    index = 0
    while index < len(word):
        if word[index] == letter:
            return index
        index = index + 1
    return -1
```


The Search Pattern

Example.

```
def find(word, letter):
    index = 0
    while index < len(word):
        if word[index] == letter:
            return index
        index = index + 1
    return -1
```


The Counter Pattern

Example.

```
word = 'banana'
count = 0
for letter in word:
    if letter == 'a':
        count = count + 1
print count
```


The Search Pattern

Modify find to have 3rd parameter for starting index.

```
def find(word, letter):
    index = 0
    while index < len(word):
        if word[index] == letter:
            return index
        index = index + 1
    return -1
```

String Methods

- A ***method*** is similar to a function—it takes arguments and returns a value—but the syntax is different:

```
>>> word = 'banana'
>>> new_word = word.upper()
>>> print new_word
BANANA
```

Comparing Strings

- The relational operators work on strings.
- In Python uppercase 'Z' comes before lowercase 'a'

```
>>> 'A' < 'a'
True
>>> 'Z' < 'a'
True
>>> 'A' == 'a'
False
```

The Counter Pattern Function

Example.

```
def count(word, ch)
    count = 0
    for letter in word:
        if letter == ch:
            count = count + 1
    return count
```

String Methods

- See: http://docs.python.org/release/2.5.2/lib/string-methods.html

```
>>> word = 'banana'
>>> index = word.find('a')
>>> print index
1
>>> word.find('na')
2
>>> word.find('na', 3)
4
```

In-class Example

- Write a function that converts a word input into Pig Latin:
- story --> ory + st + ay
- alpha --> alpha + yay
- I --> i + yay
- bring --> ing+ br + ay
- any --> any + ay