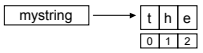
Strings

Strings

- A string is a series of *characters*
- Characters can be referenced by using brackets
- The first character is at position 0

mystring = “the”
letter = mystring[2] #letter becomes ‘e’



length

- The len function returns the length of a string

```
mystring=“bob”
len(mystring) #3
len(“adam”) #4
length=len(mystring)
last = mystring[len-1] #retrieves last char
```

for loops

```
mystring = "CS is cool!"
for c in mystring:
        print c

index=0
while index < len(mystring):
   print mystring[index]
   index += 1
```

Exercises

1. Write a for or while loop to print a string backwards

Slices

- Select a segment of a string
- Specify [start:end]
  - include start but do not include end
  - if you do not specify start slice starts from the beginning
  - if you do not specify end slices goes to end

```
mystring=“CS is cool”
print mystring[6:10]
print mystring[2:7]
print mystring[:4]
print mystring[:]
```

String Comparison/in

- == tests to see if strings are the same
- >, < compares strings alphabetically
- The `in` operator tests whether a given character appears in a given string
  - 'c' in "chocolate" #true
  - 'z' in "chocolate" #false

Immutability

- Strings are immutable
  - they cannot be changed

string module

- Contains useful methods for strings
  http://docs.python.org/lib/string-methods.html
- Dot notation allows us to call a method on a string object

```
import string
mystring="adam"
string.find(mystring, "a") #returns index of first instance found
mystring="CS is cool"
mystring.split() #result ['CS','is','cool']
newstring = mystring.replace("CS", "Econ")
```

Exercises

1. Write a program that prompts the user for two strings and determines the number of two-character sequences that appear in both the first and second strings. Exclude spaces in your comparison.
   - "CS is cool" "the old cow" would have two matches "co" for "cool" and "cow" and "ol" for "cool" and "old"