Introduction to Computer Programming
Section 8
Lecture 21
More Strings and Lists
Final Exam

May 20, 2:00-3:50  CIWW 317

Early Make-up for Graduating Seniors
(You must get my permission if you want this option.)

May 13  2:00-3:50  Room RBD
Homework

• All homework is required – LAST DAY 5/8
• Missing homework will hurt your grade
  – Don’t fall into a black hole
    • See a tutor
    • Make an appointment with me
• Check Classes to make sure all of your homework grades are posted.
  – Should be up to date
  – Late homework --- cc: es180@nyu.edu
String Slicing

- **Slice**: span of characters in a string, also called a *substring*

  ```
  mystring[start : end]
  ```

  - Evaluates to a string containing a copy of the characters from `start` up to, but not including, `end`

  - If `start` not specified, 0 is used for start index

  - If `end` not specified, `len(string)` is used for end index

  - Out-of-bounds does not generate an exception!
String Slicing Examples

name = “Alice Waters”
name[6 : 12]
name[6 : len(name)]
name[ : 5]
name[ : ]
name[1:3]
String Slicing Examples

name = "Alice Waters"

name[6 : 12]  # "Waters"

name[6 : len(name)]

name[ : 5]

name[ : ]

name[1:3]
String Slicing Examples

name = “Alice Waters”

name[6 : 12]  # “Waters”
name[6 : len(name)]  # “Waters”
name[ : 5]
name[ : ]
name[1:3]
String Slicing Examples

```python
name = "Alice Waters"

name[6 : 12]     # "Waters"
name[6 : len(name)]  # "Waters"
name[ : 5]       # "Alice"
name[ : ]        # "Alice"
name[1:3]
```
String Slicing Examples

name = “Alice Waters”

name[6 : 12]  # “Waters”
name[6 : len(name)]  # “Waters”
name[ : 5]  # “Alice”
name[ : ]  # “Alice Waters”
name[1:3]
String Slicing Examples

name = “Alice Waters”

name[6 : 12]          # “Waters”
name[6 : len(name)]   # “Waters”
name[ : 5]            # “Alice”
name[ : ]             # “Alice Waters”
name[1:3]             # “li”
String Methods

mynam = “Mary”
Yourname = “Jill”

str.upper(mynam)  # “MARY”
str.upper(yourname)  # “JILL”

Can also be written as a method of the variables myname and yourname:

mynam.upper()  # “MARY”
yourname.upper()  # “JILL”
String methods that are old friends...

word = “some string”
word.upper()
word.lower()
word.isalpha()
word.isalnum()
word.isdigit()
New Methods We Saw Last Time..

# tests whether word ends with “ing”
If word.endswith(“ing”) :
    ....

# returns string with “his” replaced by “her”
sentence = sentence.replace(“his”, “her”)

# returns the index of substring “his”
index = sentence.find(“his”)

What is an index?

The term *index* means position

```python
>> sentence = "This is a sentence."
>> index = sentence.find("is")
>> index
2
```
Lists (Chapter 8 in Gaddis)

• A list is a sequence of values

• Lists are like strings in some ways
  A string is a sequence of characters

• Elements of lists can be anything
  Integers, strings, floating point numbers, booleans

• Lists can be nested
  Lists can contain lists, which can contain lists
Creating Lists

colors = ["Red", "Blue", "Green"]

numbers = [456, 42, -11, 45.7]

flags = [True, False, True, True]

mixed_stuff = ["Red", 5, False, "Purple"]
Creating Lists

• Empty List

emptylist = []

• Nested List

quant=[[“shirt”, 23], [“dress”, 55], [“hat”, 15]]
Fruit Loop

fruits=['apple', 'grape', 'cherry', 'kiwi']

# We can traverse a list stored in a variable
for fruit in fruits:
    print(fruit)
Other built-ins and operations

- **Length of a list**
  
  ```python>> len(["apple", "pear", "banana"]))
3
```

- **List Concatenation**
  
  ```python>> [\'apple', 'pear'] + [\'cherry', 'kiwi' ]
[\'apple', 'pear', 'cherry', 'kiwi' ]
```

- **List Replication**
  
  ```python>> [\'a', 'b', 'c'] * 3
[\'a', 'b', 'c', 'a', 'b', 'c', 'a', 'b', 'c']
```
Membership Test (in, not in)

fruits=['apple', 'grape', 'cherry', 'kiwi']

if 'apple' in fruits :
    print("apple is a fruit")

if 'mushroom' not in fruits :
    print("mushroom is not a fruit")
Something new: sum, min, max

```python
>> numbers = [34, 23, 5]  # only for numbers
>> sum(numbers)
62
>> min(numbers)  # strings too!
5
>> max(numbers)  # strings too!
34
```
Fruit Loop 2

fruits=[‘apple’, ‘grape’, ‘cherry’, ‘kiwi’]

# We can traverse a list by iterating over indices
for index in range(len(fruits)):
    fruit = fruits[index]
    print(fruit)
Indexing Nested Lists

quant=[[‘shirt’, 23], [‘dress’, 55], [‘hat’, 15]]

>>quant[1]
[‘dress’, 55]

>>quant[1][1]
55
List Slicing

fruits=['apple', 'grape', 'cherry', 'kiwi']

>>fruits[0:2]
['apple', 'grape']

>>fruits[:2]
['apple', 'grape']

>>fruits[2:] ['cherry', 'kiwi']
Lists are *Mutable*

```python
>> word = 'Test'
>> word[2] = 'x'
```

Error! # Strings are not mutable

```python
>> letters = ['T', 'e', 's', 't']
>> letters[2] = 'x'
>> letters
['T', 'e', 'x', 't']
```
Modifying Slices

```python
>>> letters = ['T', 'e', 's', 't']
>>> letters[1:3] = ['o', 'o']
>>> letters
['T', 'o', 'o', 't']
```