

ECS 10

12/05

Announcements

- Final is Mon, Dec. 8, 8-10 AM
- Open book, open notes. Bring sample programs from class, your programs, etc.
- Bring a Scantron 2000, and a pencil.

Materials for Study

- On class Web page: example final, with three programming problems. Try doing them first on paper, then on the computer.
- Lecture slides, in-class programs. See especially midterm review sessions.
- On myUCDavis: Midterm 2 solutions, solutions to program 4 and 5, soon 6.
- Read the book and work through its examples, if you haven't yet.

Topics

- Everything from Midterm 1
 - Boolean variable and expressions
 - int(), float(), str()
- Everything from Midterm 2
 - string methods
 - for loops
 - dictionaries
 - Files
- Functions

Boolean Variables

```
x = 5
a = x>2
b = x<4
print (not a) or (not b)
```

- What does this program do?

Boolean Variables

```
x = 5
a = x>2
b = x<4
print (not a) or (not b)
```

- The value of a is True
- The value of b False
- The value of (not b) is (not False) is True
- So the program prints True

Dictionaries

```
D = {} # frequency of letters in English
D["e"] = 1
D["t"] = 2
D["a"] = 3
for i in range(3):
    print D[i+1]
```

- What does this program do?

Dictionaries

```
D = {} # frequency of letters in English
D["e"] = 1
D["t"] = 2
D["a"] = 3
for i in range(3):
    print D[i+1]
```

- It crashes, because D[1] does not exist.
- Keys here are letters, not integers.
- How could you do this correctly?

When to use a List

```
D = [] # frequency of letters in English
D[1] = "e"
D[2] = "t"
D[3] = "a"
for i in range(1,4):
    print D[i]
```

- Better to use a list, since we want to access the data using a small range of integer indices, in order.

Dictionaries

```
key = raw_input('Choose a letter: ')
if key in D:
    print D[key]
```

- Use this construct if you can't be sure the key is in the dictionary.
- Keys are letters, not integers.
- Indexing with a non-key causes a crash.

Dictionaries

```
D = {}
letters = 'abcdefg'
for i in range(len(letters)):
    D[letters[i]] = i
print D['c']
```

- What does this program do?

Dictionaries

```
D = {}
letters = 'abcdefg'
for i in range(len(letters)):
    D[letters[i]] = i
print D['c']
```

- Letters are keys, values are ints.
- range(len(letters)) is [0,1,2,3,4,5,6]
- D['a'] = 0, D['b'] = 1,...
- Prints 2

Lists of Lists

```
L = []
for i in range(3):
    L = L + [[i, i+1]]
```

- What is L after these lines?

Lists of Lists

```
L = []
for i in range(3):
    L = L + [[i, i+1]]
```

- `[i, i+1]` is a list of integers, with length 2
- `[[i, i+1]]` is a list of lists, with length 1
- L is `[[0, 1], [1, 2], [2, 3]]`
- L is a list of integers and lists, with length 3.

Sentry Variable

```
import random
rolling = True
while rolling:
    dice = random.randrange(6)
    if dice == 6:
        rolling = False
```

- What does this program do?

Sentry Variable

```
import random
rolling = True
while rolling:
    dice = random.randrange(6)
    if dice == 6:
        rolling = False
```

- Boolean variable used to keep track of what the program is doing.
- While loop will not end until sentry variable becomes false.
- Sadly, dice is always in the range 0...5.

Functions

```
def cleanString(s):
    out = s.replace(',', '')
```

```
inStr = '34,308'
cleanString(inStr)
x = int(out)
```

- What does this program do?

Functions

```
def cleanString(s):
    out = s.replace(',', '')
```

```
inStr = '34,308'
cleanString(inStr)
x = int(out)
```

- Crashes.
- The local variable `out` is still undefined in the main program.

Functions

```
def cleanString(s):
    s = s.replace(',', '')
```

```
inStr = '34,308'
cleanString(inStr)
x = int(inStr)
```

- How about this one?

Functions

```
def cleanString(s):
    s = s.replace(',', '')
```

```
inStr = '34,308'
cleanString(inStr)
x = int(inStr)
```

- Also crashes. Functions don't change their arguments. inStr is defined but cannot be converted.
- How do you write this program correctly?

Functions

```
def cleanString(s):
    s = s.replace(',', '')
    return s
```

```
inStr = '34,308'
newStr = cleanString(inStr)
x = int(newStr)
```

- Information is sent back to the main program by with the return value.
- In the main program, it has to be assigned to a variable (newStr).

Files

```
inFile = open('myFile.txt', 'r')
G = []
giftStr = inFile.readline()
while giftStr != "":
    giftStr = giftStr[:-1]
    G = [giftStr]
    giftStr = inFile.readline()
```

- What does this code do?

Files

```
inFile = open('myFile.txt', 'r')
G = []
giftStr = inFile.readline()
while giftStr != "":
    giftStr = giftStr[:-1]
    G = [giftStr]
    giftStr = inFile.readline()
```

- Typical file-reading while loop.
- File line is "" after end of file.
- Every line ends with a '\n' newline character
- G ends up being a list containing the last string in the file (without newline)

Files

```
inFile = open("myFile.txt", "r")
G = []
giftStr = inFile.readline()
while giftStr != "":
    giftStr = giftStr[:-1]
    G = G + [giftStr]
    giftStr = inFile.readline()
```

- In this version G becomes a list of strings (all without newlines)
- Another way to remove newlines?

String Methods

```
fruitStr = ' pineapple, pear\n'
str1 = fruitStr.strip()
fruits = str1.split(',')
for item in fruits:
    print item, len(item)
```

- What does this program print?

String Methods

```
fruitStr = ' pineapple, pear\n'
str1 = fruitStr.strip()
fruits = str1.split(',')
for item in fruits:
    print item, len(item)
```

- Prints:
pineapple 9
pear 5
- There is a space at the beginning of " pear"

Exceptions

```
def isFloat(s):
    try:
        float(s)      # Try to do the conversion
    except:
        return False # Conversion failed!
    else:
        return True  # Conversion succeeded
```

String Processing

```
string = ' Nov 16\tNASDAQ\t 2634.93\t+0.63%\n'
string = string.strip()
words = string.split('\t')
change = words[len(words)-1]
change = change.replace('%', '')
```