there are lots & lots of objects here!
Any other approach & you may "lose your mind..."!

Test early and often! (and keep backup copies of working files...)
Problem #1

def spamScore(S):
    if len(S) == 0:
        return 0
    else:
        return charScore(S[0]) + spamScore(S[1:])

similar to scrabble score...

Could also be done with a list comprehension...

LC = [charScore(c) for c in S]
return LC
def revCase(s):
    ''' revCase(s) should return a reverse-case version of s 
    input: s, a string 
    if s is: 
        return s 
    else: 
        return revChar(s[0]) + revCase(s[1:]) 
    '''

def revChar(c):
    ''' revChar(c) should return a reverse-case version of c 
    input: c, a single-character string 
    if 'a' == c: 
        return 'I' 
    elif 'a' == c: 
        return 'i' 
    else: 
        return c 

thanks to Callie for this answer...!
Problem #2

```python
def revCase(s):
    """ revCase(s) should return a reverse-cased version of s. 
    input: s, a string
    """
    if s == 'i':
        return s
    else:
        return revChar(s[0]) + revCase(s[1:])

def revChar(c):
    """ revChar(c) should return a reverse-cased version of c. 
    input: c, a single-character string.
    """
    if c == 'i':
        return 'I'
    elif c == 'I':
        c += 'need semi-colon
        return i'
    else:
        return c
```

thanks to Callie for this answer...!

and Gretchen for catching the number of eyes!
Problem #3

```
def fun( L ):
    """ takes in a list, L,
    counter = 0
    i = 0
    while L[i] == L[0]:
        counter += 1
        i += 1
    return counter
```

# of elements IN A ROW equal to the first element, L[0]

Extra (up to +3 pts) There are some inputs to the original `fun` function (The inputs are still valid lists.) For up to +2 extra points, what are some errors when input to `fun` (up to 2 of them)? For +1 more pt., what error when input to fun is an empty list? How about `fun( ["a", "b", "c"] )`?

empty list crashes!

So does a list with elements all equal...

thanks to Yok for this answer...!
Problem #4

```python
result = 0
for i in range(len(L)):
    if L[i] == i:
        result += 1
    elif L[i] == 42:
        result = 42
    elif L[i] == L[i-1]:
        result *= 2

return result
```

if \( L[i] == i \), add one

elif \( L[i] == 42 \), set to 42

elif \( L[i] == L[i-1] \), double it!

Part (A) What value is returned for the input list \( L = [5, 6, 7, 42] \)?

42

Part (B) What value is returned for the input list \( L = [0, 1, 2, 5] \)?

3

Part (C) What value is returned for the input list \( L = [0, 0, 1, 1] \)?

4

Part (D) What value is returned for the input list \( L = [42, 42, 5, 5] \)?

84

Extra (up to +3) What is the largest output possible for an input list \( L \)?

Also, what is an example of an input \( L \) which outputs that largest value?

\[ L = [42, 1, 1, 1] \]

largest output...!

thanks to Martin for these answers...!
Kate

Create a sketch, poem, or other artistic rendering that captures the essential and eternal tension among recursion, Picochet, three-eyed aliens, loops, Python (or pythons) and spam...

Yes!

None of them
Are missing
Out of the
Trash

And now, a haiku:
I have spam
Dont use
Death comes
Swat
For canned meat

My blade tastes of
Salt

Riley
Lucas: Recursive is stuck looking in the mirror and Pico-bot unlucky. Facing the wall, three-eyed aliens repeat "42, 42!" and Python's interpreter can't find a clue, and spam continues to misbehave. But the loop might finally go away and the tension falls one day. But hey! This is not on the group. The bugs are not them, they're you!

Gretchen: The internal tension is best shown by my own mental questioning of whether there are indeed 3-eyed alien hats or rather space bugs?

John: I wonder if we ever see Pluton if we zoom in on its sky.

Rory: The internal tension is best shown by my own mental questioning of whether there are indeed 3-eyed alien hats or rather space bugs?
def python():
    Count = 0
    H2 = "H2"
    if count == 0:
        print("_ _ _ ₹")
        count = 1
    if count == 1:
        print("_ _ ₹")
        print("_ ₹")
        count = 2
    if count == 2:
        print("_ ₹")
        print("_ ₹")
        print("_ ₹")
        count = 3
    if count == 3:
        print("_ ₹")
        print("_ ₹")
        print("_ ₹")
        print("_ ₹")
        count = 4
    if count == 4:
        print("_ ₹")
        print("_ ₹")
        print("_ ₹")
        print("_ ₹")
        print("_ ₹")
        count = 5
    # continued
Callie

Yok

recursion, Picobot, three-eyed aliens, loops, Python (or pynons):

I travel S in state 2 until I hit a while, then I switch to state 1 and travel E.

While I hold this picture, recursion occurs.

It's a midnight.
CS5's view from here...

Applications of CS beyond software

CS 5 Applications
- simulation + analysis/ algorithms
- graphics / media / games ...
- feature-based modeling + classification
- other state-machines + fun stuff

CS 5 Parts
- recursion
- variables
- loops
- functions
- circuits and binary
- data: classes and objects

Fundamental building blocks of CS

I've got all my eyes on this view!
Labs *next* week

These are *entirely optional* meeting times:

<table>
<thead>
<tr>
<th>Labs:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday 6/25</td>
<td>7-10 pm</td>
<td></td>
</tr>
<tr>
<td>Monday 6/26</td>
<td>7-10 pm</td>
<td></td>
</tr>
</tbody>
</table>

- to work on your *final projects*
- to work on hw 12's *finite-state machines*
CS5's view from here...

What's next?

Final Projects

- Picobot
- vPool
- TextID

CS 5 Applications
- simulation + analysis/algorithms
- graphics / media / games...
- feature-based modeling + classification
- other state-machines + fun stuff

CS 5 Parts
- recursion
- variables
- loops
- functions
- circuits and binary
- data: classes and objects

is under here:

CS Foundations

What can we compute... ... and how well?

CS Practice

CS Theory
CS5's view from here...

CS 5 ~ all corners of CS

CS Foundations

What can we compute... ... and how well?
The final week in CS5...

1. define "computer" precisely
2. define "compute" precisely
3. see what computers provably can't compute
4. go back to step (1) and define things better...
5. ... until time runs out...

CS Foundations What can we compute... ... and how well?
The final week in CS5...

things get *Strange* ...
Unifying idea: State

The state of a computation is all the internal information needed to take the next step.

for Picobot, next step is taken literally!
states as *subtasks*

State Machine:

each oval represents a different Picobot state

transitions move from state to state

starting funnel

the "go North" state

the "go South" state

---

<table>
<thead>
<tr>
<th>state</th>
<th>pattern</th>
<th>move</th>
<th>new state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x***</td>
<td>N</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>N***</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>***x</td>
<td>S</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>***S</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>
XKCD FSMs!
A model of computation:

Computer ~ a finite state machine

Example FSM

A set of states: s0, s1

Transitions:
- From s0 to s1 on input 1
- From s1 to s0 on input 1
- Self-loop on s0 on input 0
- Self-loop on s1 on input 0
A model of computation: FSM

Finite State Machine

Example FSM
A model of computation: FSM

Finite State Machine

Example input

input sequence 100101

This input's output:

start state “input funnel”

transitions “where to go” labeled by input!

accepting states double circled
A *model* of computation: **FSM**

*Finite State Machine*

**Example input**

Input sequence: 100101  
read left-to-right

Another accepted str:

Rejected example str:

---

Start state: "input funnel"

Transitions: "where to go"  
labeled by input!

Accepting states: double circled

This input's output:
Finite state machine

another input sequence always left-to-right

0010111

output for this input

State 0

transition on 0

cell

transition on 1

State 1

transition on 0

cell

transition on 1

cell

What does each state MEAN?
What does this FSM do overall?
JFLAP! graphical state-machine builder for the Ex. Cr.
JFLAP! Multiple-input runs ~ lots of tests provided!

empty string, or $\lambda$
What does each state say about the current state of the input?!?

Could you get the same behavior with fewer states? What's the minimum # possible? How do you know?

Extra! Hint: do strings have to be in separate states?

In general, what English phrase describes the rejected inputs?

This machine rejects strings that ...

List three different-length inputs that this FSM accepts:

List three different-length inputs that this FSM rejects:

Name(s) _____________________________

Extra! Could you get the same behavior with fewer states? What's the minimum # possible? How do you know?

Hint: do strings have to be in separate states?

What does each state say about the current state of the input?!?

s0 means ACCEPTING + ending in a 0 ... or ...

s1 means ACCEPTING + ending in a 1

s2 means REJECTING + ending in a 0 ... or ...

s3 means REJECTING + ending in a 1
What does each state say about the current state of the input?!?

Could you get the same behavior with fewer states?

What's the minimum # possible? How do you know?

List three different-length inputs that this FSM accepts:

List three different-length inputs that this FSM rejects:

In general, what English phrase describes the rejected inputs?

This machine rejects strings that ... have two 1's in a row (anywhere in the string)

What does each state say about the current state of the input?!

s0 means ZERO 1's in a row ... 
s1 means ONE 1 in a row...
s2 means TWO 1's in a row...

Extra! Could you get the same behavior with fewer states?

What's the minimum # possible? How do you know?

Hint: do strings have to be in separate states?
FSMs are everywhere!

Locks
FSMs are everywhere!

(penny, fifty cent piece, silver dollar, Canadian currency, CS 5 Euro, ….

mechanical vending machine

www.youtube.com/watch?v=85C4eh0mEJg @ 1:42 !
The state-machine that controls Quake's *Shambler* monsters...

**FSM ~ Game AI**

I'm *Quaking* in my AstroBoots
FSM ~ Game AI

Recognize this street?

Here, it's Ghost AI
All robots use FSM control

What states can you "factor out" from watching this towel-folding?
Towel-folding states!

Fig. 2. The state machine model of the procedure: dashed lines indicate failure recovery cases. The images show an actual run.
Build-your-own FSMs

Draw a FSM accepting strings with at least two 1s (anywhere). Others are rejected.

**Accepted examples:** 0101, 00010110, 111011, 11

**Rejected examples:** 0100, 1000, 000000, 1, 0

**Hint** - modify this starter FSM by adding labels, transitions, and one more state:

| 1 | 1 | 0 |

Draw a FSM that accepts strings that don't contain the pattern 110 anywhere.

**Accepted:** 1010001, 011

**Rejected:** 101001100, 01101

**Hint** - there are FIVE more transitions – but no more states - needed here

Draw a FSM accepting strings in which the third digit (3d from the left) is a 1.

**Accepted:** 1010001, 011

**Rejected:** 11000100, 11, 0

**Extra!** Draw a FSM accepting strings whose third-to-last digit (3d from the right) is a 1.

| 0100 and 01101 | 01001 and 11 |
**Build-your-own FSMs**

Draw a FSM accepting strings with at least two 1s (anywhere). Others are rejected.

**Accepted examples:** 0101, 00010110, 111011, 11

**Rejected examples:** 0100, 1000, 000000, 1, 0

**Hint:** there are FIVE more transitions – but no more states - needed here

---

Draw a FSM accepting strings in which the number of zeros (0s) is a multiple of 3, so there are 0, 3, 6, ... zeros. 1s don't matter!

**Accepted:** 110101110, 11, 0000010

**Rejected:** 101, 0000, 11101101111

**Hints:**
- Is never change the state!
- Another hint: make a triangle!

```python
if count('0', s)%3 == 0:
    return True
else:
    return False
```

---

Draw a FSM accepting strings in which the third digit (3d from the left) is a 1.

**Accepted:** 10

**Rejected:** 11, 000100, 11, 0

**Hints:**
- Modify this starter FSM by adding labels, transitions, and one more state:

```python
if s[2] == '1':
    return True
else:
    return False
```

---

Draw a FSM accepting strings whose third-to-last digit (3d from the right) is a 1.

**Accepted:** 010001, 011

**Rejected:** 10100100, 11, 0

**Hints:**
- What's the minimum number of states needed?
- Another hint: make a triangle!

```python
if s[-3] == '1':
    return True
else:
    return False
```

---

Extra! Draw a FSM accepting strings whose third-to-last digit (3d from the right) is a 1.

**Accepted:** 0100 and 01101

**Rejected:** 101001 and 11

---
**Build-your-own FSMs**

Draw a FSM accepting strings with at least two `1`s (anywhere). Others are rejected.

- **Accepted examples:** 0101, 00010110, 111011, 11
- **Rejected examples:** 0100, 1000, 000000, 1, 0

**Hint:** modify this starter FSM by adding labels, transitions, and one more state.

Draw a FSM that accepts strings that don't contain the pattern `110` anywhere.

- **Accepted:** 1010001, 011
- **Rejected:** 10100, 110, 0, 011

**Hint:** there are FIVE more transitions – but no more states - needed here.

Draw a FSM accepting strings in which the number of zeros (0s) is a multiple of 3, so there are 0, 3, 6, ... zeros. **1s don't matter!**

- **Accepted:** 11010110, 11, 0000010
- **Rejected:** 101, 0000, 11101101111

**Hint:** 1s never change the state!

Another hint: make a triangle!

What's the minimum number of states needed?

Draw a FSM accepting strings in which the third digit (3d from the left) is a `1`.

- **Accepted:** 1010001, 011
- **Rejected:** 1010010, 11, 0

What's the minimum number of states needed?

**Extra!** Draw a FSM accepting strings whose third-to-last digit (3d from the right) is a `1`.

- **Accepted:** 0100 and 01101
- **Rejected:** 101001 and 11
Has at least two 1s...?

Draw a FSM accepting strings with at least two 1s (anywhere). Others are rejected.

Accepted: 0101, 00010110, 111011, 11
Rejected: 0100, 1000, 000000, 1, 0

What do we need to complete this machine?
Number of 0s is div. by 3

Draw a FSM accepting strings in which the number of zeros (0s) is a multiple of 3, so there are 0, 3, 6, ... zeros. 1s don't matter.

Accepted: 110101110, 11, 0000010

Rejected: 101, 0000, 111011101111

Minimum number of states?

Combine two of these?
No occurrences of 110?

Draw a FSM accepting strings that do NOT anywhere contain the pattern 110

Accepted: 1010001, 0001011
Rejected: 101001100, 011001

Which transitions are still needed here?
Third character is a 1

Draw a FSM accepting strings in which the third digit (from the left) is a 1.

Accepted: 1010001 and 0110
Rejected: 11000100 and 11

Why must s1 and s2 be separate states?

Minimum number of states?
Third-to-last character is a 1?

Draw a FSM accepting strings whose third-to-last digit (from the right) is a 1.

Accepted: 0100 and 01101
Rejected: 101001 and 11

Minimum number of states?
Third-to-last character is a 1

I don't accept this solution!
Something's not right here: it's down-right arrowing!

Do we need 15 states?
Third-to-last character is a 1

8 states?

Are 8 states required?
All robots use FSM control
An autonomous vehicle's FSM

Fig. 9. Situational Interpreter State Transition Diagram. All modes are sub-modes of the system RUN mode (Fig 4(b)).
FSMs driving robots...

MIT's car, Talos
FSMs driving robots...

MIT's car, Talos - and its sensor suite
State-machine *limits*?

Are there *limits* to what FSMs can do?

they can't necessarily *drive safely*...

But are there any *binary-string problems* that FSMs can't solve?
Let's build a FSM that accepts strings with any # of 0s followed by the same # of 1s.
State-machine _limits_?

Let's build a FSM that accepts strings with _any_ # of 0s followed by the _same_ # of 1s.

- **accepted**
  - 000111
  - 0011
  - 01
  - λ

- **rejected**
  - 011
  - 001
  - 11100
  - 00110
Let's build a FSM that accepts strings with **any # of 0s** followed by the **same # of 1s**

You don't need three eyes to see some problems here!

FSMs "can't count"

- accepted
  - 000111
  - 0011
  - 01
  - $\lambda$

- rejected
  - 011
  - 001
  - 11100
  - 00110
State-machines are limited.

*FSMs can't count*

at least not arbitrarily high...

We need a **more powerful model** than FSMs...

*What do we need to add?*
Thursday: Turing Machines

Lab session tonight:
hw4pr3.py ~ 3d graphics!
VPython in the browser...

```python
GlowScript 2.5 VPython
# you need the line above for browser-based vpython

# the simplest possible vpython program:
box( color=vector(1.0,1.0,1.0) )

# try changing the color: the components are
# red, green, blue    each from 0.0 to 1.0

# then, add a second parameter:    size=vector(2.0,1.0,0.1)
#     the order of those three #s: Length, Height, Width

# then, a third parameter:    axis=vector(2,5,1)
#     the order of those three #s: x, y, z
```
Starting physics ~ "animation loop"

velocity is now: < 4.2, 0, 0 >
velocity is now: < 2.90026, 0, 3.32807 >
velocity is now: < 3.80093, 0, 3.34543 >
Try this week's lab problem!

vpython can be alien