CS 5: Putting loops to work...


[   26250,  5250,  1050,  210,  ?  ]

[  90123241791111,  93551622,  121074,  3111,  ?  ]

[  1,  11,  21,  1211,  111221,  ?  ]  

What's next?

I'm glad you asked!

Homework 8: due Mon., 10/31 by midnight
"Office" hrs. Fri! + lots of tutoring, LAC & ...
Midterm  11/3; review on the CS5 homepage  quizzes!
Final Exam: choice of  12/16 or 17 @ 7pm
Pop tarts > candy

Official CS5 snack comparison

But this should exist!

pop-tart recursion!
Next Thursday will be the CS 5 in-class midterm

Un-warnings:

worries? concerns? See me...
five problems, written
worth 1 hw assignment
score worries? Extra extra-credit in hw9 and beyond

Suggestions:

go over in-class exercises and hwk problems
create a page of notes, 2-sided is OK
consider small variations of the problems – and how they would change the solutions...

only 5 minutes? Try list comprehensions & LoL!

all quizzes so far? they're posted!
Mid-term feedback...

I would love to know any thoughts you have about CS5 thus far in the term. In particular, how you feel about the time and effort CS5 requires...

On average, how much time per week do you spend on CS5 *outside class + lab*?

How does CS5’s workload compare to other classes you're taking this term?

How would you judge the *pace* of CS5?

Circle your year: First-year Sophomore Junior Senior Other

Somewhere you'd keep about CS5 ...?

Something you'd *change about* / get rid of / add to CS5 ...

Other thoughts optional, but 142% welcome:

Later today
CS 5:  Putting loops to work...


[ 26250, 5250, 1050, 210, ? ]

[ 90123241791111, 93551622, 121074, 3111, ? ]

[ 1, 11, 21, 1211, 111221, ? ]

What's next?

Homework 8: due Mon., 10/31 by midnight
"Office" hrs. Fri! + lots of tutoring, LAC & ...
Midterm  11/3; review on the CS5 homepage quizzes!
Final Exam: choice of 12/16 or 17 @ 7pm
The *read it and weep* sequence

1
11
21
1211
111221
312211
13112221
...

```
str vs. int
```

When does the first 4 appear?

```
How fast do these terms grow?
```

Extra extra credit: in wk9!
In the limit, the length of the Nth term of the read-it-and-weep sequence is $(1.303577...)^N$. This base was found computationally by taking repeated ratios of term lengths... 

Growth determined empirically...
Growth determined analytically...

\[ \lambda = 1.30357726034296\ldots \]

"Conway's Constant" has an **analytic** definition!

It is the largest real root of this **71st-degree** polynomial!!


the 71 roots (complex plane)
Happy Oct 31!
def fac( N ):
    result = 1
    for x in range(1,N+1):
        result *= x
    return result

def fac( N ):
    if N == 1:
        return 1
    else:
        return N*fac(N-1)

Is one more reasonable than the other?

Loops

Basic design strategies

Recursion

theoretical?

empirical?
Loops

Design strategy: look for repetition + describe it...

Recursion

Design strategy: look for self-similarity + describe it...

Is one more reasonable than the other?

def fac(N):
    if N == 1:
        return 1
    else:
        return N*fac(N-1)
for: two types

\[ L = [3, 15, 17, 7] \]

```
for x in L:
    print x
```

"deceptively easy"
for: two types

$L = [3, 15, 17, 7]$

for $x$ in $L$:
    print $x$

index-based loops

for $i$ in range(len($L$))
    print $L[i]$

for $x$ in $L$:
    print $x$

element-based loops
def sum(L):
    total = 0
    for x in L:
        total += x
    return total

def sum(L):
    total = 0
    for i in range(len(L))
        total += L[i]
    return total

elements vs. indices

L = [3, 15, 17, 7]

$x$ 0 1 2 3

$element$-based loops

$index$-based loops
hw8pr3:  T. T. Securities (TTS)

Analyzes a sequence of stock prices

\[
L = [40, 80, 10, 30, 27, 52, 5, 15]
\]

Implement a (text) menu:

(0) Input a new list
(1) Print the current list
(2) Find the average price
(3) Find the standard deviation
(4) Find the min and its day
(5) Find the max and its day
(6) Your TTS investment plan
(9) Quit

Enter your choice:
User input...

```python
meters = input('How many m? ')
cm = meters * 100
print('That is', cm, 'cm.')
```

What will Python think?

I think I like these units better than light years per year!
User input...

```python
meters = input('How many m? ')
cm = meters * 100
print('That is', cm, 'cm.')
```

I think I like these units better than light years per year!

What will Python think?
Fix #1: **convert** to the right type

```python
m_str = input('How many m? ')
meters = float(m_str)

cm = meters * 100
print('That is', cm, 'cm. ')
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>m_str</td>
<td>string</td>
<td>'42'</td>
</tr>
<tr>
<td>meters</td>
<td>float</td>
<td>42.0</td>
</tr>
<tr>
<td>cm</td>
<td>float</td>
<td>4200.0</td>
</tr>
</tbody>
</table>

... but **crash-able**
Fix #2: convert and check

```python
m_str = input('How many m? ')

try:
    meters = float(m_str)
except:
    print("What? Does not compute!"
print("Setting meters = 42")
    meters = 42.0

print('That is', cm, 'cm.')
```
Fix #2:

These errors are called **exceptions**. This is **exception handling**.

```python
try:
    meters = float(m_str)
except:
    print("What? Does not compute!")
    print("Setting meters = 42")
meters = 42.0

cm = meters * 100
print('That is', cm, 'cm.')
```

try-except lets you try code and – if it crashes – catch an error and handle it
Fix #3: **eval** executes Python code!

```python
m_str = input('How many m? ')

meters = eval(m_str)

cm = meters * 100
print('That is', cm, 'cm.')
```
Fix #3: **eval** executes Python code!

```python
m_str = input('How many m? ')

try:
    meters = eval( m_str )
except:
    print("What? Does not compute!")
    print("Setting meters = 42")
meters = 42.0

cm = meters * 100
print('That is', cm, 'cm.')
```

What could go wrong here?
A larger application

def menu():
    """ prints our menu of options """
    print("(0) Continue")
    print("(1) Enter a new list")
    print("(2) Predict")
    print("(9) Break (quit)")

def main():
    """ handles user input for our menu """
    while True:
        menu()
        uc = input('Which option? ')
        try:
            uc = int(uc)  # was it an int?
        except:
            continue  # back to the top!

Calls a helper function

Perhaps uc the reason for this?
def main():
    """ handles user input for our menu """
    L = [30, 10, 20]  # a starting list

    while True:
        menu()  # print menu
        uc = input('Which option? ') ...

        if uc == 9:
            (9) Quit

        elif uc == 0:
            (0) Continue

        elif uc == 1:
            (1) Get new list

        elif uc == 2:
            (2) Predict !  ... and so on ...

        ... and so on ...
def main():
    """ handles user input for our menu """
    L = [30,10,20]  # a starting list

    while True:
        menu()  # print menu
        uc = input('Which option? ')

        if uc == 9:
            break  # break jumps out of the loop

        elif uc == 0:
            continue  # continue jumps back to the top

        elif uc == 1:
            ... input ... eval ...

        elif uc == 2:
            ... and so on ...

(9) Quit
(0) Continue
(1) Get new list
(2) Predict!
other functions as needed...
# example looping program with user-input

def menu():
    """ a function that simply prints the menu ""
    print()
    print("(0) Continue!")
    print("(1) Enter a new list")
    print("(2) Predict the next element")
    print("(9) Break! (quit)")
    print()

def main():
    """ the main user loop """
    print("+++++++++++++++++++++++++")
    print("Welcome to the PREDICTOR!")
    print("+++++++++++++++++++++++++")
    print()

    menu()

    secret_value = 4.2
    L = [30,10,20] # an initial list

    while True:   # the user-interaction loop
        print("\nThe list is", L)
        menu()
        uc = input( "Choose an option: " )
        if uc == 0:   # we want to continue...
            continue   # goes back to the top of the while loop
        elif uc == 1: # we want to enter a new list
            newL = input("Enter a new list: ")
            L = eval(newL)
            # "clean and check" the user's input
            try:
                newL = eval(newL) # eval runs Python's interpreter! Note: Danger
                if type(newL) != type([]):
                    print("That didn't seem like a list. Not changing L.")
                    L = newL # here, things were OK, so let's set our list, L
                else:
                    print("I didn't understand your input. Not changing L.")
            except:
                L = newL # # other menu option!
                pass

        elif uc == 3: # unannounced menu option!
            pass # this is the "nop" (do-nothing) statement in Python

        elif uc == 4: # unannounced menu option (slightly more interesting...)
            m = find_min(L)
            print("The minimum value in L is", m)

        elif uc == 5: # another unannounced menu option (even more interesting...)
            minval, minloc = find_min_loc(L)
            print("The minimum value in L is", minval, "at day ", minloc)

        else:
            print("Running again...
"
            print("I predict...

... that you'll be back!")

    print("\nI predict...

... that you'll be back!")

if __name__ == '__main__':
    main()
Functions you'll write

All use loops...

Menu

(0) Input a new list
(1) Print the current list
(2) Find the average price
(3) Find the standard deviation
(4) Find the min and its day
(5) Find the max and its day
(6) Your TTS investment plan
(9) Quit

Enter your choice:

```
def average(L):

    sum = 0
    for i in range(len(L)):
        sum += L[i]
    avg = sum / len(L)
    return avg

def stdev(L):
    avg = average(L)
    sum = 0
    for i in range(len(L)):
        sum += (L[i] - avg)**2
    stdev = (sum / len(L))**0.5
    return stdev

def minday(L):
    min = L[0]
    min_day = 0
    for i in range(len(L)):
        if L[i] < min:
            min = L[i]
            min_day = i
    return (min, min_day)

def maxday(L):
    max = L[0]
    max_day = 0
    for i in range(len(L)):
        if L[i] > max:
            max = L[i]
            max_day = i
    return (max, max_day)
```

webbrowser.open_new_tab(url)
Min price

L = [ 40, 80, 10, 30, 27, 52, 5, 15 ]

m =

m is the "min so far"

What's the idea for finding the smallest (minimum) price?

track the value of the minimum so far as you loop over L
Min price vs. min \textit{day}

\begin{align*}
L &= [40, 80, 10, 30, 27, 52, 5, 15] \\
\text{def } \text{minprice}( L ): \\
&\quad m = L[0] \\
&\quad \text{for } x \text{ in } L: \\
&\quad \quad \text{if } x < m: \\
&\quad \quad \quad m = x \\
&\quad \text{return } m
\end{align*}

What about the \textit{day} of the minimum price?
Mid-term feedback ...

I would love to know any thoughts you have about CS5 thus far in the term. In particular, how you feel about the time and effort CS5 requires...

On average, how much time per week do you spend on CS5 outside class + lab?

How does CS5’s workload compare to other classes you’re taking this term?

How would you judge the pace of CS5?

Circle your year:  First-year  Sophomore  Junior  Senior  Other

Something you’d keep about CS5 ...?

Something you’d change about / get rid of / add to CS5 ...?

Other thoughts optional, but 142% welcome:
Finish this code to return the index (location) of L's min.

```python
>>> i_min( [9, 8, 5, 7, 42] )
2

def i_min( L ):
    minval = L[0]
    minloc = 0
    for i in range(len(L)):
        if __________:
            minval = ___
            minloc = ___
    return minloc
```

Hints:
- track of the minimum value in minval
- track the location of the min inside minloc

What does this print?

```python
for i in range(4):
    for j in range(4):
        print(abs(i-j),end=')
    print()
```

Write mindiff to return the smallest absolute difference between any two elements from L. Only consider abs differences.
L will be a list of numbers.
Hint: Use a nested loop!

```python
>>> mindiff( [42,3,47,100,-9] )
5
```

Quiz, p.2
Brots...!
def i_min(L):
    minval = L[0]
    minloc = 0
    for i in range(len(L)):
        if minval > L[i]:
            minval = L[i]
            minloc = i
    return minloc

L = [40, 80, 10, 30, 27, 52, 5, 15]

minloc = 0
minval = 40

day 0

minloc = 2
minval = 10

day 1

minloc = 6
minval = 5

day 2

6 is returned
```python
def i_min(L):
    minval = L[0]
    minloc = 0
    for i in range(len(L)):
        if L[i] < minval:
            minval = L[i]
            minloc = i
    return minloc
```

---

L = [40, 80, 10, 30, 27, 52, 5, 15]

<table>
<thead>
<tr>
<th>day</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>1</td>
<td>80</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>27</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
</tr>
</tbody>
</table>

- minloc = 0
- minval = 40
- day 1: minval = 10
- day 2: minval = 10, minloc = 2
- day 3: minval = 5
- day 5: 6 is returned

**Implementation Notes:**

- Track both day and price
- Loop!
- Update when needed
Nested loops...

```python
[0,1,2,3]
for i in list(range(4)):
    for j in list(range(4)):
        print(abs(i-j), end='')
    print()
```
Write `mindiff` to return the **smallest** abs. diff. between any two elements from `L`.

```python
def mindiff( L ):
    m = abs(L[1]-L[0])
    for i in range(len(L)):
        for j in range(len(L)):
            if
    return m
```

**Hint:** Use nested loops:
```python
for i in range(4):
    for j in range(4):
```

Track the value of the **minimum so far** as you loop over `L` **twice**...
Write `mindiff` to return the smallest abs. diff. between any two elements from `L`.

```python
def mindiff(L):
    m = abs(L[1]-L[0])
    for i in range(len(L)):
        for j in range(i+1, len(L)):
            if abs(L[j]-L[i]) < m:
                m = abs(L[j]-L[i])
    return m
```

`mindiff([42, 3, 7, 100, -9])`

4

L

**Hint:** Use nested loops:
```python
for i in range(4):
    for j in range(4):
        # Track the value of the minimum so far as you loop over `L` twice...
```
T. T. Securities

"Taking the broke out of brokerage."

(0) Input a new list
(1) Print the current list
(2) Find the average price
(3) Find the standard deviation
(4) Find the min and its day
(5) Find the max and its day
(6) Your TTS investment plan
(9) Quit
Enter your choice:

Software side ...

Hardware side...

Investment analysis for the 21st century ... and beyond
The TTS advantage!

Your stock's prices: \[ L = [40, 80, 10, 30, 27, 52, 5, 15] \]

<table>
<thead>
<tr>
<th>Day</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>40.0</td>
</tr>
<tr>
<td>1</td>
<td>80.0</td>
</tr>
<tr>
<td>2</td>
<td>10.0</td>
</tr>
<tr>
<td>3</td>
<td>30.0</td>
</tr>
<tr>
<td>4</td>
<td>27.0</td>
</tr>
<tr>
<td>5</td>
<td>52.0</td>
</tr>
<tr>
<td>6</td>
<td>5.0</td>
</tr>
<tr>
<td>7</td>
<td>15.0</td>
</tr>
</tbody>
</table>

*Important fine print:*

To make our business plan **realistic**, however, we only allow selling **after** buying.
The TTS advantage!

Important fine print:

For each buy-day, \( b \):

For each sell-day, \( s \):

Compute the profit

If it's the max-so-far:

Remember it in a variable!

Your stock's prices:

\[ L = [40, 80, 10, 30, 27, 52, 5, 15] \]

To make our business plan **realistic**, however, we only allow selling **after** buying.