Presentation 24

Coroutines
Announcements for This Lecture

Assignment 7

- We heard the complaints
  - Task 1 too hard for C-level
  - Task 2 just right for B/B+

- Relaxing **no code rule**
  - Can show for **Task 1 only**
  - Will guide you through it

- Due **December 21**
  - No extensions or lates
  - We have to grade it!

Lesson Videos

- **ALL** all are now posted
  - **Lesson 29** for today
  - **Lesson 30** is the last

- Will **not** cover Lesson 30
  - Not relevant for A7
  - Really if going to 2110
  - Last class = Office Hours

- Special lecture **December 10**
  - The “advising” lecture
More Announcements

Labs

- Lab today is **optional**
  - Not required at all
  - But can do for **extra credit**
  - Need 19/23 labs finished
- **No more official labs**
  - Office hours for A7 help
  - Easier than waiting in queue
  - Follows new code rules

Surveys

- Will have a **survey for A7**
- Will also have an **exit survey**
  - Follow up to Survey 0
  - Should I keep the **videos**?
  - Would you rather had a **final**?
- Also the **course evaluation**
  - Comes from engineering
  - Completely **anonymous**
  - But we know if you submit
From Last Time: Chaining Generators

```python
def sumfold(input):
    
    """
    Generates the sums of the numbers seen so far in input
    
    Example: sumfold([1,2,3]) generates the numbers 1, 3, and 6
    
    Parameter input: The input data to sum
    Precondition: input is a iterable of numbers (int or float)
    """
    pass
```
def sumfold(input):
    
    Generates the sums of the numbers seen so far in input

Example: sumfold([1,2,3]) generates the numbers 1, 3, and 6

Parameter input: The input data to sum
Precondition: input is an iterable of numbers (int or float)

pass
def filterdiv(input, n):
    """Generates all elements of input evenly divisible by n

    The elements are generated in the order they appear in input.

    Example: filterdiv([1, 2, 3, 4], 2) generates the numbers 2 and 4

    Parameter input: The input data to filter
    Precondition: input is a iterable of int

    Parameter n: The number to divide by
    Precondition: n is an int"

pass
def sumfold(lst):
    """Returns list of sums""
    sum = []
    g = pushsum(len(lst))
    next(g)
    for x in lst:
        a = g.send(x)
        sum.append(a)
    return sum

def pushsum(n):
    """Yields sum of all sent""
    sum = 0
    for x in range(n):
        val = (yield sum)
        sum = sum + val
    yield sum
## Activity: Call Frame Time

### Parent Function

```python
def sumfold(lst):
    
    """Returns list of sums""
    sum = []
    g = pushsum(len(lst))
    next(g)
    for x in lst:
        a = g.send(x)
        sum.append(a)
    return sum
```

### Coroutine

```python
def pushsum(n):
    
    """Yields sum of all sent""
    sum = 0
    for x in range(n):
        val = (yield sum)
        sum = sum + val
    yield sum
```

Sometimes called the **driver**

<table>
<thead>
<tr>
<th>Parent Function</th>
<th>Coroutine</th>
</tr>
</thead>
<tbody>
<tr>
<td>def sumfold(lst):</td>
<td>def pushsum(n):</td>
</tr>
<tr>
<td>&quot;&quot;&quot;Returns list of sums&quot;&quot;&quot;&quot;</td>
<td>&quot;&quot;&quot;&quot;Yields sum of all sent&quot;&quot;&quot;&quot;</td>
</tr>
<tr>
<td>sum = []</td>
<td>sum = 0</td>
</tr>
<tr>
<td>g = pushsum(len(lst))</td>
<td>for x in range(n):</td>
</tr>
<tr>
<td>next(g)</td>
<td>val = (yield sum)</td>
</tr>
<tr>
<td>for x in lst:</td>
<td>sum = sum + val</td>
</tr>
<tr>
<td>a = g.send(x)</td>
<td>yield sum</td>
</tr>
<tr>
<td>sum.append(a)</td>
<td></td>
</tr>
<tr>
<td>return sum</td>
<td></td>
</tr>
</tbody>
</table>

12/8/20

Coroutines
**Activity: Call Frame Time**

---

**Parent Function**

```python
def sumfold(lst):
    
    """Returns list of sums"""
    sum = []
    g = pushsum(len(lst))
    next(g)
    for x in lst:
        a = g.send(x)
        sum.append(a)
    return sum
```

**Function Call**

```python
>>> x = sumfold([1,2])
```

Assume we are here:

```
sumfold  lst  id3  g  id5  34
sum  id4
id3  list
  0
  1
  2
id4  list
```

What is the next step?
Which One is Closest to Your Answer?

A:  
\[
\begin{array}{c}
\text{sumfold} & \text{lst} & \text{id3} & g & \text{id5} & 35 \\
\text{sum} & \text{id4} \\
\end{array}
\]

B:  
\[
\begin{array}{c}
\text{sumfold} & \text{lst} & \text{id3} & g & \text{id5} & 34 \\
\text{sum} & \text{id4} \\
\text{pushsum} & n & 2 & 16 \\
\text{sum} & 0 \\
\end{array}
\]

C:  
\[
\begin{array}{c}
\text{sumfold} & \text{lst} & \text{id3} & g & \text{id5} & 34 \\
\text{sum} & \text{id4} \\
\text{pushsum} & n & 2 & 16 \\
\end{array}
\]

D:  
\[
\begin{array}{c}
\text{sumfold} & \text{lst} & \text{id3} & g & \text{id5} & 34 \\
\text{sum} & \text{id4} \\
\text{pushsum} & 16 \\
\end{array}
\]

12/8/20 Coroutines
Which One is Closest to Your Answer?

A:
```
sumfold  lst  id3  g  id5  35
sum     id4
```

B:
```
sumfold  lst  id3  g  id5  34
sum     id4
```

C:
```
sumfold  lst  id3  g  id5  34
sum     id4
```

D:
```
pushsum  n  2  16
sum     id4
```

In all cases, the heap is unchanged
def pushsum(n):
    '''Yields sum of all sent'''
    sum = 0
    for x in range(n):
        val = (yield sum)
        sum = sum + val
    yield sum

>>> x = sumfold([1, 2])

What is the next step?
Which One is Closest to Your Answer?

A:

\[
\begin{array}{cccc}
\text{sumfold} & \text{lst} & \text{id3} & \text{g} \\
\text{sum} & \text{id4} & & \\
\text{pushsum} & \text{n} & 2 & 17 \\
\text{sum} & 0 & & \\
\end{array}
\]

B:

\[
\begin{array}{cccc}
\text{sumfold} & \text{lst} & \text{id3} & \text{g} \\
\text{sum} & \text{id4} & & \\
\text{pushsum} & \text{n} & 2 & x \\
\text{sum} & 0 & & \\
\end{array}
\]

C:

\[
\begin{array}{cccc}
\text{sumfold} & \text{lst} & \text{id3} & \text{g} \\
\text{sum} & \text{id4} & & \\
\text{pushsum} & \text{n} & 2 & x \\
\text{sum} & 0 & & \\
\end{array}
\]

D:

\[
\begin{array}{cccc}
\text{sumfold} & \text{lst} & \text{id3} & \text{g} \\
\text{sum} & \text{id4} & & \\
\text{pushsum} & \text{n} & 2 & x \\
\text{sum} & 0 & & \\
\end{array}
\]

12/8/20
Which One is Closest to Your Answer?

In all cases, the heap is unchanged
**Activity: Call Frame Time**

### Coroutine

```python
def pushsum(n):
    
    #"Yields sum of all sent"
    sum = 0
    for x in range(n):
        val = (yield sum)
        sum = sum + val
    yield sum
```

### Function Call

```python
>>> x = sumfold([1,2])
```

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td><code>sum = 0</code></td>
</tr>
<tr>
<td>17</td>
<td><code>for x in range(n):</code></td>
</tr>
<tr>
<td>18</td>
<td><code>val = (yield sum)</code></td>
</tr>
<tr>
<td>19</td>
<td><code>sum = sum + val</code></td>
</tr>
<tr>
<td>20</td>
<td><code>yield sum</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td><code>sumfold lst id3 g id5 35</code></td>
</tr>
<tr>
<td></td>
<td><code>sum id4</code></td>
</tr>
<tr>
<td></td>
<td><code>pushsum n 2 17</code></td>
</tr>
<tr>
<td></td>
<td><code>sum 0</code></td>
</tr>
</tbody>
</table>

What is the **next step**?
Which One is Closest to Your Answer?

A:

<table>
<thead>
<tr>
<th>Pushsum</th>
<th>lst</th>
<th>Id3</th>
<th>g</th>
<th>Id5</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum</td>
<td>Id4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

B:

<table>
<thead>
<tr>
<th>Pushsum</th>
<th>lst</th>
<th>Id3</th>
<th>g</th>
<th>Id5</th>
<th>34</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum</td>
<td>Id4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C:

<table>
<thead>
<tr>
<th>Pushsum</th>
<th>lst</th>
<th>Id3</th>
<th>g</th>
<th>Id5</th>
<th>34</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum</td>
<td>Id4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D:

<table>
<thead>
<tr>
<th>Pushsum</th>
<th>lst</th>
<th>Id3</th>
<th>g</th>
<th>Id5</th>
<th>34</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum</td>
<td>Id4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

12/8/20
def pushsum(n):
    """Yields sum of all sent"""
    sum = 0
    for x in range(n):
        val = (yield sum)
        sum = sum + val
    yield sum

>>> x = sumfold([1,2])

What is the next step?
### Which One is Closest to Your Answer?

<table>
<thead>
<tr>
<th></th>
<th>sumfold</th>
<th>lst</th>
<th>id3</th>
<th>g</th>
<th>id5</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>sum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>pushsum</td>
<td>n</td>
<td>2</td>
<td>x</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>sum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>pushsum</td>
<td>n</td>
<td>2</td>
<td>x</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>sum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>sum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>pushsum</td>
<td>n</td>
<td>2</td>
<td>x</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>sum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>pushsum</td>
<td>n</td>
<td>2</td>
<td>x</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>sum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

12/8/20

Coroutines
def sumfold(lst):
    
    """Returns list of sums"""

    sum = []
    g = pushsum(len(lst))
    next(g)
    for x in lst:
        a = g.send(x)
        sum.append(a)
    return sum

>>> x = sumfold([1,2])

What is the next step?
Which One is Closest to Your Answer?

A: 
```
<table>
<thead>
<tr>
<th>sumfold</th>
<th>lst</th>
<th>id3</th>
<th>g</th>
<th>id5</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>sum</td>
<td>id4</td>
<td>x</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

B: 
```
<table>
<thead>
<tr>
<th>sumfold</th>
<th>lst</th>
<th>id3</th>
<th>g</th>
<th>id5</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>sum</td>
<td>id4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pushsum</td>
<td>n</td>
<td>2</td>
<td>x</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>sum</td>
<td>0</td>
<td>RETURN</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

C: 
```
<table>
<thead>
<tr>
<th>sumfold</th>
<th>lst</th>
<th>id3</th>
<th>g</th>
<th>id5</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>sum</td>
<td>id4</td>
<td>x</td>
<td>1</td>
<td>a</td>
<td>0</td>
</tr>
<tr>
<td>pushsum</td>
<td>n</td>
<td>2</td>
<td>x</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>sum</td>
<td>0</td>
<td>RETURN</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

D: 
```
<table>
<thead>
<tr>
<th>sumfold</th>
<th>lst</th>
<th>id3</th>
<th>g</th>
<th>id5</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>sum</td>
<td>id4</td>
<td>x</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pushsum</td>
<td>n</td>
<td>2</td>
<td>x</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>sum</td>
<td>0</td>
<td>RETURN</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

12/8/20
def sumfold(lst):
    """Returns list of sums""
    sum = []
    g = pushsum(len(lst))
    next(g)
    for x in lst:
        a = g.send(x)
        sum.append(a)
    return sum

>>> x = sumfold([1,2])

What is the next step?
Which One is Closest to Your Answer?

A: \[ \text{sumfold} \quad \text{lst} \quad \text{id3} \quad \text{g} \quad \text{id5} \quad 36 \]
\[ \text{sum} \quad \text{id4} \quad x \quad 0 \quad a \quad 0 \]

B: \[ \text{sumfold} \quad \text{lst} \quad \text{id3} \quad \text{g} \quad \text{id5} \quad 36 \]
\[ \text{sum} \quad \text{id4} \quad x \quad 0 \]

C: \[ \text{sumfold} \quad \text{lst} \quad \text{id3} \quad \text{g} \quad \text{id5} \quad 36 \]
\[ \text{sum} \quad \text{id4} \quad x \quad 1 \quad a \quad 1 \]

D: \[ \text{sumfold} \quad \text{lst} \quad \text{id3} \quad \text{g} \quad \text{id5} \quad 36 \]
\[ \text{sum} \quad \text{id4} \quad x \quad 1 \]
def sumfold(lst):
    """Returns list of sums"""
    sum = []
    g = pushsum(len(lst))
    next(g)
    for x in lst:
        a = g.send(x)
        sum.append(a)
    return sum

>>> x = sumfold([1, 2])

What is the next step?
Which One is Closest to Your Answer?

A: 
```
sumfold lst id3 g id5 37
sum id4 x 1 a 1
```

B: 
```
sumfold lst id3 g id5 36
sum id4 x 1
pushsum n 2 16
```

C: 
```
sumfold lst id3 g id5 36
sum id4 x 1
pushsum n 2 x 0 18
sum 0 RETURN 0
```

D: 
```
sumfold lst id3 g id5 36
sum id4 x 1
pushsum n 2 x 0 19
sum 0 val 1
```
Activity: Call Frame Time

Coroutine

def pushsum(n):
    """Yields sum of all sent"""
    sum = 0
    for x in range(n):
        val = (yield sum)
        sum = sum + val
    yield sum

Function Call

>>> x = sumfold([1,2])

Try the rest on your own
def chunkify(input):
    """Coroutine to break a list into chunks.
    Each call to send is the number of elements to chunk. At each call, it yields a new list of the size of the number of elements in send.
    If the size sent is not an int or is <= 0, it yields the empty list.
    Parameter input: The data to process
    Precondition: input is an iterable"
    pass
def chunkify(input):
    """Coroutine to break a list into chunks.

    Each call to send is the number of elements to chunk. At each call, it yields a new list of the size of the number of elements in send.

    If the size sent is not an int or is <= 0, it yields the empty list

    Parameter input: The data to process
    Precondition: input is an iterable"
    pass

Kind of like a precondition
Demo: Writing the Parent/Driver

```python
def printit(lst, step):
    """Prints the contents of lst in groups of size step

    Each group is printed on a line by itself.

    Parameter lst: The list to process
    Precondition: lst is a list

    Parameter step: The number of elements to print at a time
    Precondition: step is an int > 0"
    pass
```

12/8/20  Coroutines  28
Questions?