Compsci 101
Introduction
Part 1 of 4

Susan Rodger
Nicki Washington
February 11, 2021

G is for …

- Google
  - How to find the answer to everything
- Global Variable
  - Accessible everywhere, don’t do this at home?
- GIGO
  - Garbage In, Garbage Out
- Git
  - Working Together or Solo

PFTD

- Lists continued
- String methods and more

List Cloning (or copying)

```python
lst1 = ['a', 'b', 1, 2]
lst2 = lst1
lst3 = lst1[:]
```
List Cloning (or copying)

```python
lst1 = ['a', 'b', 1, 2]
lst2 = lst1
lst3 = lst1[:]
```

Frames

Objects

```
['a' 'b' 1 2]
```

List Cloning (or copying)

```python
lst1 = ['a', 'b', 1, 2]
lst2 = lst1
lst3 = lst1[:]
```

Global frame

```
list
```

```
0  "a"  "b"  1  2
```

List Cloning (or copying)

```python
lst1[0] = "SUN"
```

Global frame

```
list
```

```
0  "a"  "b"  1  2
```

```
0  "SUN"  1  2
```
List Concatenation Steps

1. Calculate the **length** of the new list
2. **Create** list of that length
3. **Copy** values from first list
4. **Copy** values from second list
5. **Assign the variable to the new list**

\[
\begin{align*}
1 & \text{ lst0 } = [1,2] \\
2 & \text{ lst1 } = [3,4,5] \\
3 & \text{ lst2 } = \text{ lst0 + lst1}
\end{align*}
\]
Concatenation: Makes new List

1. Calculate length
2. Create new list
3. Copy left list
4. Copy right list
5. Assign lst2

What will Python Tutor Display? How many lists will there be?

Concatenation: Makes new List

1. lst0 = [1, 2]
2. tmp = lst0
3. lst0 = lst0 + [4]
Concatenation: Makes new List

1. `lst0 = [1, 2]`
2. `tmp = lst0`
3. `lst0 = lst0 + [4]`

```
<table>
<thead>
<tr>
<th>Frames</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global frame</td>
<td>list</td>
</tr>
<tr>
<td>lst0</td>
<td>1</td>
</tr>
<tr>
<td>tmp</td>
<td>1 2</td>
</tr>
<tr>
<td>lst0</td>
<td>1 2 4</td>
</tr>
</tbody>
</table>
```

5. Assign `lst0`

---

Concatenation: length, create, copy, copy, assign

- How is the inner list copied?

```
1. `lst0 = [1, ['b', 3.0]]`
2. `lst1 = [4]`
3. `lst2 = lst0 + lst1`
```

What will Python Tutor Display? How many copies of ['b', 3.0] will be present?

---

Concatenation: length, create, copy, copy, assign

- How is the inner list copied?

```
1. `lst0 = [1, ['b', 3.0]]`
2. `lst1 = [4]`
3. `lst2 = lst0 + lst1`
```

```
<table>
<thead>
<tr>
<th>Frames</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>frame</td>
<td>list</td>
</tr>
<tr>
<td>lst0</td>
<td>1</td>
</tr>
<tr>
<td>lst1</td>
<td>1</td>
</tr>
<tr>
<td>lst2</td>
<td>1 2 3 4</td>
</tr>
</tbody>
</table>
```

3. Copy `lst0`

---

Concatenation: length, create, copy, copy

- How is the inner list copied?

```
1. `lst0 = [1, ['b', 3.0]]`
2. `lst1 = [4]`
3. `lst2 = lst0 + lst1`
```

```
<table>
<thead>
<tr>
<th>Frames</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>frame</td>
<td>list</td>
</tr>
<tr>
<td>lst0</td>
<td>1</td>
</tr>
<tr>
<td>lst1</td>
<td>1</td>
</tr>
<tr>
<td>lst2</td>
<td>1 2 3</td>
</tr>
</tbody>
</table>
```

4. Copy `lst0`

This is a shallow copy!
Don't copy inner lists
List Mutation: `.append(…)`

- `.append()` – list function that adds element to end of list
  - Mutates list to left of “.”
  - “.” – call function to the right of the dot on the thing to the left of the dot (LEFT.RIGHT)

```python
1 lst0 = [1, 2, 3]
2 tmp = lst0
3 lst0.append(4)
```

What will Python Tutor Display? One or two lists?
List Mutation: `.append(…)`

```python
1. lst0 = [1, 2, 3]
2. tmp = lst0
3. lst0.append(4)
```

List Mutation: `.append(…)`

```python
1. lst0 = [1, 2, 3]
2. tmp = lst0
3. lst0.append(4)
```

List Mutation: `.append(…)`

```python
1. lst0 = [1, 2, 3]
2. tmp = lst0
3. lst0.append(4)
```

List Mutation: `.append(…)`

```python
lst0 = [1, 2, 3]
tmp = lst0
lst0.append(4)
lst0.append([5, 6])
```

Same list! No new list

```
.frame
lst0
.tmp
```

```
.frame
lst0
.tmp
```

```
.frame
lst0
.tmp
```

```
.frame
lst0
.tmp
```
List Mutation: `.append(...)`

```python
lst0 = [1, 2, 3]
tmp = lst0
lst0.append(4)
lst0.append([5, 6])
```

```
<table>
<thead>
<tr>
<th>Frames</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global frame</td>
<td>lst</td>
</tr>
<tr>
<td></td>
<td>0 1 2 3 4 5 6</td>
</tr>
<tr>
<td>lst0</td>
<td>tmp</td>
</tr>
</tbody>
</table>
```

Same list! No new list.

String’s `split(...)`

- Strings have functions too!
- `TYPE_STRING.FUNCTION(PARAMETERS)`
  - "." means apply function to what is on the left
  - `'one fish two fish'.split()` returns a list
    - `['one', 'fish', 'two', 'fish']`
  - What did it divide the string by?
    - When no parameter, default whitespace
  - `'one fish, two fish'.split(',')` returns a list
    - `['one fish', 'two fish']`

String’s `join(...)`

- `TYPE_STRING.join(SEQ_OF_STRINGS)`
  - Opposite of `split()`
  - Creates string from sequence's items separated by the string to the left of `join`
  - `' .join(['one','fish','two','fish'])` returns a string
    - `'one fish two fish'`
  - `'+'.join(['one','fish','two','fish'])` returns a string
    - `'one+fish+two+fish'`
• Professor MIT
• MacArthur Genious Grant Awardee
• ACM Fellow, IEEE Fellow
• Robotics, mobile computing

She says about her robotics lab: “Everyone thinks about how to make the future better, what kinds of things we need in the future. People have wild and crazy ideas and people are fun. We are excited, we are full of life and we love what we do, most importantly.”