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
---|---
```
Global frame
lst1
```
```
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
```
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

- Global frame
  - `lst1`
  - `lst2`
  - `lst3`

Objects

```
list
0  "a"  1  "b"  2  3  2
```

```
list
0  "a"  1  "b"  2  3  2
```
List Cloning (or copying)

```python
lst1 = ['a', 'b', 1, 2]
lst2 = lst1
lst3 = lst1[:]
lst1[-1] = "SUN"
```
Compsci 101
Introduction
Part 2 of 4

Susan Rodger
Nicki Washington
February 11, 2021
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

\[
\begin{align*}
1 & \quad \text{lst0} = [1, 2] \\
2 & \quad \text{lst1} = [3, 4, 5] \\
3 & \quad \text{lst2} = \text{lst0} + \text{lst1}
\end{align*}
\]
Concatenation:
length, create, copy, copy, assign

1 \( \text{lst0} = [1,2] \)
2 \( \text{lst1} = [3, 4, 5] \)
3 \( \text{lst2} = \text{lst0} + \text{lst1} \)
Concatenation:

length, create, copy, copy, assign

1. Calculate length
2. Create new list
3. Copy left list
Concatenation:
length, create, copy, copy, assign

1. \( \text{lst0} = [1, 2] \)
2. \( \text{lst1} = [3, 4, 5] \)
3. \( \text{lst2} = \text{lst0} + \text{lst1} \)

4. Copy right list

5. Assign lst2

```python
lst0 = [1, 2]
lst1 = [3, 4, 5]
lst2 = lst0 + lst1
```
Concatenation: Makes new List

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

What will Python Tutor Display? How many lists will there be?
Concatenation: Makes new List

1 \texttt{lst0 = [1,2]}
2 \texttt{tmp = lst0}
3 \texttt{lst0 = lst0 + [4]}
Concatenation: Makes new List

1. Calculate length
2. Create new list
3. Copy left list

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

1. \( \text{lst0} = [1,2] \)
2. \( \text{tmp} = \text{lst0} \)
3. \( \text{lst0} = \text{lst0} + [4] \)

Frames

- Global frame
  - lst0
  - tmp

Objects

- list
  - 0 1 2
  - 1 2
  - 1 2 4

4. Copy right list
5. Assign lst0
Concatenation:
length, create, copy, copy, assign

• How is the inner list copied?

```python
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. Calculate length

2. Create new list

3. Copy left list
Concatenation: length, create, copy, copy

• How is the inner list copied?

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

This is a shallow copy!
Don’t copy inner lists
Compsci 101
Introduction
Part 3 of 4

Susan Rodger
Nicki Washington
February 11, 2021
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`)
List Mutation: .append(…)

```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(…)`

```
1  lst0 = [1, 2, 3]
2  tmp = lst0
3  lst0.append(4)
```
List Mutation: `.append(...)`

1. `lst0 = [1, 2, 3]`
2. `tmp = lst0`
3. `lst0.append(4)`
List Mutation: `.append(…)`

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

Same list! No new list
List Mutation: `.append(…)`

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

![Diagram showing list mutation with code examples]
List Mutation: `.append(…)`

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

Same list! No new list.
Compsci 101
Introduction
Part 4 of 4

Susan Rodger
Nicki Washington
February 11, 2021
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(',',')'
  - ['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"])`
  • 'one fish two fish'

• `'+'.join(["one","fish","two","fish"])`
  • ‘one+fish+two+fish’
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.”