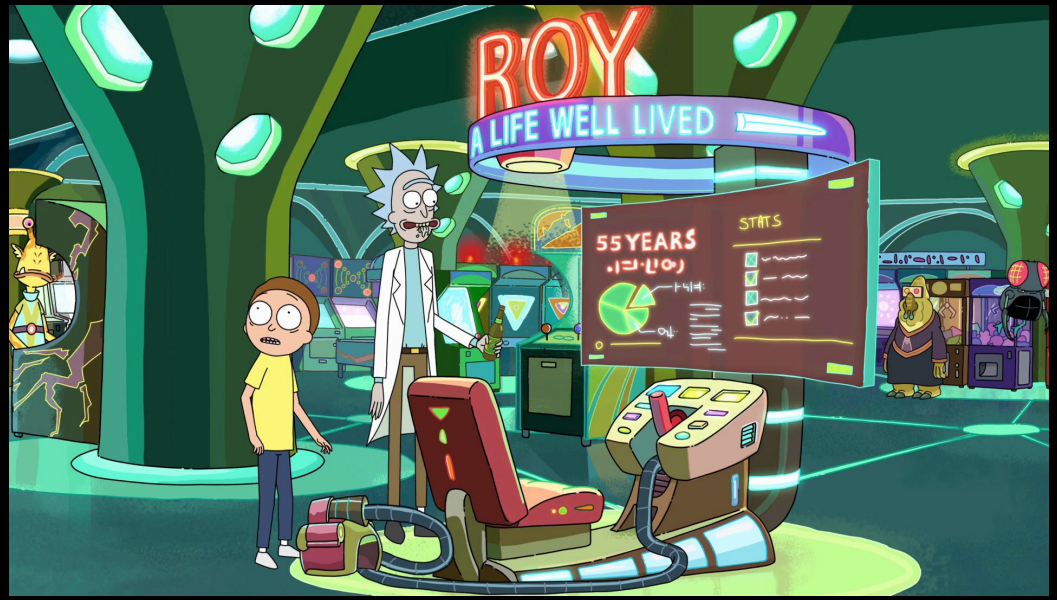
CS61A Discussion #04

TA: Hyun Jae Moon (hyunjaemoon@berkeley.edu)

Nonlocals & Mutation



General Comments from the Survey

- More Practice!
  - Midterm Problems?!?!
- Keep up with the Live Coding!
- Try to finish Discussion worksheet?
  - This one is kinda hard :(I'm sorry, but I'll try to pick the important questions to go over.
- Slides?
  - It's on the website!
- Mixed reax from Explaining Practice Problems
  - Some want more / Some want less. I'll try to be more efficient :)
  - I'll also try to be more clear in my thought process

Agenda

- Nonlocals, Dictionaries,List Mutation
  - 1.2 (or 1.3), 2.1, 2.2, 3.1, 3.2

Attendance

Link: tinyurl.com/moondisc04

Secret Word: donewithmaps!

Nonlocals

- So far we were only allowed to modify variables in the current frame.
- Now you can modify variables in the parent frame!!
    - Except for Global Frame :(

Nonlocals

```
>>> def f(x):
...     def g():
...         x += 1
...         return x
...     return g
...
>>> f(1)()
UnboundLocalError
```

```
>>> def f(x):
...     def g():
...         nonlocal x
...         x += 1
...         return x
...     return g
...
>>> f(1)()
2
```

Nonlocals - Confusing

```
>>> x = 1
>>> def f():
...     nonlocal x
...     return x + 1
...
SyntaxError
```

```
>>> def f():
...     x = 1
...     def g():
...         x = 2
...         def h():
...             nonlocal x
...             return x
...         return h
...     return g
...
>>> f()()()
2
```

Nonlocals - More Confusing

```
>>> def f():
...     x = 1
...     def g(x):
...         nonlocal x
...         return x
...     return g
...
SyntaxError
```

LIVE CODING!!!

Solve 1.2 OR 1.3?

List Mutations

```
>>> a = [1, 2, 3]
>>> a.append(4)
>>> a
[1, 2, 3, 4]

>>> a.insert(2, 1)
>>> a
[1, 2, 1, 3, 4]
```

```
>>> a.remove(1)
>>> a
[2, 1, 3, 4]

>>> b = a.pop()
>>> a
[2, 1, 3]
>>> b
4
```

More List Mutations

```
>>> a = [1, 2, 3]
>>> b = a + [4, 5]
>>> a
[1, 2, 3]
>>> b
[1, 2, 3, 4, 5]
```

```
>>> a = [1, 2, 3]
>>> c = a
>>> c.append(4)
>>> c
[1, 2, 3, 4]
>>> a
[1, 2, 3, 4]
```

‘is’ vs ‘==’

```
>>> a = [1]
>>> b = [1]
>>> a is b
False
>>> a == b
True
```

```
>>> a = [1]
>>> b = a
>>> a is b
True
>>> a == b
True
```

```
>>> a = [1]
>>> b = a[:]
>>> a is b
False
>>> a == b
True
```

Is there a case where ‘`a == b`’ is False, but ‘`a is b`’ is True? Discuss!

Solve 2.1

Coding for List Mutations

```
>>> def cycle_push(lst):
...     x = lst.pop()
...     lst.insert(0, x)
...
>>> a = [1, 2, 3, 4]
>>> cycle_push(a)
>>> a
[4, 1, 2, 3]
```

Solve 2.2

Dictionaries

```
>>> d = {}

>>> d[1] = 2

>>> d['a'] = abs

>>> d

{1: 2, 'a': <built-in function abs>}
```

Dictionaries with for loop

```
>>> d = {'a': 1, 'b': 2, 'c': 3}
>>> for key in d:
...     print(key, d[key])
...
a 1
b 2
c 3
```

Solve 3.1, 3.2