Iterators

Announcements

List Practice

Spring 2023 Midterm 2 Question 1

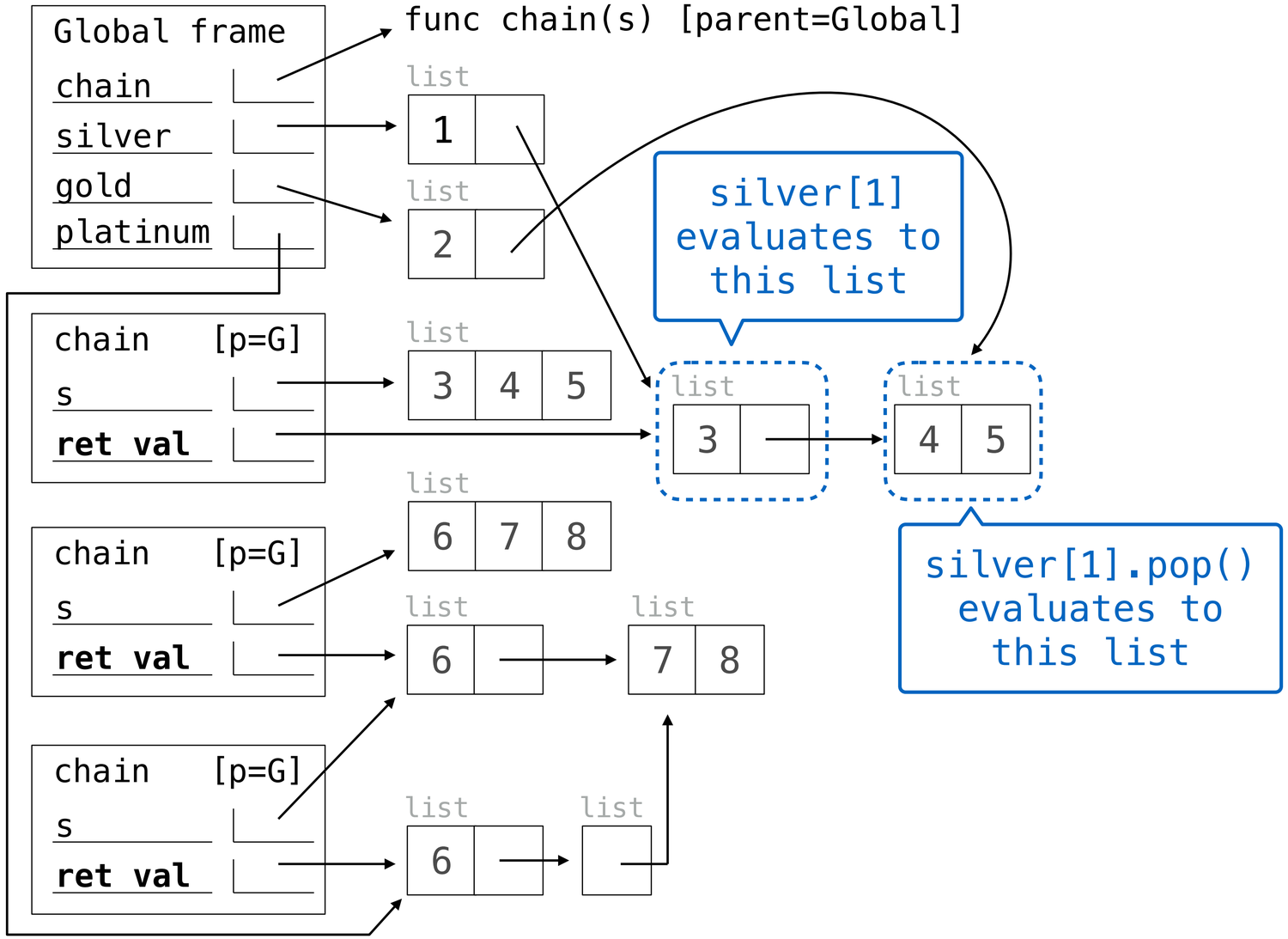
```
def chain(s):
    return [s[0], s[1:]]
silver = [2, chain([3, 4, 5])]
gold = [silver[0], silver[1].pop()]
silver[0] = 1
platinum = chain(chain([6, 7, 8]))
```

Reminder: s.pop() removes and returns the last item in list s.

```
>>> silver
[1, [3]]

>>> gold
[2, [4, 5]]

>>> platinum
[6, [[7, 8]]]
```



Tuples

Tuples vs. Lists

```
t = ( 1, 2, 3 )
```

- Immutable
- Smaller
- Faster
- Can be used as a key in a dictionary key

```
l = [ 1, 2, 3 ]
```

- Mutable (can reassign items)
  - More functionality
  - ex: pop()
- Bigger, slower
- Cannot be used as a dictionary key

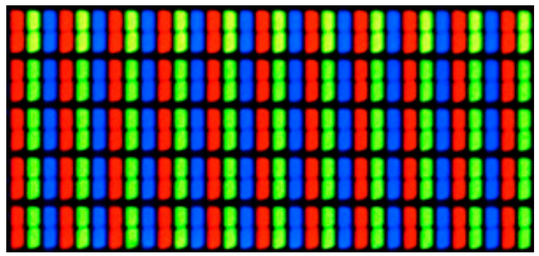
Both can be…
Accessed
Sliced (makes a new tuple/list)
Can concatenate (combine) with another tuple/list (makes a new tuple/list)

(Demo)

Sequences (lists, tuples, dictionaries, and more) in the Real World

Images are sequences of pixel data…

```
Pixel at (16, 38): (0, 128, 224)
Pixel at (17, 38): (0, 0, 0)
Pixel at (18, 38): (128, 128, 128)
```



We can store data to “look it up” later using dictionaries…

```
grades = { “george”: 82, “Belinda”: 78}

temp = { “high”: 82, “low”: 62, “humidity”: 75 }
```

Files are sequences of text…

```
>>> open(“textfile.txt”).readlines()
[‘first line in text file’,
‘second line in file’,
‘etc’]
```

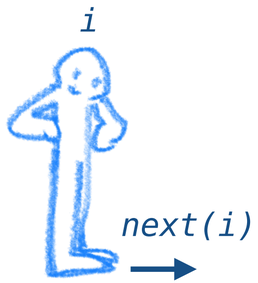
Iterators

Iterators

A container can provide an iterator that provides access to its elements in order

iter(iterable): Return an iterator over the elements of an iterable value

next(iterator): Return the next element in an iterator



s = [3, 4, 5]

```
>>> s = [3, 4, 5]
>>> i = iter(s)
>>> next(i)
3
>>> next(i)
4
>>> u = iter(s)
>>> next(u)
3
>>> next(i)
5
>>> next(u)
4
```

(Demo)

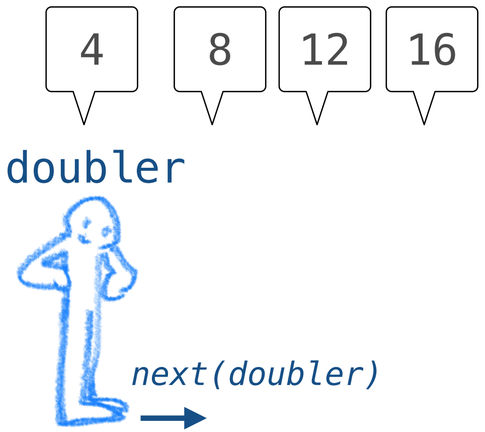
Discussion Question

What will be printed?

```
a = [1, 2, 3]
b = [a, 4]
c = iter(a)
d = c
print(next(c))
print(next(d))
print(b)
```

Map Function

(Demo)




```
l = [2, 4, 6, 8]

doubler = map(double, l)
```

```
def double(x):
    print(f"*** doubling {x} ***")
    return x*2
```

Discussion Question

all(s) iterates through s until a false value is found (or the end is reached).

What's printed when evaluating:

```
x = all(map(print, range(-3, 3)))
```

Why?

- print(-3) returns None after displaying -3
- None is a false value
- all([None, ...]) is False for any ...
- The map iterator never needs to advances beyond -3