Mutability,Iterators And Generators

Aditya Iyer

Section Outline

Total time: 50 mins

Mini lecture : 10 mins

Question 1: 5 mins

Question 3: 10 mins

Question 1 (Generator): 10 mins

Question 3 (Generator): 15mins

[Optional] exam level question

[Optional] Q&A

Mutability

Mutable: The contents of a data structure can be changed after its creation

Examples: Lists, Dictionaries

Immutable: The contents of the data structure can not be changed after its creation

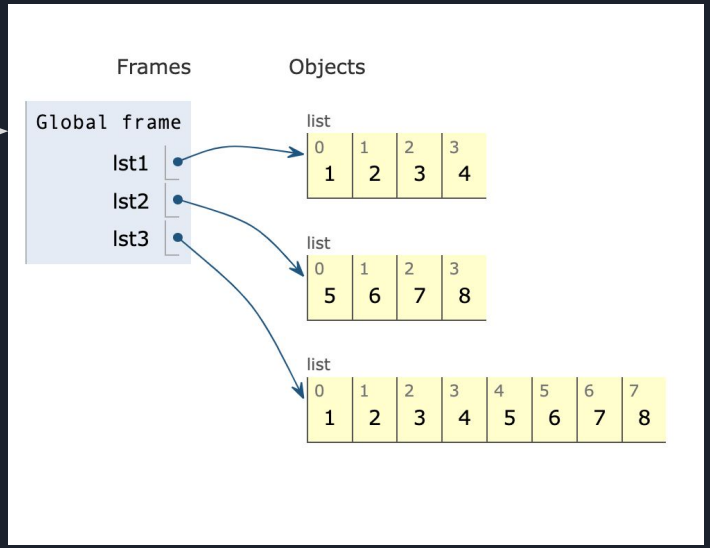
Examples: Tuples, Strings

List Mutation Functions (Credits: CSM Content Team)

- **lst.append(x)**: add x to the end of lst, only adds one element, returns None
- **lst.extend(iterable)**: add all elements of iterable to the end of lst, returns None
- **lst.insert(i, x)**: inserts x into lst at index i index, don't replace existing element
- **lst.remove(x)**: remove first appearance of x in list, error if not found
- **lst.pop(i)**: removes and <u>returns</u> element at <u>index</u> i
  - Default argument for i is the last element
- Built-in mutative methods:
  - **lst += lst1** (This is distinct from lst = lst + lst): <u>modifies</u> lst by concatenating lst1 onto the end
  - **lst[i] = x:** <u>modifies</u> lst by changing the element at index i

Non Mutative List Operations

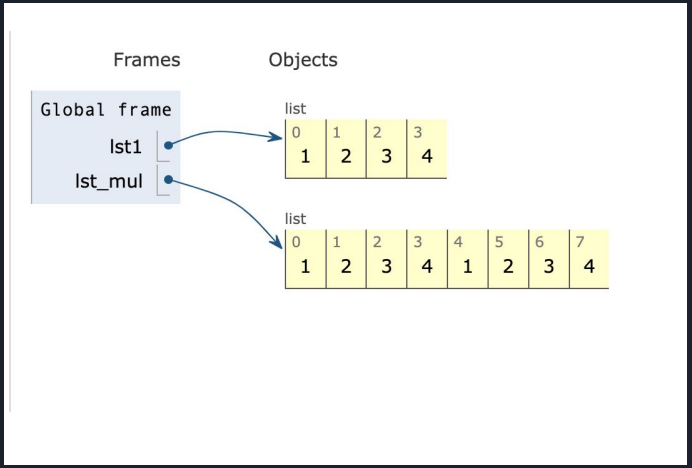
```
lst1 = [1,2,3,4]
lst2 = [5,6,7,8]

lst3 = lst1 + lst2
```



CREATES A NEW LIST

Non Mutative List Operations

```
lst_mul = 2 * lst1
```



CREATES A NEW LIST

Remember:

1. List splicing creates a new list
2. list(lst) creates a new list as long as lst is iterable

Iterators

ITERABLE: A data structure that can be iterated over. (Rough rule of thumb: if you can apply a for loop on a particular data structure it's iterable)

- Strings, lists, dictionaries, tuples are all iterables. Integers are not (can you use a for loop on an int?)

ITERATOR: An **iterator** is an object that provides sequential access to values, one by one.

ONCE YOU ACCESS A VARIABLE FROM AN ITERATOR YOU CANNOT GO BACK TO IT UNLESS YOU HAVE STORED IT SOMEWHERE ELSE

Creating Iterators

```
lst1 = ['a', 'b', 'c', 'd']
itrb = iter(lst1) # making an iterator from an iterable
```

IMPORTANT: Calling iter on an iterator returns the same iterator!!

There's a good chance you might see this on the exam

Accessing values from an Iterator

```
lst1 = ['a', 'b', 'c', 'd']
itrb = iter(lst1) # making an iterator from an iterable
```

```
>>> next(itrb)
'a'
>>> next(itrb)
'b'
>>> next(itrb)
'c'
>>> next(itrb)
'd'
>>> next(itrb)
Traceback (most recent call last):
  File <string>, line 1, in <module>
StopIteration: StopIteration
```

Generators

- Generators are a specific kind of iterator
- They only return the desired value rather than the whole sequence
- They have a yield statement instead of a return statement

Generator Example

```python
#a generator function that yields one odd value at a time
def odd():
    i = 1
    while i < 10:
        if (i%2) != 0:
            yield i
        i += 1

odd_num = odd()
```

```
>>> next(odd_num)
1
>>> next(odd_num)
3
>>> next(odd_num)
5
>>> next(odd_num)
7
>>> next(odd_num)
9
>>> next(odd_num)
Traceback (most recent call last):
  File <string>, line 1, in <module>
StopIteration: None
>>>
```

Some more useful functions

MAP: Takes in a function and a list and applies that function to every element in the list

FILTER: Takes in a function and a list and applies the function to every element in the list. Returns the values for which the function returns a truthy value.

REDUCE: Takes in a 2 argument function and a list of inputs and combines them from start to end using the function

Question 1 [Hint]

Draw a box and pointer diagram!

Question 1 (Generators) [Hint]

When generators are called they return a generator function?

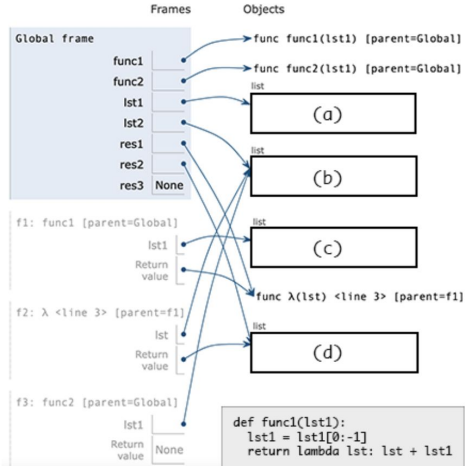
Question 3 (Generators) [Hint]

Trees are a recursive data structure!!

Exam Level Question

1. (5.0 points) Mutation Station

The environment diagram below was generated by code that is provided at the bottom right of the diagram. The diagram represents the full execution of the code.




```
def func1(lst1):
  lst1 = lst1[0:-1]
  return lambda lst: lst + lst1

def func2(lst1):
  lst1[0] = lst1.pop()

lst1 = [1, 19, 2022]
lst2 = [5, 10, 2022]

res1 = func1(lst1)
res2 = res1(lst2)
res3 = func2(lst2)
```

Exam Level Solution

(a) **(1.0 pt)** Fill in blank (a).

```
[1, 19, 2022]
```

(b) **(1.0 pt)** Fill in blank (b).

```
[2022, 10]
```

(c) **(1.0 pt)** Fill in blank (c).

```
[1, 19]
```

(d) **(2.0 pt)** Fill in blank (d).

```
[5, 10, 2022, 1, 19]
```

Anonymous Feedback Form