6.101 Recitation 6: Week 2 Image Processing 2 Wrap-up: Iterable Properties 2/26/24

This sheet is yours to keep!

"linear" means the operation gets proportionally slower as the iterable grows
"constant" means the operation takes roughly the same amount of time regardless of the size of the iterable
"hashable" means the object is immutable and all nested elements (if they exist) are also immutable. Ex: t = ([1], .5) →t is immutable but not hashable because its inner list is mutable.

Type	**list**	**tuple**	**set**	**frozenset**	**dict**
Description	A mutable sequence (collection of ordered arbitrary objects)	An immutable sequence	A mutable unordered collection of distinct hashable objects	An immutable and hashable collection of distinct hashable objects	A mapping from hashable keys to arbitrary object values.
Example?	`lst = [[1], .5, "hi"]`	`tup = ([1], .5, "hi")` `tup = 5, 4` `#()optional`	`s = {(1,), .5, "hi"}`	`fs = frozenset({(1,), .5, "hi"})`	`d = { (1,): [1], "hi": .5 }`
Contains what type of elements?					
falsey value?					
Is it ordered?					
mutable?					
hashable?					
indexable? x[…]					
Adding elements?					
Removing elements?					
Containment check? y in x					

Question 1: Fill in the body of first_occurrence below.

```
def first_occurrence(data):
    """
    Given a list of integers or strings, return a new list with the same
    set of items in the same order, but keeping only the first occurrence of
    each item.

    Example: first_occurrence([1, 9, 1, 1, 5, 3, 2, 9, 10]) == [1, 9, 5, 3, 2, 10]
    """
```

Question 2: Fill in the body of how_old below.

```
def how_old(data):
    """
    Given a list of integers or strings, returns a list of the same length
    where the ith entry is the distance of the ith entry in the input list
    to the last occurrence of the same value in the input list,
    or None if there was no previous occurrence.

    Example: how_old([1, 2, 1, 1, 2]) == [None, None, 2, 1, 3]
    """
```

R6 Participation Credit **Kerberos : ________________@mit.edu**

Hand this sheet in at the end of recitation to get participation credit for today.

Question 0: The following code creates a large ordered list and a large set and then times how large it takes to find the largest 1,000 numbers in both. For each timed example, indicate in the provided box which option best matches how long the code will take to execute.

1: << 0.1 seconds
2: ~ 0.1 seconds
3: ~ 1 second
4: ~ 10 seconds
5: >> 10 seconds

```
import time

big_num = 10_000_000  # ten million
big_num_list = list(range(big_num))
big_num_set = set(big_num_list)

small_num = 1000
small_num_list = list(range(big_num - small_num, big_num))


start = time.time()
count = 0
print("counting 1000 largest numbers in a set...")
for i in small_num_list:
    count += i in big_num_set
end = time.time()
print("count using set:", count, "time:", end - start, "sec")
```

Set Guess:

Set Actual:

```
start = time.time()
count = 0
print("counting last 1000 numbers in a list...")
for i in small_num_list:
    count += i in big_num_list
end = time.time()
print("count using list:", count, "time:", end - start, "sec")
```

List Guess:

List Actual: