CSCI 141

Lecture 23
Lists and Dictionaries Review
Reading and Writing Files

Announcements

- A5 Code and A5 Written are due Friday.
- Slip days still apply, **but**:
  - **No late submissions accepted after Tuesday 6/4 at 10pm**
- Now is the time to start studying for the final exam.

Goals

- Know how to modify lists using the following: `insert`, `remove`, `del`
- Know the basics of how to use dictionaries (dicts):
  - Creation, assignment, indexing
  - in, del, iterating over keys and values
- Know the basics of file input/output:
  - Reading - iterating over lines, `read`, `readlines`
  - Writing - `write` method

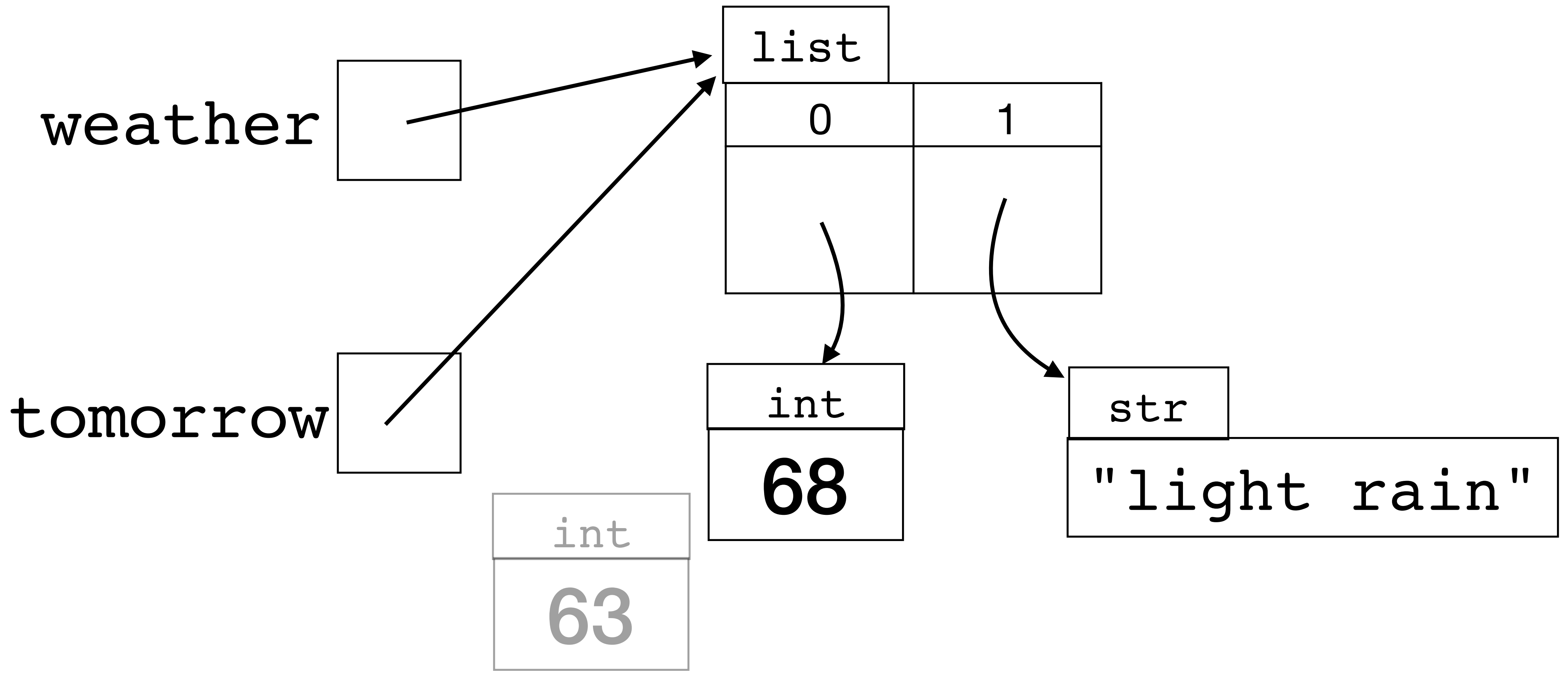
Last Time

- Understand the implications of variables holding references to mutable objects

Implications of Mutability

```
weather = [63, "light rain"]
tomorrow = weather
tomorrow[0] = 68
print(weather[0])
```

State after the above is executed:



Mutable Objects and Functions

```python
def z0(y):
    y[0] = 4
    return y

b = [5, 6]
c = z0(b)
print(b[0], c[0])
```

What does this code print?

```
A.  4 4
B.  4 5
C.  5 4
D.  5 5
```

Last Time

- Know the basics of how to use dictionaries (dicts):
  - Creation:
    `d = {`*`key1`*`: `*`value1`*`, `*`key2`*`: `*`value2`*`, ...}`
  - Access:
    `d[`*`key`*`] # => `*`value`*, or **error** if key not in d
  - Assignment:
    `d[`*`key`*`] = `*`new_value`*

Today’s Quiz

- 3 minutes

Today's Quiz

- 3 minutes
- Working with a neighbor: do your answers agree? (2 minutes)

A few more list operations:

```
my_list.index(value)
```
Return the index of value in my_list
Throw an error if value is not in my_list.

```
my_list.insert(index, value)
```
Inserts value into my_list at index, shifting all following elements on spot to the right.

```
my_list.remove(value)
```
Removes the first item from the list whose value is equal to *value*.
Causes an error if value is not in my_list.

```
del my_list[index]
```
Removes the element at index, shifting all following elements one spot to the left.

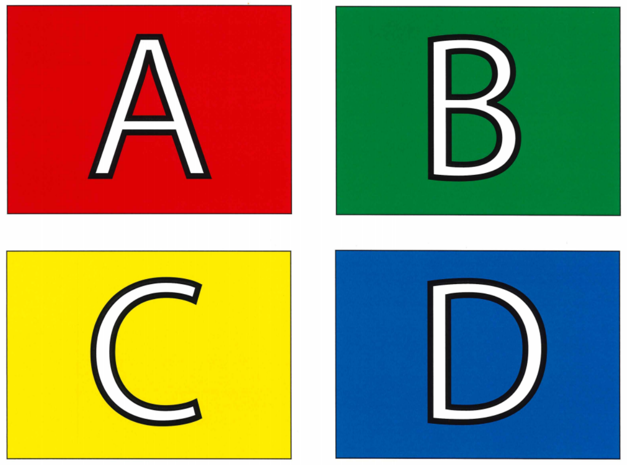
index, insert, remove, del:

Live coding example (if time):

```python
def the_snap(avengers):
    """ Remove a randomly chosen half of the
        elements from the given list of avengers
    """
```

What does this print?

```python
a = []
b = [1]
a.insert(0, b)
b[0] = 4
a.insert(0, b)
print(a)
```



```
A.  [1, 4]
B.  [4, 4]
C.  [[1], [4]]
D.  [[4], [4]]
```

Demo

```
b = [1]
a.insert(0, b)
b[0] = 4
a.insert(0, 4)
print(a)

del b[0]
print(a)
```

```
A.  [1, 4]
B.  [4, 4]
C.  [[1], [4]]
D.  [[4], [4]]
```

Dictionaries: TL;DR

- Creation:
  `d = {`*`key1`*`: `*`value1`*`, `*`key2`*`: `*`value2`*`, ...}`

- Access:
  `d[`*`key`*`] # => `*`value`*, or **error** if key not in d
  `d.get(`*`key`*`) # => `*`value`*, *or* ***`None`*** *if key not in d*
  `d.get(`*`key, alt`*`) # => `*`value`*, *or* ***`alt`*** *if key not in d*

- Assignment:
  `d[`*`key`*`] = `*`new_value`*

- Membership:
  `key`` in d # => True` if `d[`*`key`*`]` exists

- Removal:
  `del d[`*`key`*`] # deletes key and its associated value`

Worksheet - Exercise 2

```
def count(values):
    """ Return a dictionary that maps each element of values to
        the number of times it appears in the list.
        Precondition: values is a list of immutable objects """
```

- Creation:
  `d = {key1: value1, key2: value2, ...}`

- Access:
  `d[key] # =>  value`, or **error** if key not in d
  `d.get(key) # =>  value`, *or* ***None*** *if key not in d*
  `d.get(key, alt) # =>  value`, *or* ***alt*** *if key not in d*

- Assignment:
  `d[key] = new_value`

- Membership:
  `key in d # => True` if `d[key]` exists

Dictionaries: Iterating

```
d = {key1: value1, key2: value2, ...}

for key in d:
    print(key)

for key in d.keys():
    print(key)

for val in d.values():
    print(val)

for (key, val) in d.items():
    print(key, val, sep=": ")
```

Note: Like `range`, these methods return sequences that are not lists.
To get a list of values use `list(d.values())`

Worksheet - Exercise 3

```
def mode(values):
    """ Return the most frequently-appearing value in values,
        or one of the most frequent values in case of a tie.
        Precondition: values is a list of immutable objects
    """
```

- Hint: use your count function, then find the **key** whose **value** is largest.