On your way in...

Pick-up:
1. Homework 7

Drop-off:
1. Homework 6 on the side table (two piles)
Welcome to CS 134!

Introduction to Computer Science
Iris Howley

-Hashing & Linked Lists-

Spring 2019
HASHING

Finding dictionary values quickly
Dictionary Keys

- `d[['bill l','bill j']] = 'williams college'`
  - ERROR
- `d[('bill l','bill j')] = 'williams college'`
- `d = {}(('bill l', 'bill j'): 'williams college')`

What’s the difference?

Dictionary keys must be immutable types
- int, float, string, bool, tuple, frozenset
Dictionary Keys

**Dictionary keys must be immutable types**

int, float, string, bool, tuple, frozenset

**Why?**
Mutable Types as Dictionary Keys

- Lists are mutable
- When you append() to a list, it changes that list object
- If you used a list object as a key in a dictionary, you wouldn’t be able to find it again, after it’s been changed

```python
mylist = ['a', 'b']
mydict = dict()
mydict[mylist] = 'throws an error'
mylist.append('c')
print(mydict[mylist])
# Now mylist is no longer findable in the dict!
```

We’re going to see why!
Dictionary Keys

- Dictionaries index their items by a hash
- A hash is an fixed sized integer that identifies a particular value.
- Each value needs to have its own hash
  - For the same value you will get the same hash even if it's not the same object.

Why not just index items based on their value?
Hashing
Hashing FIND:
Hashing
Hashing

FIND:
Hashing

FIND:
Hashing

Why not just index items based on their value?

• We could organize all words in memory by the letter they start with...

• But words that start with ‘A’ could be numerous

• Compared to words that start with ‘Z’
  ▪ ...Sort of like arranging clothes by color

• Hashing is a different way of mapping items to make them easier to find
Hashing

• Other concerns
  ▪ Bad hashing function for your data, resulting in clustering
  ▪ Running out of space in the pile you’ve assigned
  ▪ Placing shirts in the wrong pile

• Stored in the order that makes it easiest to look them up
hash(o) → o.__hash__()  

• s = "hello world"
• t = s + "!"
• hash(s) → 4960501519247167238
• hash(s) → 4960501519247167238
• hash(t) → -8774050965770600213
• hash(t[::−1]) → 4960501519247167238

If the 2 strings are the same, they’ll get the same hash
If the 2 strings are different, they *might* get a different hash.
\[ \text{hash}(o) \rightarrow o.\_\_\text{hash}\_\_() \]

Some hash codes are expensive (million-long tuple)

- \[ \text{hash}(1) \rightarrow 1 \]
- \[ \text{hash}(2) \rightarrow 2 \]
- \[ \text{hash}(1000000000000000000) \rightarrow 1000000000000000000 \]
- \[ \text{hash}(10000000000000000000) \rightarrow 776627963145224196 \]

At some length, it starts treating the numbers like a string
If the hash codes are the same, the values might be the same
Hash Tables

How to access mydict[‘wally’]?

<table>
<thead>
<tr>
<th>Keys</th>
<th>Hashes</th>
<th>Buckets</th>
<th>Overflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘pixel’</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>‘tally’</td>
<td>1</td>
<td>tally bananas</td>
<td></td>
</tr>
<tr>
<td>‘wally’</td>
<td>2</td>
<td>linus everything</td>
<td></td>
</tr>
<tr>
<td>‘linus’</td>
<td>3</td>
<td>pixel cheese</td>
<td></td>
</tr>
</tbody>
</table>

collision!

Overflow

What to do with Wally?

Could re-hash into new table and increase # buckets...

...or...
Immutable Objects

- Have no way to set/change the attributes, without creating a new object
  - Like `int`, `string`, etc.
  - Like the `Color` class from this week’s lab!
  - `__slots__ = []`

- Can be used in sets
  - i.e., you cannot have a set of lists

- Can be used as keys for dictionaries
  - If the class has a `__hash__()` function defined!
MAKING OUR OWN DATA STRUCTURES

Classes, Part IV
What is a list?

class list(object)
    list() -> new empty list
    list(iterable) -> new list initialized from iterable's items

Methods defined here:

    __add__(self, value, /)
        Return self+value.

    __contains__(self, key, /)
        Return key in self.

    __delitem__(self, key, /)
        Delete self[key].

    __eq__(self, value, /)
What is a list?
What is a list?

What is the last elephant holding onto?

None
What is a list?

class Element:

_value

3

_next

_value

15

_next

_value

2019

_next
Linked Lists

- See example code in shared/examples/03.15!
- Lecture notes from that day are also useful!
class LinkedList:

    _head

class Element:

    _value

    _next

    _value

    _next

    _value

    _next
QUESTIONS?
Leftover Slides
Hashing

• Don’t know how it’s computed → Abstraction

• There’s many ways to implement a hash function, here’s a description of some of them:
Tuples, Strings, other built-in types aren’t particularly special!

You can build your own!
Thought question:
How would you build a doubly-linked list?