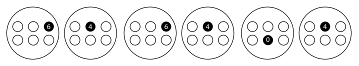
Computer Science 134C: Introduction to Computer Science — Spring 2019

Iris Howley
`iris@cs.williams.edu`

Office:	TCL 308.
Office Hours:	Iris (TCL308): Tues. TBD, Wed. 12:30-2:30p, Thurs. 1-2:30p
	Duane (TPL306): Mon. 2:30-4:30p, Tues. 7:30-9p, Thurs. 9:45-11:20a
Co-instructor:	Duane A. Bailey (`bailey@cs.williams.edu`), TPL 306.
Assistants:	Noah Andrew, Chris Anton, Will Burford, Jimmy DeLano, Jacob Justh, Julia Kawano,
	Aidan Lloyd-Tucker, Grace Mazzarella, Nevyn Neal, Nathan Thimothe, Alex Trevithick, Linda Zeng
TA Hours:	Sun. 4-9:30pm, Mon-Thu. 7-10pm, +Wed. 2-4p (in TPL312), +Wed. 4:30-6:30pm, Thu. 6-10p., + Thu. 4-5:30p
Text:	Allen Downey's *Think Python, 2ed*, at `greenteapress.com/thinkpython2/thinkpython2.pdf`.
Web resources:	`cs.williams.edu/~bailey/cs134`
Technical Support:	Mary Bailey (`mary@cs.williams.edu`), TCL 312.
Lecture:	Chemistry 123, Monday, Wednesday, and Friday at 11:00 a.m.
Lab Times:	Mon 1-2:30pm (Iris), 2:30-4pm (Iris), Tue 10-11:30am (D), 1-2:30pm (D), 2:30-4pm (D)
Lab Location:	TCL 217a
CS Lab Code:	6-4-6-4-0-4 (remember visually, or **think**: 8^2, 8^2, 2^2).



We are surrounded by information. This course introduces fundamental computational concepts for representing and manipulating data. Using the programming language Python, this course explores effective ways to organize and transform information in order to solve problems. Students will learn to design algorithms to search, sort, and manipulate data in application areas like text and image processing, scientific computing, and databases. Programming topics covered include procedural, object-oriented, and functional programming, control structures, structural self-reference, arrays, lists, streams, dictionaries, and data abstraction. This course is appropriate for all students who want to create software and learn computational techniques for manipulating and analyzing data.

Organization. During lecture hours we will typically learn new concepts through the building of new tools to solve simple problems. While the learning process is initially supported by an online text, we expect a dynamic approach to the class that will allow us to steer lectures in directions of mutual interest. During formal lab hours, we will meet for 90 minutes to begin work on a more extended problem. We expect that this work will be continued outside of scheduled time. There are also weekly written homework assignments to support lecture and lab learning.

Work. You are responsible for reading supporting material (*Think Python* (TP)) and participating as the semester progresses. In addition, some topics may require you to investigate online resources (documentation, tutorials, and the like). Each week you will be responsible for completing a programming assignment (35%) in addition to a written homework (15%). **There will be a midterm examination on March 5** (25%), and a scheduled final (T.B.A., 25%). We reserve the right to adjust grades by as much as 5% to reflect course participation.

Week of	**Monday**	LAB	**Wednesday**	**Friday**
Feb. 1	—		—	1. Hello, world! (TP1)
Feb. 4	2. Expressions (TP2)	I. PYTHON AND GIT	3. Functions (TP3)	4. Conditions (TP5-6)
Feb. 11	5. Abstraction (TP4)	II. PROCEDURE	6. Iteration (TP7)	*Winter Carnival*
Feb. 18	7. Strings (TP8-9)	III. TOOLBOX BUILDING	8. Interpretation	9. Lists, Tuples (TP10,12)
Feb. 25	10. Sets, Dicts, (TP11)	IV. FACULTY TRIVIA	11. Files (TP14)	12. Generators
Mar. 4	13. Iterators	V. PRESENTING DATA	14. *Slack*	15. Classes (TP15-17)
Mar. 11	Classes & n-grams	VI. GENERATORS	16. Properties	17. Data Structures
M. 18&25	*Spring Break*	*Spring Break*	*Spring Break*	*Spring Break*
Apr. 1	18. Images	VII. IMAGES	19. Hashing	20. Linked Lists
Apr. 8	21. Lambda Sorting	VII. VIZ ANALYSIS	22. Big-O Notation	23. Sorting I
Apr. 15	24. Sorting II.	VIII. JUPYTER	25. HTML & Hex	26. Binary Trees I.
Apr. 22	27. Binary Trees II.	IX. MOUNTAIN DAY	28. Object Persistence	29. Java I.
Apr. 29	30. Java II.	X. JAVA	31. Java III.	32. Java IV.
May 6	33. *Slack*	X. JAVA (CONT.)	34. *Slack*	35. Evaluations

Comments from previous renditions

"1. Go to office hours, 2. Go to TA sessions, 3. Don't stress about homeworks."
"Go to office hours! GO TO OFFICE HOURS! Go to office hours!" ⋆ "Read the textbook."
"Look at the code Duane posts after class; don't try to copy it down in class." ⋆ "THINK about how your code should works logically before typing anything." ⋆ "Stop complaining and start coding!!! Feel swervy!"
"Don't be intimidated...a programming language is just a language...practice the idioms."
"Practice writing code outside of class." ⋆ "Write code on paper beforehand; it helps to pinpoint errors."
"TAs are soooo helpful and just great to talk to." ⋆ "You are learning a lot...Enjoy!" ⋆ "Stay swervy."

Intellectual Property. No part of this course may be reproduced and distributed in any manner without prior permission from the instructors.
Community. We embrace diversity. We welcome all students and expect everyone to contribute and support a respectful and welcoming environment. If you have concerns, please share them with us or the college administration.
Students Who Need Accommodations. If formal accommodations need to be made to meet your specific learning or physical abilities, please contact one of us as soon as possible to discuss appropriate accommodations. Please also contact the Director of Accessible Education, Dr. G. L. Wallace (4135974672) or the Dean's office (4135974171). We will work together to ensure this class is as accessible and inclusive as possible.
Mental Health. Students experiencing mental or physical health challenges that are significantly affecting their academic work are encouraged to contact one of us or to speak with a dean. The deans can be reached at 4135974171.
Honor Code. The Honor Code as it applies to non-programming assignments is outlined in the Student Handbook.

For programming assignments in computer science courses, the honor code is interpreted in very specific ways. When a program is assigned, it will be described as a "test" or "laboratory" program. The Honor Code applies to each as follows (unless otherwise specified):

TEST PROGRAMS. Any assignment designated as a test program is to be treated exactly as a take-home, open-book test. You are allowed to read your textbook, class notes, and any other source approved by your instructor. You may not consult anyone other than your instructor. The instructor encourages the asking of questions, but reserves the right not to answer, just as you would expect during an exam.
Guideline: Any work that is not your own is considered a violation of the Honor Code.

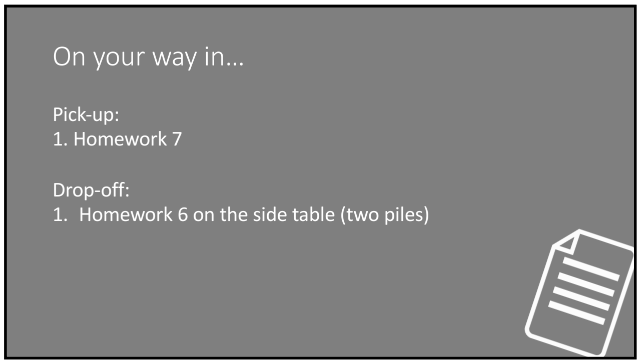
LABORATORY PROGRAMS. Laboratory programs are expected to be the work of the individual student, designed and coded by him or her alone. Help locating errors and interpreting error messages are allowed, but a student may only receive help in correcting errors of syntax; help in correcting errors of logic is strictly forbidden. In general, if you are taking photos of someone else's screen, looking at someone else's screen, or telling someone else what to type, it is likely your work is no longer the work of an individual student.
Guideline: Assistance in the design or coding of program logic will be considered a violation of the Honor Code.

If you do not understand how the Honor Code applies to a particular assignment, consult your instructor. Students should be aware of the Computer Ethics outlined in the Student Handbook. Violations (including uninvited access to private information and malicious tampering with or theft of computer equipment or software) are subject to disciplinary action.
Guideline: To protect your work dispose of printouts and copies of your work carefully, and avoid leaving your programs on hard disks in labs and other public storage areas.

The Department of Computer Science takes the Honor Code seriously.
Violations are easy to identify and will be dealt with promptly.

The College and Department also have computer usage policies that apply to courses that make use of computers. You can read more about these policies at
`csci.williams.edu/the-cs-honor-code-and-computer-usage-policy`

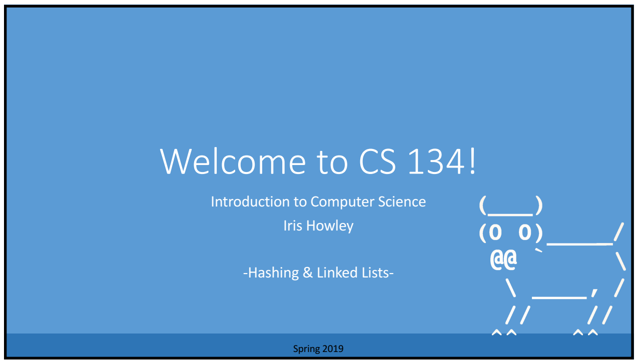
Anonymous ID. We grade anonymously. When asked, your Anonymous ID (AID) is:



On your way in…

Pick-up:
1. Homework 7

Drop-off:
1. Homework 6 on the side table (two piles)



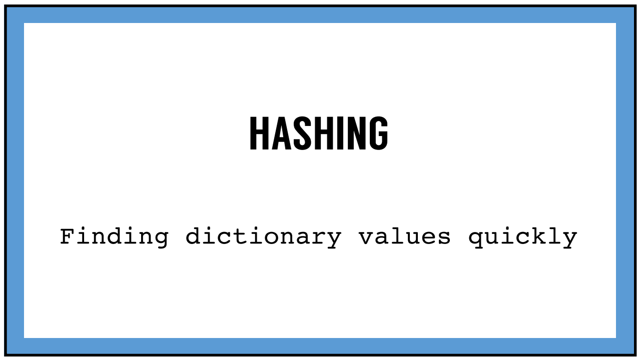
Welcome to CS 134!

Introduction to Computer Science
Iris Howley

-Hashing & Linked Lists-

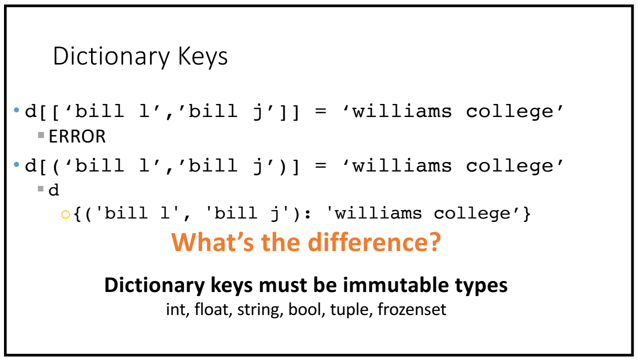
```
(___)
(o o)_____/
 @@ `     \
  \ _____, /
  //    //
 ^^    ^^
```

Spring 2019



HASHING

`Finding dictionary values quickly`

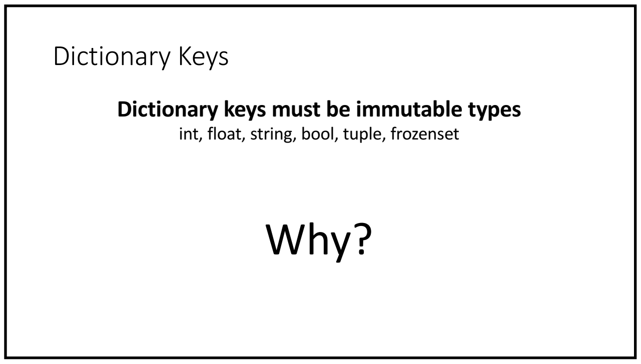


Dictionary Keys

- `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?

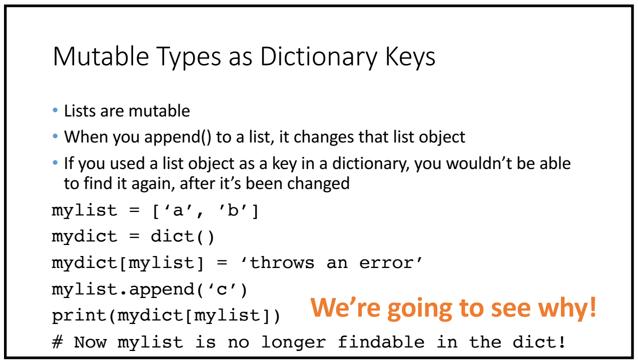
Dictionary keys must be immutable types
int, float, string, bool, tuple, frozenset



Dictionary Keys

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

Why?

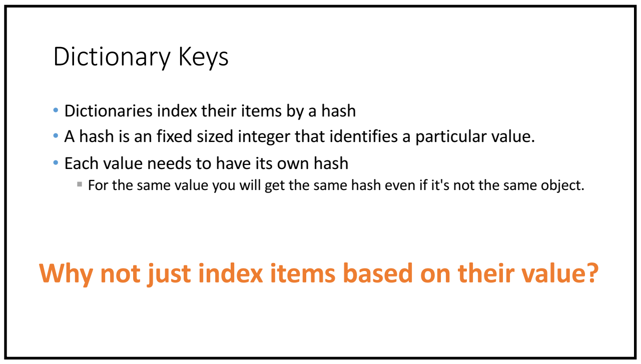


Mutable Types as Dictionary Keys

- 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

```
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!
```

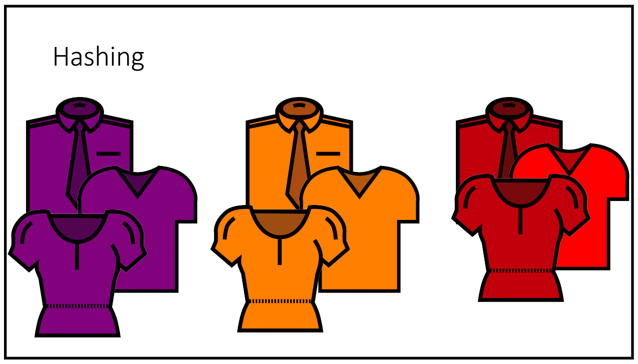
We’re going to see why!



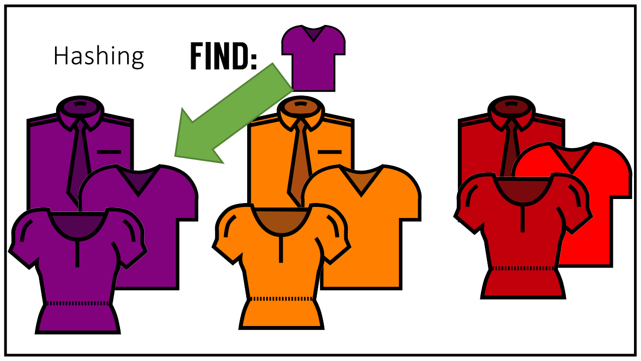
Dictionary Keys

- 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?

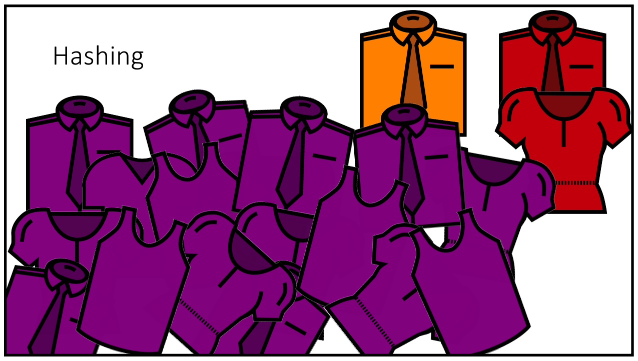


Hashing

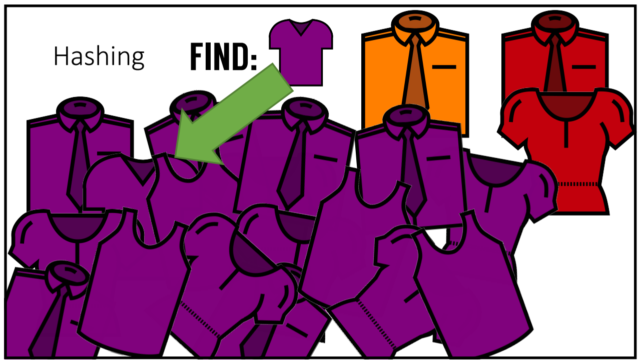


Hashing

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.

hash(o) → o.__hash__()

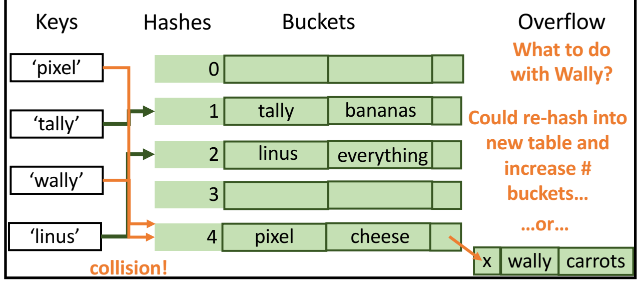
Some hash codes are expensive (million-long tuple)

- `hash(1)` → `1`
- `hash(2)` → `2`
- `hash(1000000000000000000)` → `1000000000000000000`
- `hash(10000000000000000000)` → `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’]?



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!

QUESTIONS?

Leftover Slides

Hashing

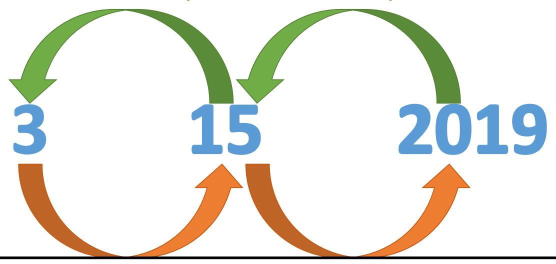
- Don't know how it's computed → Abstraction
- There's many ways to implement a hash function, here's a description of some of them:
  - https://www.cs.hmc.edu/~geoff/classes/hmc.cs070.200101/homework10/hashfuncs.html

`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?



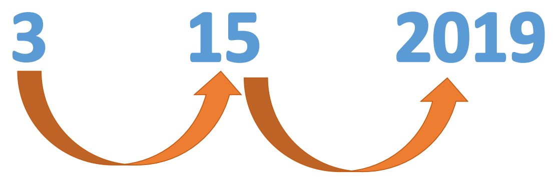
MAKING OUR OWN DATA STRUCTURES

`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, /)
```

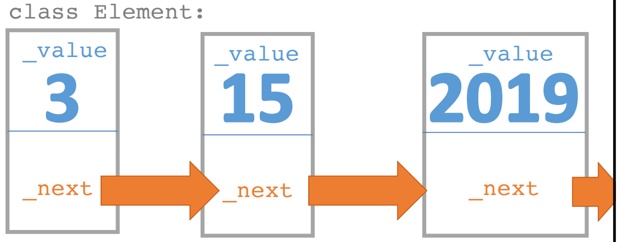
What is a list?



What is a list?

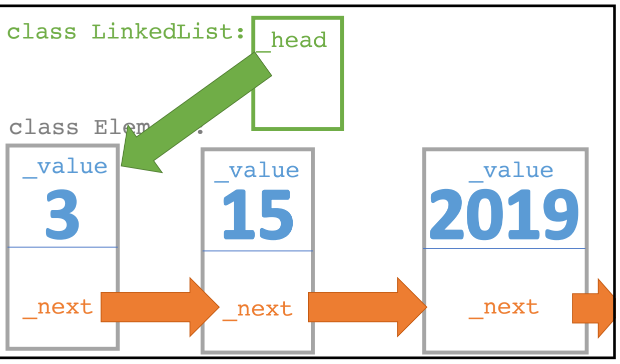
3
15
2019
What is the last elephant holding onto?
None

What is a list?



Linked Lists

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


class LinkedList:
_head
class Elem
_value
3
_next
_value
15
_next
_value
2019
_next