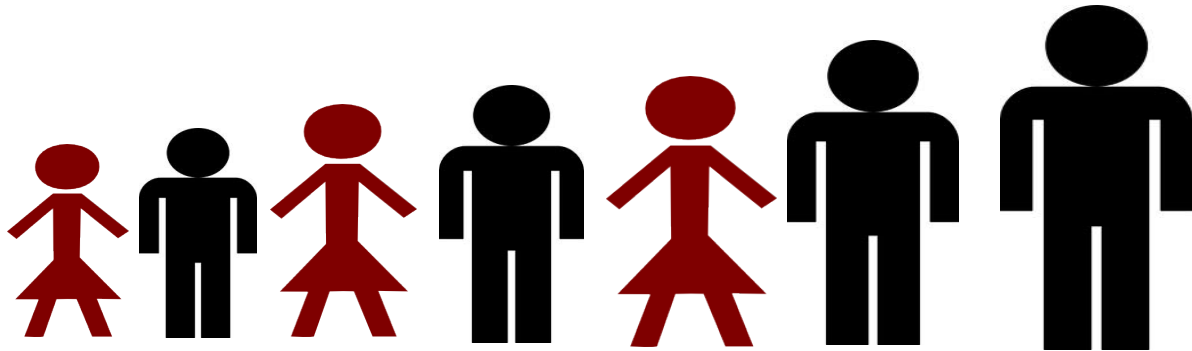


Compsci 101

Stable Sorting, Lambda, Clever Hangman

Live Lecture



Announcements

- Assignment 4 Hangman due today!
- APT-6 due Thurs. April 1
- APT-7 out Thurs. April 1
- Assignment 5 Clever Hangman due Tues. April 6
- Lab 9 this Friday
 - There is a prelab, it is out!

More Announcements

- APT Quiz 2 out Thurs. April 8
- Exam 3 Tues. April 13
 - Old exams up on the old tests page

Piazza

- Use it to study for APT Quiz 2 and Exam 3
- Answer questions if you know the answer
 - This is one data point we use in selecting new UTAs for CompSci 101

Taraneh BigBow

- Software Engineer, Backend Development
 - Apple
 - BigBow Technologies
 - Microsoft
- Oklahoma Native
 - Kiowa Tribe
- “Advice that I tell myself is, ‘There is a way.’ Even if they say no, there is a way.”



PFTD

- **Sorting in Python and sorting in general**
 - How to use `.sort` and `sorted`, differences
 - Key function – change how sorting works
 - Lambda – create anonymous functions
- **Stable sorting**
 - How to leverage when solving problems
 - Why Timsort is the sort-of-choice (! quicksort)
- **Clever Hangman –**
 - How does it work? Greedy Algorithm

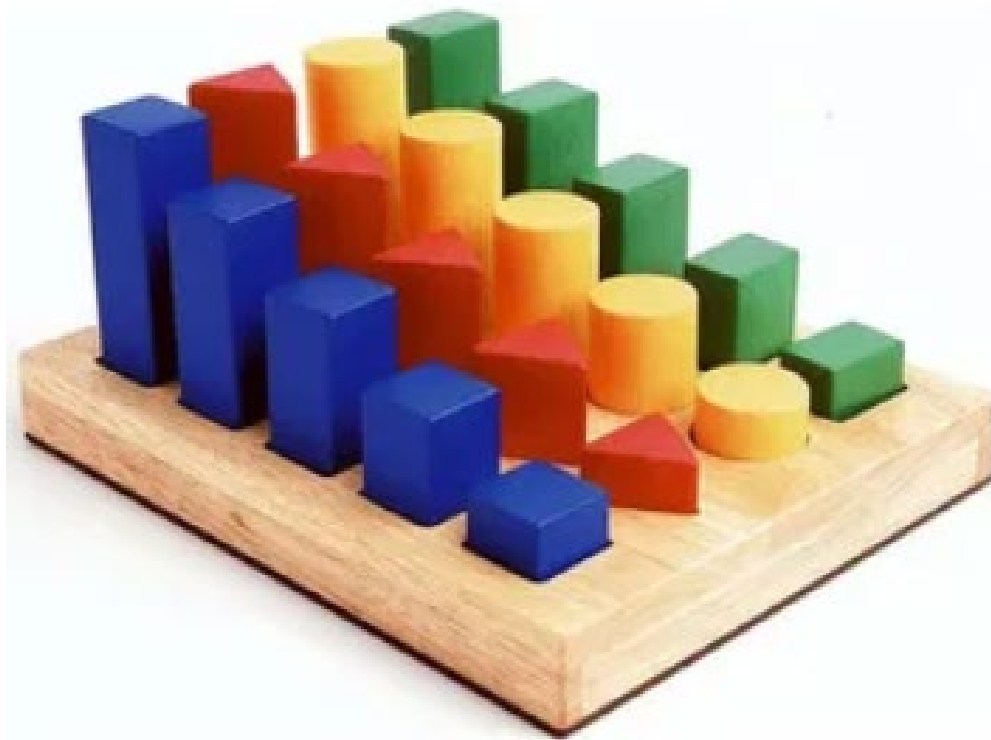
Review: Syntax and Semantics of Lambda

- Major use: single variable function as key

```
>>> fruits = ["banana", "apple", "lemon", "kiwi", "pineapple"]
>>> sorted(fruits)
['apple', 'banana', 'kiwi', 'lemon', 'pineapple']
>>> min(fruits)
'apple'
>>> max(fruits)
'pineapple'
>>> min(fruits, key=lambda f: len(f))
'kiwi'
>>> max(fruits, key=lambda z: z.count("e"))
'pineapple'
>>> sorted(fruits, key=lambda z: z.count("e"))
['banana', 'kiwi', 'apple', 'lemon', 'pineapple']
```

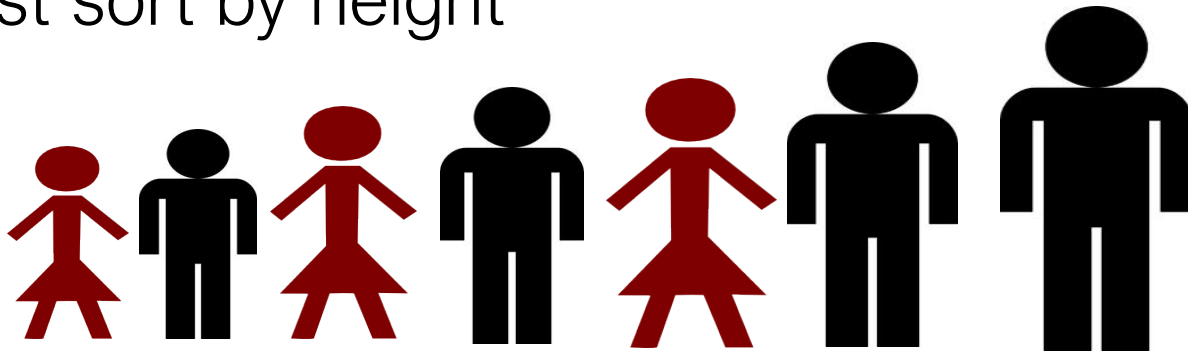
WOTO-1 Sorting

<http://bit.ly/101s21-0330-1>



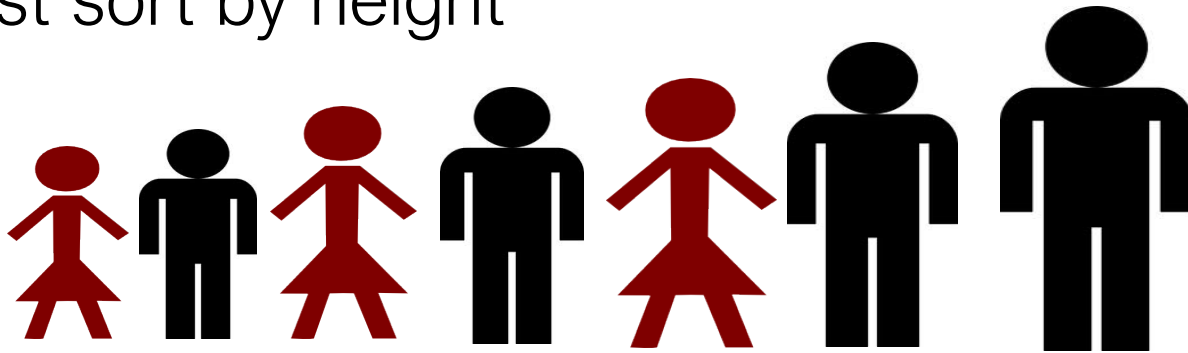
Review: Stable sorting: respect "equal" items

- Female before male, each group height-sorted
 - First sort by height



Stable sorting: respect "equal" items

- Female before male, each group height-sorted
 - First sort by height



- Then sort by gender



Review: Understanding Multiple-Pass Sorting

```
> a0 = sorted(data, key = lambda x: x[0])
> a1 = sorted(a0, key = lambda x: x[2])
> a2 = sorted(a1, key = lambda x: x[1])
> a0
[('a', 2, 0), ('b', 3, 0), ('c', 2, 5),
 ('d', 2, 4), ('e', 1, 4), ('f', 2, 0)]
> a1
[('a', 2, 0), ('b', 3, 0), ('f', 2, 0),
 ('d', 2, 4), ('e', 1, 4), ('c', 2, 5)]
> a2
[('e', 1, 4), ('a', 2, 0), ('f', 2, 0),
 ('d', 2, 4), ('c', 2, 5), ('b', 3, 0)]
```

WOTO-2 Multipass Sorting

<http://bit.ly/101s21-0330-2>



Use paper, help your brain

- unpack from inside out

```
sorted(sorted(sorted(lst,key=sum),key=min),key=max)
```

```
In[4]: lst
```

```
Out[4]: [[4, 6, 7], [5, 2], [3, 9], [6, 2, 9]]
```

```
In[5]: x = sorted(lst,key=sum)
```

```
In[6]: x
```

```
Out[6]: [[5, 2], [3, 9], [4, 6, 7], [6, 2, 9]]
```

```
In[7]: y = sorted(x,key=min)
```

```
In[8]: y
```

```
Out[8]: [[5, 2], [6, 2, 9], [3, 9], [4, 6, 7]]
```

```
In[9]: z = sorted(y,key=max)
```

```
In[10]: z
```

```
Out[10]: [[5, 2], [4, 6, 7], [6, 2, 9], [3, 9]]
```

Can we “move the goalpost?”

- When playing Hangman?
 - Never
 - Perfectly fine, just being clever! 
- See also: <http://blog.wolfram.com/2010/08/13/25-best-hangman-words/>
 - Hard words? "jazziest", "joking", "bowwowing"

Clever Hangman

- **Current Hangman: Pick random secret word**
 - Don't mislead the guesser, don't say "oh! I forgot, there is an 'a' in the word !
- **Can you change secret word: user oblivious?**
 - Given a user's letter, can change secret word!?
 - Change consistent with all guesses
 - Make the user work harder to guess!

Programming A Clever Game

- Instead of guessing a word, you're guessing a *group*, *category*, or *equivalence class* of words

Ex: _ _ _ _ _ and user guesses 'a'

- **["asked", "adult", "aided", ... "axiom"]**
 - 209 words 'a' as first letter and the only 'a'
- **["baked", "cacti", "false", ... "walls"]**
 - 665 words 'a' as second letter and the only 'a'
- **["beets", "humor", ... "spoof"]**
 - 2,431 words with no 'a'
- What should our secret word be? "asked", "baked" or "beets"?

Sometimes there will be letters

- The letter “u” has been guessed and is the 2nd letter
Ex: _ u _ _ _ and user guesses ‘r’
- [**"ruddy"**, **"rummy"**, **"rungs"**, ... **"rusty"**]
 - *5 words start with “ru” and no other “r” or “u”*
- [**"burch"**, **"burly"**, **"burns"**, ... **"turns"**]
 - *17 words only ‘u’ as second letter and only ‘r’ third letter*
- [**"bucks"**, **"bucky"**, ... **"tufts"**]
 - *98 words with only “u” second letter and no ‘r’*
- What should our secret word be? "ruddy" ,"burch" or "bucks"?

More Details on Game

- Pick 8-letter word at random: *catalyst*
 - User guesses 'a', what should computer do?
 - Print `_ a _ a _ _ _ _` and continue?
- Look at all groups of words and decide on a new word that is more likely to stump player
 - Why “*designed*” better choice than “*tradeoff*”?
 - 3,475 words with no 'a', 498 with 'a' 3rd letter

Creating Groups/Categories

- For each of 7,070 words (8 letters), given word and 'a', find its group, represented by a template
- Use dictionary
 - Template is KEY, the VALUE is a list of matching words

- Choose biggest list
- Repeat
- # words smaller over time

Group/Template	Size of Group
_ a _ _ _ _ _ _	587
_ a _ a _ _ _ _	63
_ _ a _ _ _ _ _	498
_ _ _ a _ _ _ _	406
_ _ _ _ _ _ _ _	3,475

Changes to Regular Hangman

- List of words from which secret word chosen
 - Initially this is all words of specified length
 - User will specify the length of the word to guess
 - After each guess, word list is a new subset
- Keep some functions, modify some, write new ones
- *Changes go in another function* to minimize changes to working program
 - Minimizing changes helps minimize introducing bugs into a working program

Shall we play a game?

- Can you guess a six letter word with ≤ 8 misses
 - *Regular* hangman
 - *Clever* hangman
 - Even with debugging on!!
- Fun?



Details from Assignment

- We've missed four times, what's happening?
 - "belted" is one of 20 words that fit guesses

```
letters not yet guessed:  bc  fgh jklmn pq  t vwxyz
misses remaining = 2
_ e _ _ e d
(word is belted)
# possible words: 20
letter> l
_e__ed : 10
_el_ed : 4
_elled : 5
le__ed : 1
# keys = 4
you missed: l not in word
```

Greedy Algorithms

- “Choosing largest group” -> *greedy algorithm*
 - Make a locally optimal decision that works in the long run
 - Choose largest group to make game last ...
- Greed as in “it chooses the best current choice every time, which results in getting the best overall result”
- Canonical example? Change with coins
 - Minimize # coins given for change: 57 cents

Making change for 57 cents

- When choose next coin, always pick biggest
- With half-dollar coins



- With quarters and no half dollars



When greedy doesn't work

- What if no nickels? Making change for 31 cents:




- Can we do better? Yes!



Woto-3 Clever Hangman

<http://bit.ly/101f20-1020-3>

DID
NOT
GET
TO

 **Hangman** [Play](#) [How-to](#) [Words](#) [Create](#)

Hangman Words

Want to frustrate your friends? Use these techniques to pick a good hangman word or just pick from the list of words that have proved to be the most challenging to guess.

Hard Hangman Words:

• abruptly	• absurd	• abyss	• affix
• askew	• avenue	• awkward	• axiom
• azure	• bagpipes	• bandwagon	• banjo
• bayou	• beekeeper	• bikini	• blitz
• blizzard	• boggle	• bookworm	• boxcar