CompSci 101
Fall 2021

Lecture 17

Reminders

- Assignments
  - APT 6 due

- APT Quiz 2
  - 11/12 (8am) -11/15 (11pm)

Key instructions

- Input ✔
- Output ✔
- Assignments* ✔
- Math/Logic ✔
- Conditionals ✔
- Repetition ✔

**not listed in book*

Python Data Types

- int, float, bool ✔
- Collections
  - Strings ✔
  - Lists ✔
  - Tuples ✔
  - Sets ✔
  - Dictionaries ✔

PFTD

- Sorting cont.
  - Key function
  - Lambda
- Stable sorting
- Greedy Algorithms
- Clever Hangman

KISS Principle

- Think of the non-computing context for any word/terms
- KISS model
  - Work smarter, not harder!!
- "Good programmers are simply good designers."
  - *-Dr. Washington*
- Design first and always!
- Importance of reusability
- *USE PyCharm/PythonTutor IF YOU HAVE QUESTIONS!*

People to Know: Dr. Kylie Ariel Bemis

- BS/MS/PhD (Purdue)
- Lecturer (Northeastern)
  - Data science
  - Machine learning and large-scale statistical computing for bioinformatics.
- Author
  - Fiction and poetry
- Zuni tribe

Top 5 Artists

- Instead of intermediary list, use **`lambda`**
- Instead of **`[-5:]`**, use **`reverse=True`**

```
31      print('\nTop 5 artists:')
32      sortedArtists = sorted([(a[1], a[0]) for a in counts.items()])
33      sortedArtists = [(a[1], a[0]) for a in sortedArtists]
34      for artist in sortedArtists[-5:]:
35          print(artist)
36
37      sortedArtists = sorted(counts.items(),
38                             key = lambda item: item[1], reverse=True)
39      for artist in sortedArtists[:5]:
40          print(artist)
```

TPS: How is the sorting happening?

```
>>> d
{'a': [1, 2, 3], 'b': [4, 7], 'c': [1, 1, 5, 8]}
>>> sorted(d.items())
[('a', [1, 2, 3]), ('b', [4, 7]), ('c', [1, 1, 5, 8])]
>>> sorted(d.items(), key=lambda x: x[1])
[('c', [1, 1, 5, 8]), ('a', [1, 2, 3]), ('b', [4, 7])]
>>> sorted(d.items(), key=lambda x: x[1][-1])
[('a', [1, 2, 3]), ('b', [4, 7]), ('c', [1, 1, 5, 8])]
```

Activity 1: https://bit.ly/101f21-11-09-1

How to do some “fancy” sorting

- lambda PARAMETER : EXPRESSION

- Given the following data: (first name, last name, age)
  - `[(‘Percival’, ‘Avram’, 51), (‘Melete’, ‘Sandip’, 24), …]`

- TPS: What is the lambda key to sort the following?
  - Sort by last name, break ties with first name
  - Sort by last name, break ties with age
  - Alphabetical by last name, then first name, then reverse age order

Creating Tuples with lambda

- Sort by last name, break ties with first name
  - key = lambda x: (x[1], x[0])
- Sort by last name, break ties with age
  - key = lambda x: (x[1], x[2])
- Alphabetical by last name, then first name, then reverse age order
  - key = lambda x: (x[1], x[0], -x[2])

- What if wanted something really different (but not use lambda)?
  - Sort alphabetical by last name, break ties by reverse alphabetical using first name

Leveraging the Algorithm

- If can't sort by creating a tuple with lambda, use:
  - Pattern: Multiple-pass *stable* sort – first sort with last tie breaker, then next to last tie breaker, etc. until at main criteria

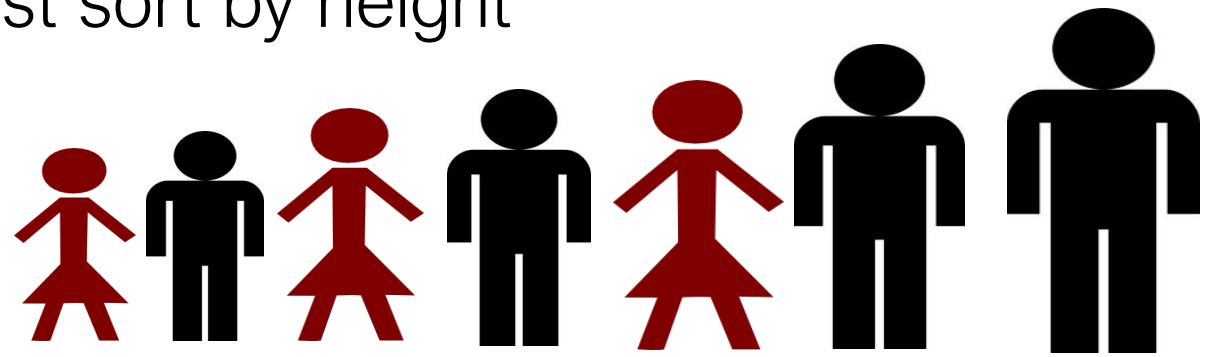
- Sort by index 0, break tie in reverse order with index 1

```
[('b', 'z'), ('c', 'x'), ('b', 'x'), ('a', 'z')]
[('b', 'z'), ('a', 'z'), ('c', 'x'), ('b', 'x')]
[('a', 'z'), ('b', 'z'), ('b', 'x'), ('c', 'x')]
```

- *Stable* sort respects original order of "equal" keys

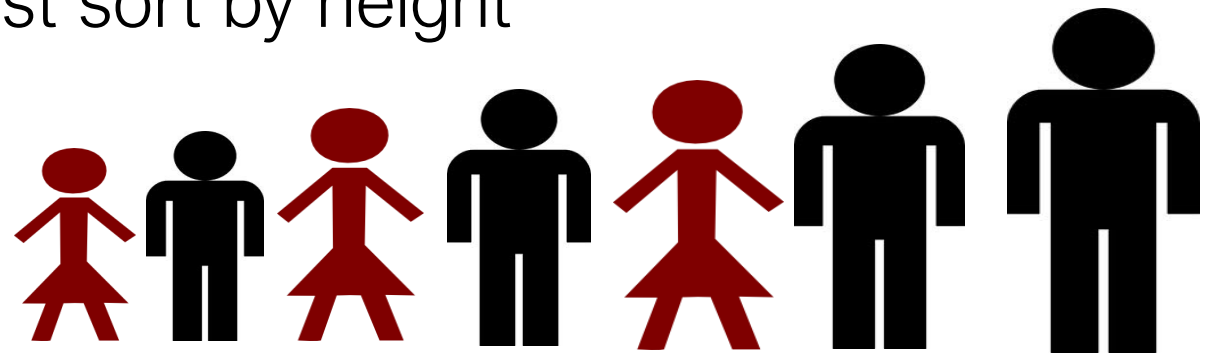
Stable sorting: respect "equal" items

- **Women before men, each group height-sorted**
  - First sort by height

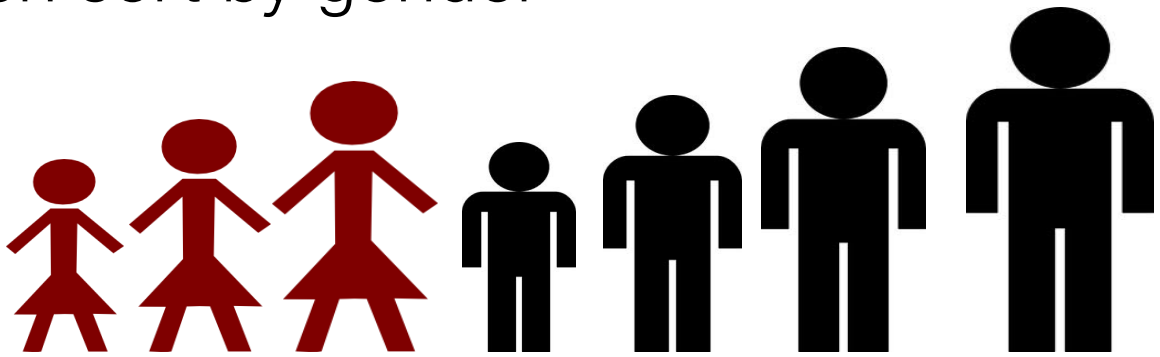


Stable sorting: respect "equal" items

- **Women before men, each group height-sorted**
  - First sort by height



  - Then sort by gender



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)]
```

Activity 2: https://bit.ly/101f21-11-9-2

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

Reminders

- Work smarter, not harder
- Design first
- Try to identify where you are stuck
  - Identify resources to help solve problem
- Leverage your design and PythonTutor to understand program flow of control
  - http://pythontutor.com