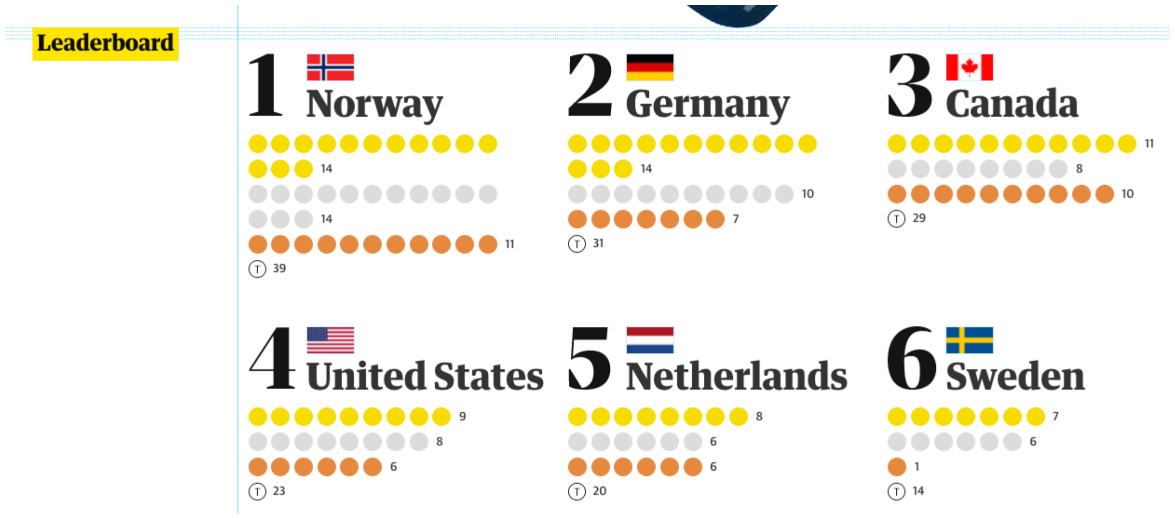
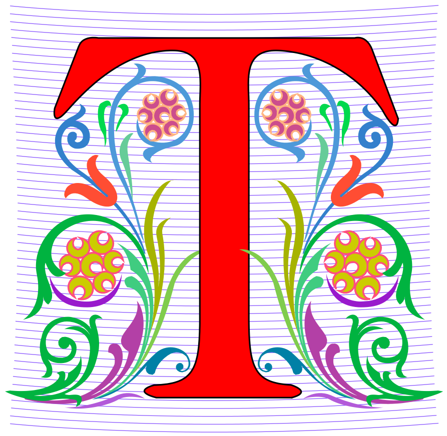
Compsci 101
Stable Sorting

Susan Rodger

March 31, 2022



T is for …



- **Type**
  - From int to float to string to list to …
- **Text**
  - From .txt to editors to …
- **Turing Award**
  - Nobel, Fields, Turing
  - Turing Duke Alums:
    - Ed Clarke (MS)
    - John Cocke (BS, PhD)
    - Fred Brooks (BS)

danah boyd

Dr. danah boyd is a Principal Researcher at Microsoft Research, … and a Visiting Professor at New York University. Her research is focused on addressing social and cultural inequities by understanding the relationship between technology and society.

"If I have learned one thing from my research, it's this: social media services like Facebook and Twitter are providing teens with new opportunities to participate in public life, and this, more than anything else, is what concerns many anxious adults."

Announcements

- APT-6 due Thursday, March 31, TODAY
- Assignment 5 Clever GuessWord due Tues. April 5
- APT-7 out today! Due April 7

- Lab 10 Friday
  - There is a prelab!

- APT Quiz 2 in one week, April 7-10
  - Covers topics thru APT 6
  - Old exams on old tests page
- Exam 4 is April 12, more on that on Tuesday

APT Quiz 2 April 7-10

- **Opens 4/7 11:30am**
- **Closes at 11pm 4/10 – must finish all by this time**
- **There are two parts based on APTs 1-7**
  - Each part has two APT problems
  - Each part is 1.5 hours – more if you get accommodations
  - Each part starts in Sakai under tests and quizzes
  - Sakai is a starting point with countdown timer that sends you to a new apt page just for each part
  - Could do each part on different day or same days
- **Will put up problems today from an old APT Quiz so you can practice (not for credit) – on APT Page**

APT Quiz 2

- Is your own work!
  - No collaboration with others!
  - Use your notes, lecture notes, your code, textbook
  - DO NOT search for answers!
  - Do not talk to others about the quiz until grades are posted
- Post private questions on Ed Discussion
  - We are not on between 10pm and 8am!
  - We are not on all the time
  - Will try to answer questions between 8am – 10pm
- See 101 APT page for tips on debugging APTs

PFTD

- Stable Sorting

- Some APTs

How is the sorting happening?

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

```
>>> sorted(d.items(), key=lambda x: x[1])
```

```
>>> sorted(d.items(), key=lambda x: x[1][-1])
```

How to do some “fancy” sorting

- lambda PARAMETER : EXPRESSION

- Given data: list of tuples: (first name, last name, age)

```
 [(‘Percival’, ‘Avram’,  51),
  (‘Melete’,   ‘Sandip’, 24), …]
```

- Think: What is the lambda key to sort the following?

```
sorted(data, key=lambda z : (z[0],z[1],z[2]))
```

  - 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
- Sort by last name, break ties with age
- Alphabetical by last name, then first name, then reverse age order

Leveraging the Algorithm

- 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

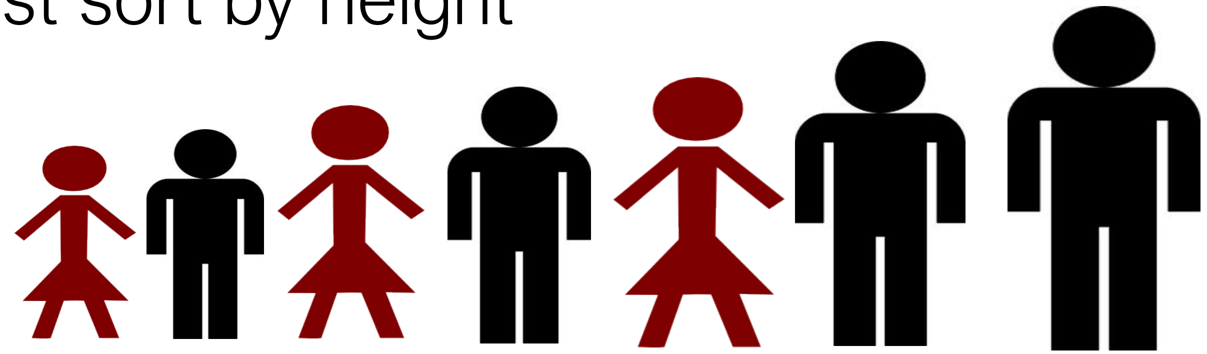
- Sort by index 0, break tie in reverse order with index 1

```
[(‘b’, ‘z’), (‘c’, ‘x’), (‘b’, ‘x’), (‘a’, ‘z’)]
```

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

Stable sorting: respect "equal" items

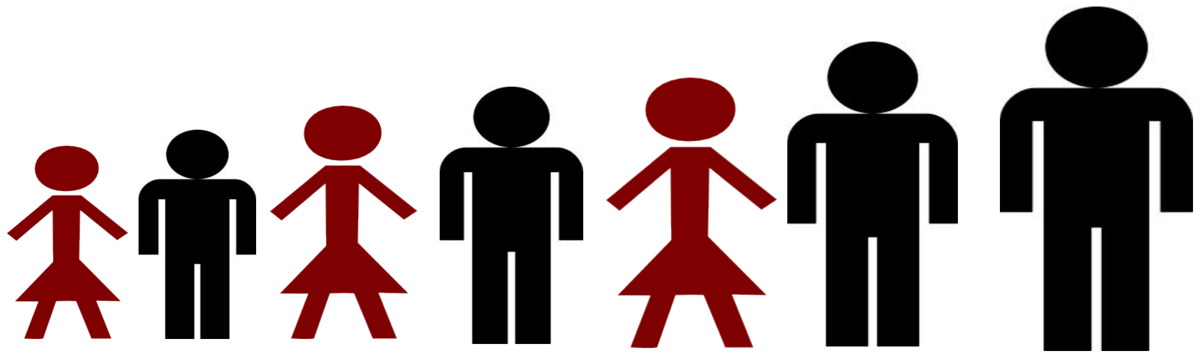
- **Women before men, each group height-sorted**
  - First sort by height



Understanding Multiple-Pass Sorting

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


> a1 = sorted(a0, key = lambda x: x[2])
> a1


> a2 = sorted(a1, key = lambda x: x[1])
> a2
```

WOTO-1 Multipass Sorting
http://bit.ly/101s22-0331-1



Unpack from Inside out

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

lst = [[4, 6, 7], [5, 2], [3, 9], [6, 2, 9]
x = sorted(lst, key=sum)

y = sorted(x, key = min)

z = sorted(y, key=max)

When and What's in CompSci 101

- Problem to solve
  - Use 7 steps
  - Step 5: How do you translate algorithm to code?
    - What do you use to solve it?
    - When do you use it?

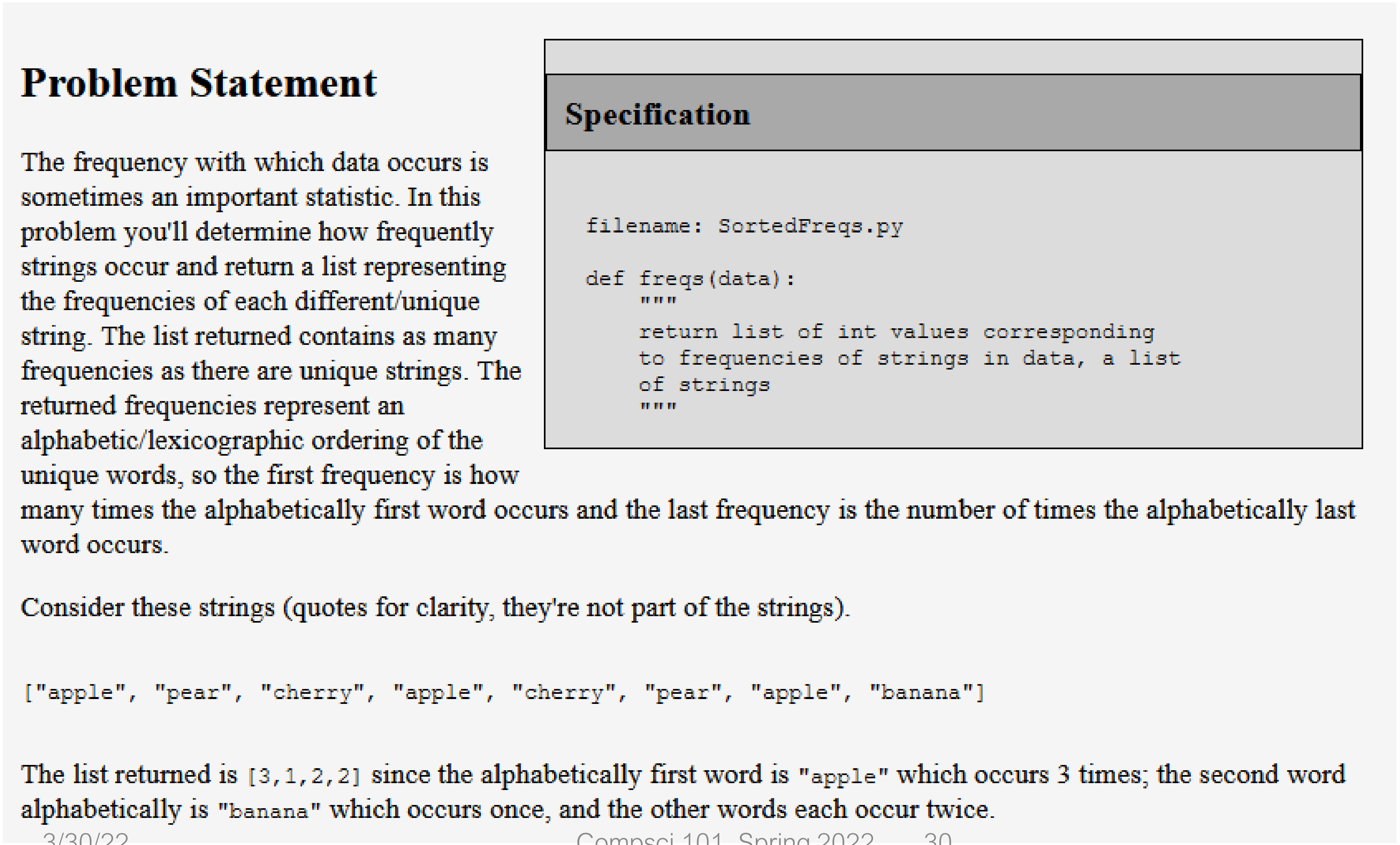
What are the “what’s”?

- Data Structures: list, set, dictionary, tuple
- Loops : from for to while
- Other:
  - List comprehensions
  - Parallel lists
  - Lambda
  - If…if…if
  - If…elif…else

Quick When's and What's for 101

- **Whichever makes more sense to you:**
  - Parallel lists vs dictionaries
  - If…if…if vs if…elif…else
  - List comprehension vs for loop
  - Tuples vs Lists
    - If you want to prevent mutation -> tuples
  - Need single line function
    - Lambda vs create normal helper function

APT – Sorted Freqs



APT SortedFreqs

Problem Statement

The frequency with which data occurs is sometimes an important statistic. In this problem you'll determine how frequently strings occur and return a list representing the frequencies of each different/unique string. The list returned contains as many frequencies as there are unique strings. The returned frequencies represent an alphabetic/lexicographic ordering of the unique words, so the first frequency is how many times the alphabetically first word occurs and the last frequency is the number of times the alphabetically last word occurs.

Consider these strings (quotes for clarity, they're not part of the strings).

```
["apple", "pear", "cherry", "apple", "cherry", "pear", "apple", "banana"]
```

The list returned is `[3,1,2,2]` since the alphabetically first word is `"apple"` which occurs 3 times; the second word alphabetically is `"banana"` which occurs once, and the other words each occur twice.

Specification

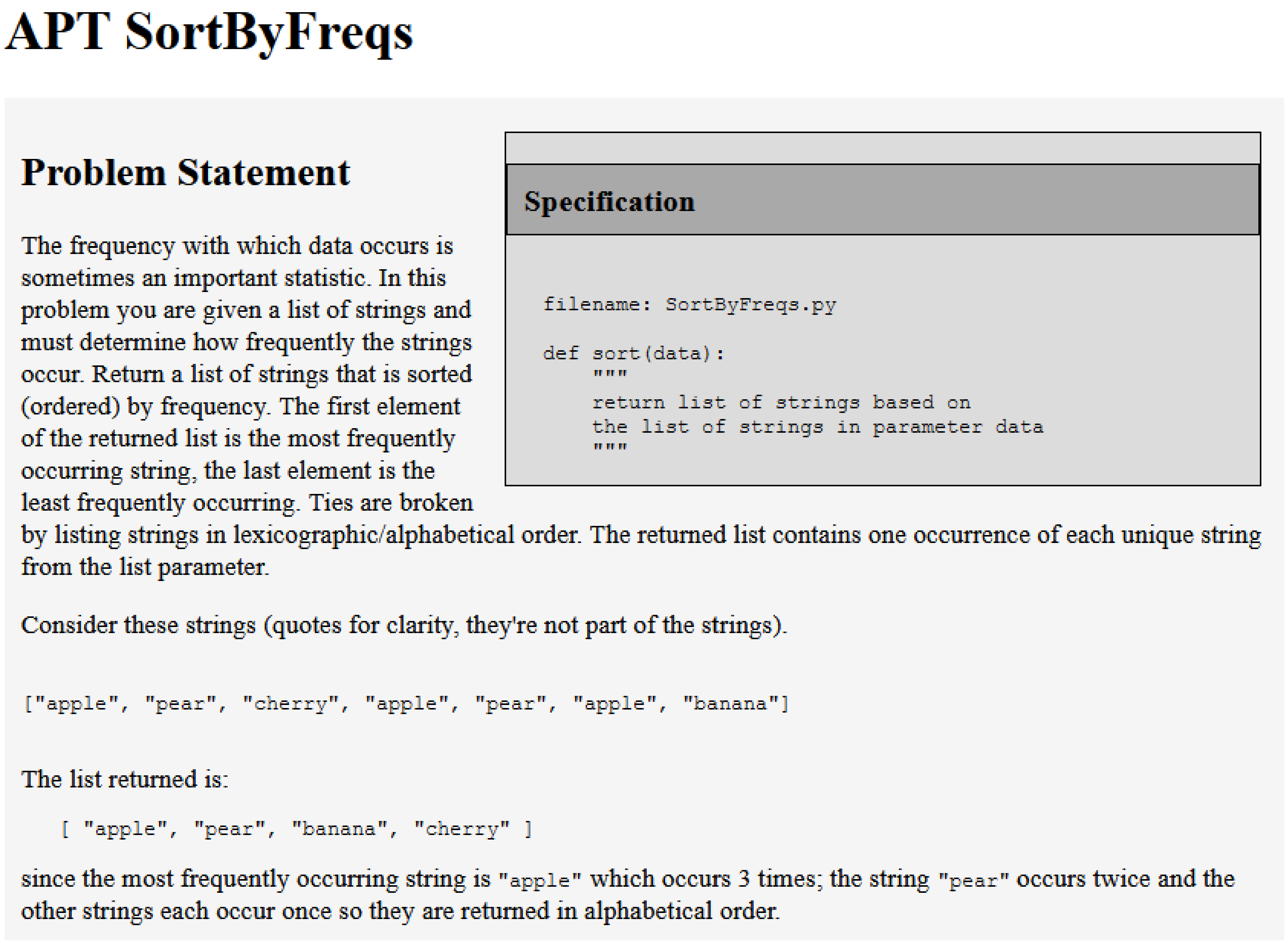
```
filename: SortedFreqs.py

def freqs(data):
    """
    return list of int values corresponding
    to frequencies of strings in data, a list
    of strings
    """
```

What's the best way to …

- SortedFreqs
  - https://www2.cs.duke.edu/csed/pythonapt/sortedfreqs.html

- Count how many times each string occurs
  - Create d = {}, iterate over list updating values
- OR
  - Use data.count(w) for each w

APT: SortByFreqs



APT SortByFreqs

Problem Statement

The frequency with which data occurs is sometimes an important statistic. In this problem you are given a list of strings and must determine how frequently the strings occur. Return a list of strings that is sorted (ordered) by frequency. The first element of the returned list is the most frequently occurring string, the last element is the least frequently occurring. Ties are broken by listing strings in lexicographic/alphabetical order. The returned list contains one occurrence of each unique string from the list parameter.

Consider these strings (quotes for clarity, they're not part of the strings).

```
["apple", "pear", "cherry", "apple", "pear", "apple", "banana"]
```

The list returned is:

```
   [ "apple", "pear", "banana", "cherry" ]
```

since the most frequently occurring string is `"apple"` which occurs 3 times; the string `"pear"` occurs twice and the other strings each occur once so they are returned in alphabetical order.

Specification

```
filename: SortByFreqs.py

def sort(data):
    """
    return list of strings based on
    the list of strings in parameter data
    """
```

Wait, wait, but what's …

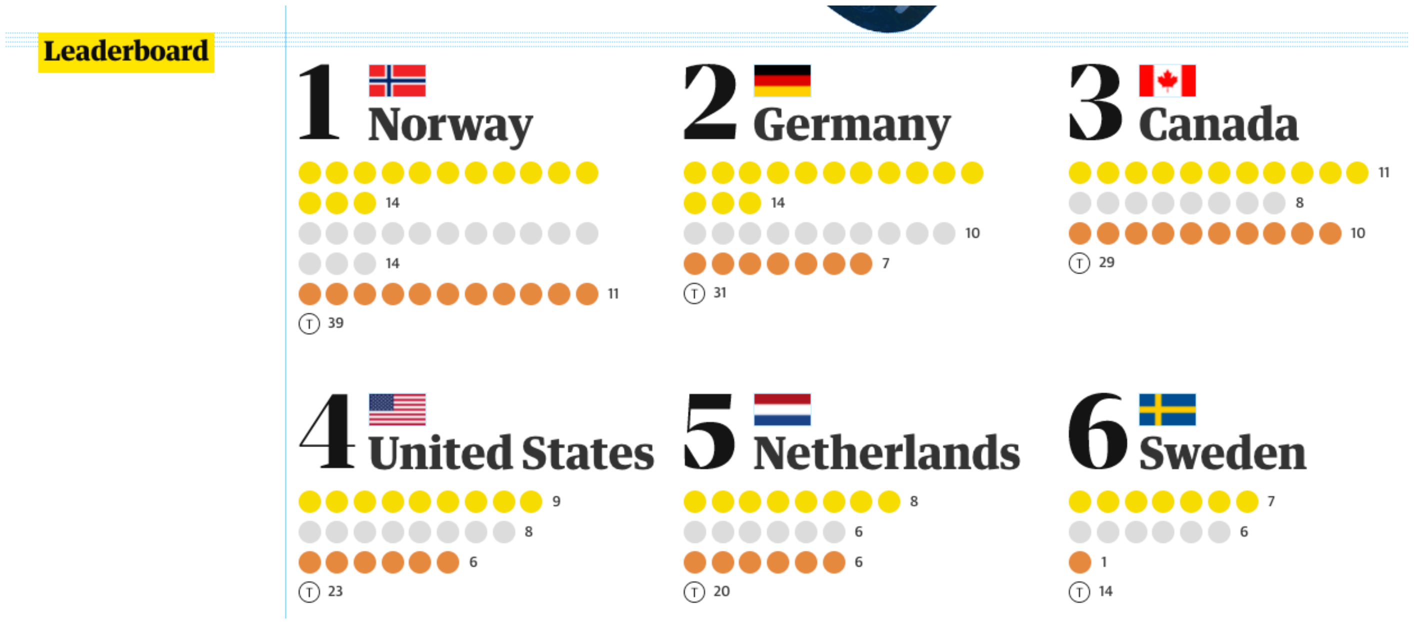
- SortByFreqs
  - https://www2.cs.duke.edu/csed/pythonapt/sortbyfreqs.html

- Sort by # occurrences high to low
  - Tuples with count/lambda and reverse=True?
  - Break ties in alphabetical order: two passes

WOTO-2
http://bit.ly/101s22-0331-2

APT: MedalTable

- http://bit.ly/apt-medal-table


Leaderboard

1 Norway
14
14
11
T 39

2 Germany
14
10
7
T 31

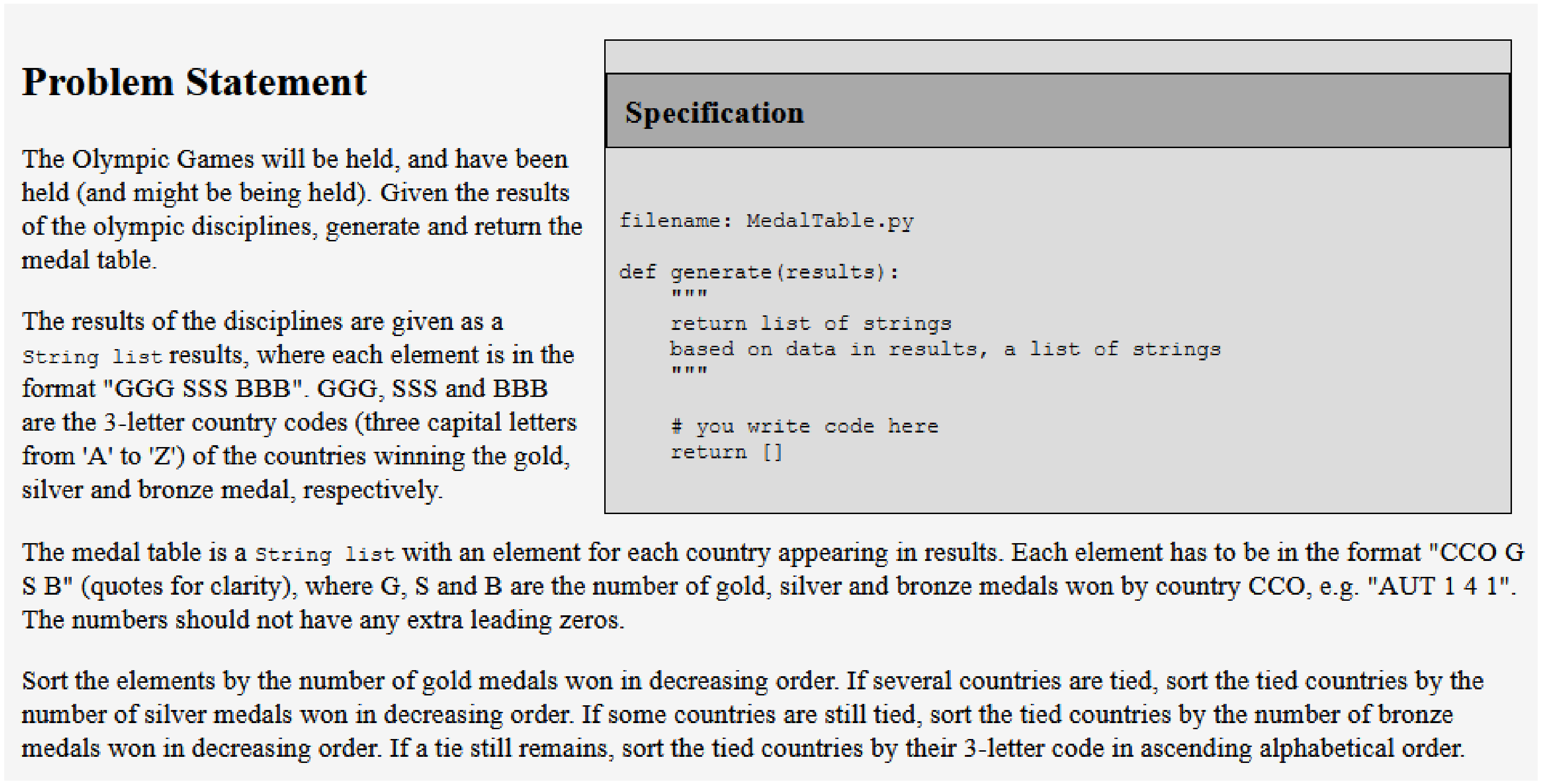
3 Canada
11
8
10
T 29

4 United States
9
8
6
T 23

5 Netherlands
8
6
6
T 20

6 Sweden
7
6
1
T 14

APT: MedalTable



Problem Statement

The Olympic Games will be held, and have been held (and might be being held). Given the results of the olympic disciplines, generate and return the medal table.

The results of the disciplines are given as a `String list` results, where each element is in the format "GGG SSS BBB". GGG, SSS and BBB are the 3-letter country codes (three capital letters from 'A' to 'Z') of the countries winning the gold, silver and bronze medal, respectively.

Specification

```
filename: MedalTable.py

def generate(results):
    """
    return list of strings
    based on data in results, a list of strings
    """

    # you write code here
    return []
```

The medal table is a `String list` with an element for each country appearing in results. Each element has to be in the format "CCO G S B" (quotes for clarity), where G, S and B are the number of gold, silver and bronze medals won by country CCO, e.g. "AUT 1 4 1". The numbers should not have any extra leading zeros.

Sort the elements by the number of gold medals won in decreasing order. If several countries are tied, sort the tied countries by the number of silver medals won in decreasing order. If some countries are still tied, sort the tied countries by the number of bronze medals won in decreasing order. If a tie still remains, sort the tied countries by their 3-letter code in ascending alphabetical order.

1.
```
["ITA JPN AUS", "KOR TPE UKR", "KOR KOR GBR", "KOR CHN TPE"]

Returns:
[ "KOR 3 1 0",  "ITA 1 0 0",  "TPE 0 1 1",  "CHN 0 1 0",  "JPN 0 1 0",
  "AUS 0 0 1",  "GBR 0 0 1",  "UKR 0 0 1"
]
```

Tracking the Data

- **What do we need to obtain for each country?**
  - What's the data, how do we store it?
  - What's the data, how do we calculate it?

- **Method and code to transform input**
  - What will we store, how do we initialize/update
  - Verifying we've done this properly

Use a dictionary?

- 3 dictionaries
  - Country to Gold count
  - Country to Silver count
  - Country to Bronze count

Use a dictionary?

- 3 dictionaries
  - Country to Gold count
  - Country to Silver count
  - Country to Bronze count
- 1 dictionary
  - Map country to [gld cnt, sil cnt, bro cnt]

Example

- How would we create a dictionary for:

[“KOR TPE UKR”, “KOR KOR TPE”, “KOR JPN JPN”]

Example: dictionary d

- Process first string: "KOR TPE UKR“

- **Process second string: "KOR KOR TPE”**

Example: dictionary d (2)

- What we have so far:

- Process third string: **"KOR JPN JPN“**

Sorting the Data

- **Write a helper function to build the dictionary**
  - d = builddict(results)
  - Where results is string of countries for each event
- **Use dictionary to get list of tuples**

```
[('JPN', [0, 1, 1]), ('KOR', [3, 1, 0]),
('TPE', [0, 1, 1]), ('UKR', [0, 0, 1])]
```

- **Then do passes to sort the data**
  - Will discuss sorting the data in lab

WOTO-3 Building Dictionary
http://bit.ly/101s22-0331-3