Compsci 101
Sorting, CSV
Live Lecture

	A	B	C
1	**Rank**	**Song**	**Artist**
2	1	Like a Rolling Stone	Bob Dylan
3	2	Satisfaction	The Rolling Stones
4	3	Imagine	John Lennon
5	4	What's Going On	Marvin Gaye
6	5	Respect	Aretha Franklin
7	6	Good Vibrations	The Beach Boys
8	7	Johnny B. Goode	Chuck Berry
9	8	Hey Jude	The Beatles
10	9	Smells Like Teen Spirit	Nirvana
11	10	What'd I Say	Ray Charles

Susan Rodger
Nicki Washington
March 25, 2021

Announcements

- Nothing due today!
- Assignment 4 due Tue, March 30
- APT-6 due Thur, Apr 1
- Assignment 5 out today, due Tues, Apr 6
  - Discuss on March 30th
  - Read it before then, also a Sakai quiz out today to go with it

- Lab 8 on Friday
  - There is no prelab!

PFTD

- Sorting
  - Sorting using standard Python APIs
- CSV Library
  - How to read data using standard Python APIs
- Lambda
  - Language construct to make sorting simpler (next week)

WOTO-1 Popular Music
http://bit.ly/101s21-0325-1

- Make a copy of this spreadsheet:
  - http://bit.ly/101s21-0325-data

WOTO-1 Popular Music
http://bit.ly/101s21-0325-1

- Make a copy of this spreadsheet:
  - http://bit.ly/101s21-0325-data
- Who are top two artists? Most Songs
- How did you do it?

	A	B	C
1	**Rank**	**Song**	**Artist**
2	1	Like a Rolling Stone	Bob Dylan
3	2	Satisfaction	The Rolling Stones
4	3	Imagine	John Lennon
5	4	What's Going On	Marvin Gaye
6	5	Respect	Aretha Franklin
7	6	Good Vibrations	The Beach Boys
8	7	Johnny B. Goode	Chuck Berry
9	8	Hey Jude	The Beatles
10	9	Smells Like Teen Spirit	Nirvana
11	10	What'd I Say	Ray Charles

Solve a Larger Problem

- **Suppose I were to give you the top 1000 artists**
  - Top 1,000 songs, find top 10 artists
  - How many songs per artist?

Scale

- As the size of the problem grows we want …
  - The algorithm to still work and be fast!
  - What to do?

- Search example
  - Google search results work
  - SoundHound/Shazam results work
  - ContentID on YouTube results work

Python to the Rescue

- Using **`.sort(…)`**, **`sorted(…)`**, and **`lambda`**
- Using CSV library and its API
  - CSV – Comma Separated Values
- Why use the CSV library?
  - How to handle the song “Hello, I Love You”?
  - Row 166 in spreadsheet

Hits by Artists: SongReader.py

- What is returned by this function?
  - details of csv: **`next`** and no **`split`** and …

```
 9  def countByArtist(name):
10      csvf = open(name, 'r', encoding='utf-8')
11      freader = csv.reader(csvf)
12      header = next(freader)
13      print("header row labels", header)
14      data = {}
15      for row in freader:
16          artist = row[2]
17          if artist not in data:
18              data[artist] = 0
19          data[artist] += 1
20
21      csvf.close()
22      return data
```

What is new?
What does it do?

WOTO-2 countByArtist
http://bit.ly/101s21-0325-2

Sorting to Print/Visualize

- Dictionary is ('Beatles', 51) tuples
  - But tuples not in order, so we must …

```
24 ▶ if __name__ == '__main__':
25       counts = countByArtist("data/top1000.csv")
26
27       print('\nFirst 5 artists:')
28       for artist in sorted(counts.items())[:5]:
29           print(artist)
30
31       print('\nTop 5 artists:')
32       sortbycount = sorted([(a[1], a[0]) for a in counts.items()])
33       sortedArtists = [(a[1], a[0]) for a in sortbycount]
34       for artist in sortedArtists[-5:]:
35           print(artist)
```

What is going on here?

Why more complicated than lines 28 & 29?

WOTO-3 Calling countByArtist
http://bit.ly/101s21-0325-3

Two APIs: CSV and Sorting

- CSV Library to read and process data
  - Comma-separated, but can by ":" separated, or any character as we'll see later

- Similar to reading a file – returned by open
  - Iterable is returned by **`csv.reader`**
  - The **`next`** function advances iterable
  - Don't call **`split`**, we can access by index
    - Also by header-row label with **`csv.dictreader`**

Sorting API and Sorting Concepts

- What is **`counts.items()`** – how is it sorted?

```
27      print('\nFirst 5 artists:')
28      for artist in sorted(counts.items())[:5]:
29          print(artist)
```

How does Python evaluate slice?

- What does sorted return?
  - A list, you can slice a list, look for clues!
  - What can be sorted? A sequence
  - **`sorted(counts.items())`**

Sorting by Number of Songs

- Sort by first value vs sort by second value
  - Need to put sequence back to original format

```
27      print('\nFirst 5 artists:')
28      for artist in sorted(counts.items())[:5]:
29          print(artist)
30
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)
```

If we comment out 33, what's printed? Why?

Python Sorting API

- We'll use both **`sorted()`** and **`.sort()`** API
  - How to call, what options are
  - How to sort on several criteria

- Creating a new list, modifying existing list
  - **`sorted(..)`** creates list from .. Iterable
  - **`x.sort()`** modifies the list x, no return value!

API to change sorting

- In SongReader.py we changed order of tuples to change sorting order
  - Then we sliced the end to get "top" songs

- Can supply a function to compare elements
  - Function return value used to sort, key=function
  - Change order: reverse=True

Sorting Examples

- Use key=function argument and reverse=True
  - What if we want to write our own function?

```
In[2]: a = ["red", "orange", "green", "blue", "indigo", "violet"]
In[3]: sorted(a)
Out[3]: ['blue', 'green', 'indigo', 'orange', 'red', 'violet']
In[4]: sorted(a,key=len)
Out[4]: ['red', 'blue', 'green', 'orange', 'indigo', 'violet']
In[5]: sorted(a,key=len,reverse=True)
Out[5]: ['orange', 'indigo', 'violet', 'green', 'blue', 'red']
```

WOTO-4 Sorting
http://bit.ly/101s21-0325-4

Assignment 5 – Clever Hangman

- Must finish Hangman assignment first!
- This is a copy and modify

- Make it hard to win
- Keep changing the word
  - More than that
- We will discuss next time