CompSci 101
Fall 2021

Lecture 16

Reminders

- Assignments
  - Assign 5 live
- APT Quiz 2
  - 11/12-11/15
- Final Exam
  - APT Quiz style

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

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!*

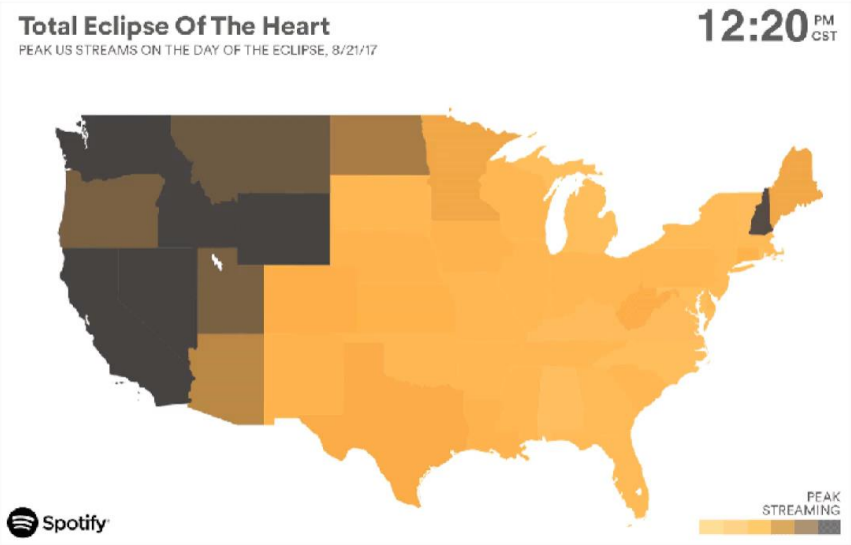
People to Know: Taraneh BigBow

- Software Engineer (backend development)
  - TEKSystems at Apple
  - Jasco Products, Microsoft
- BigBow Technologies

- Oklahoma
  - Kiowa Tribe

TPS: Why Sort Data?

- Help understand data
  - http://bit.ly/spotify-eclipse-cnet
    - https://www.youtube.com/watch?v=IcOxhH8N3Bo



Why Sort Data?

- **Every field needs to visualize and understand data**
  - Sorting helps with this from movies to policy to sports to location of infections to

https://www.esri.com/arcgis-blog/products/apps/local-government/how-your-gis-department-can-respond-to-covid-19/

How your GIS department can respond to COVID-19

Local Government

March 09, 2020

Mike Schoelen

A staggering wealth of geospatial information has emerged regarding the COVID-19 outbreak. Dashboards, near real-time services, and GitHub repositories have built the foundation for an extraordinarily transparent response effort.

How To Sort: Algorithms

- Does scale matter? It depends!

- You're playing Spades, Hearts, Bridge, Go-Fish
  - How you sort doesn't really matter, but whether you sort makes play more efficient? Better?

- Many ways to sort
  - Bubble, Insertion, Selection
  - Quick, Merge, Tim, Bogo

Activity 1: Popular Music https://bit.ly/101f21-11-2-1

- 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

- http://bit.ly/dukecs101sp20-copytop1000
  - 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”?

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?

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?

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

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']
```

  - What do you notice about 4 and 5?

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

The power of lambda

- We want to create a function "on-the-fly"
  - aka anonymous (“throw away”) function
- *lambda parameter(s): expression*

```
In[7]: a
Out[7]: ['red', 'orange', 'green', 'blue', 'indigo', 'violet']
In[8]: sorted(a,key=lambda x : x.count("e"))
Out[8]: ['indigo', 'red', 'orange', 'blue', 'violet', 'green']
```

- Why 'indigo' first and 'green' last?
  - What about order of ties? Next class! Stable

Anonymous Functions

- Useful when want “throw-away” function
  - Our case mainly sort

- Syntax: **`lambda PARAMETERS: EXPRESSION`**
  - `PARAMETERS` – 0 or more comma separated
  - `EXPRESSION` – evaluates to something

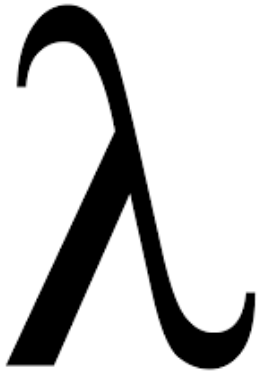
key=FUNCTION

- **`from operator import itemgetter`**
  - Can use
- **`itemgetter(N)`** same as **`lambda x: x[N]`**

- Past used itemgetter
- Now use lambda (more versatile)

Why is lambda used?

- It doesn't matter at all
  - https://en.wikipedia.org/wiki/Alonzo_Church
    - ola aside: church->martin davis->don loveland->ola
  - Lisp and Scheme have lambda expressions
  - Guido, former BDFL, learned to live with lambda



What is a lambda expression?

- It's a function object, treat like expression/variable
  - Like list comprehensions, access variables

```
>>> inc = lambda x : x + 1
>>> p = [1, 3, 5, 7]
>>> [inc(num) for num in p]
[2, 4, 6, 8]
```

Syntactic sugar (makes the medicine go down)

- Syntactic sugar for a normal function definition

```
def f(x):
    return x[1]
sorted(lst, key=f)
```

```
f = lambda x : x[1]
sorted(lst, key=f)
```

```
sorted(lst, key=lambda x : x[1])
```

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

Parameter name does not matter

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']
```

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