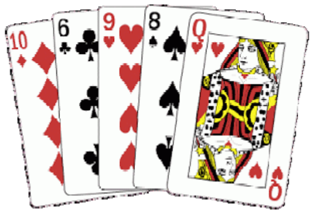
Plan for WBTB

- APT Quiz 3 – due tonight
- Solving problems in the wild
  - How can you change how things are sorted
    - Other than ordering and re-ordering tuples
    - How do Python .sort and sorted() stack up?
  - How do you access directories?
    - And all the files in a directory, and the …
  - How do you access web-based files?
    - How to parse <a href> HTML? Other formats?

Playing go-fish, spades, or …

- Finding right card?
  - What helps?
  - Issues here?
- Describe algorithm:
  - First do this
  - Then do this
  - Substeps ok
  - When are you done?



Problem Solving with Algorithms

- Top 100 songs of all time, top 2 artists?
  - Most songs in top 100
  - Wrong answers heavily penalized
  - You did this in lab, you could do this with a spreadsheet
- What about top 1,000 songs, top 10 artists?
  - How is this problem the same?
  - How is this problem different

Scale

- As the size of the problem grows …
  - The algorithm continues to work
  - A new algorithm is needed
  - New engineering for old algorithm
- Search
  - Making Google search results work
  - Making SoundHound search results work
  - Making Content ID work on YouTube

Python to the rescue? Top1000.py

```
import csv, operator

f = open('top1000.csv','rbU')
data = {}
for d in csv.reader(f,delimiter=',',quotechar='"'):
    artist = d[2]
    song = d[1]
    if not artist in data:
        data[artist] = 0
    data[artist] += 1

itemlist = data.items()
dds = sorted(itemlist,key=operator.itemgetter(1),reverse=True)
print dds[:30]
```

Understanding sorting API

- How API works for `sorted()` or `.sort()`
  - Alternative to changing order in tuples and then changing back

```
x = sorted([(t[1],t[0]) for t in dict.items()])
x = [(t[1],t[0]) for t in x]

x = sorted(dict.items(),key=operator.itemgetter(1))
```

- Sorted argument is key to be sorted on, specify which element of tuple. Must import library operator for this

Sorting from an API/Client perspective

- API is Application Programming Interface, what is this for sorted(..) and .sort() in Python?
  - Sorting algorithm is efficient, stable: part of API?
  - `sorted` returns a list, doesn't change argument
  - `sorted(list,reverse=True)`, part of API
  - `foo.sort()` modifies foo, same algorithm, API
- How can you change how sorting works?
  - Change order in tuples being sorted,
    - `[(t[1],t[0]) for t in …]`
  - Alternatively: `key=operator.itemgetter(1)`

Beyond the API, how do you sort?

- Beyond the API, how do you sort in practice?
  - Leveraging the stable part of API specification?
  - If you want to sort by number first, largest first, breaking ties alphabetically, how can you do that?
- Idiom:
  - Sort by two criteria: use a two-pass sort, first is secondary criteria (e.g., break ties)

```
[("ant",5),("bat", 4),("cat",5),("dog",4)]
[("ant",5),("cat", 5),("bat",4),("dog",4)]
```

Two-pass (or more) sorting

- Because sort is stable sort first on tie-breaker, then that order is fixed since stable

```
a0 = sorted(data,key=operator.itemgetter(0))
a1 = sorted(a0,key=operator.itemgetter(2))
a2 = sorted(a1,key=operator.itemgetter(1))
data
[('f', 2, 0), ('c', 2, 5), ('b', 3, 0),
 ('e', 1, 4), ('a', 2, 0), ('d', 2, 4)]
a0
[('a', 2, 0), ('b', 3, 0), ('c', 2, 5),
 ('d', 2, 4), ('e', 1, 4), ('f', 2, 0)]
```

Two-pass (or more) sorting

```
a0 = sorted(data,key=operator.itemgetter(0))
a1 = sorted(a0,key=operator.itemgetter(2))
a2 = sorted(a1,key=operator.itemgetter(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)]
```

Answer Questions

http://bit.ly/101fall15-nov17-1

Timingsorts.py, what sort to call?

- Simple to understand, hard to do fast and at-scale
  - Scaling is what makes computer science …
    - Efficient algorithms don't matter on lists of 100 or 1000
  - Named algorithms in 201 and other courses
    - bubble sort, selection sort, merge, quick, …
    - See next slide and TimingSorts.py

- Basics of algorithm analysis: theory and practice
  - We can look at empirical results, would also like to be able to look at code and analyze mathemetically! How does algorithm scale?

New sorting algorithms happen …

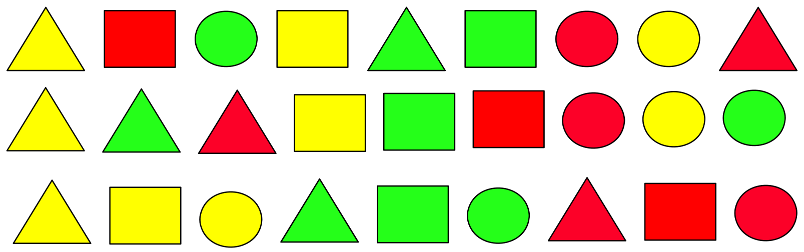
- timsort is standard on…
  - Python as of version 2.3, Android, Java 7
  - According to http://en.wikipedia.org/wiki/Timsort
    - Adaptive, stable, natural mergesort with supernatural performance
- What is mergesort? Fast and Stable
  - What does this mean?
  - Which is most important?
  - Nothing is faster, what does that mean?
  - Quicksort is faster, what does that mean?

TimingSorts.py

size	create	bubble	select	timsort
1000	0.026	0.127	0.081	0.002
2000	0.045	0.537	0.273	0.001
3000	0.058	1.126	0.646	0.002
4000	0.082	2.174	1.208	0.003
5000	0.101	3.521	1.862	0.003
6000	0.118	4.617	3.005	0.004
7000	0.168	7.504	4.237	0.005
8000	0.156	9.074	6.152	0.007
9000	0.184	11.611	8.089	0.007
10000	0.212	14.502	9.384	0.008

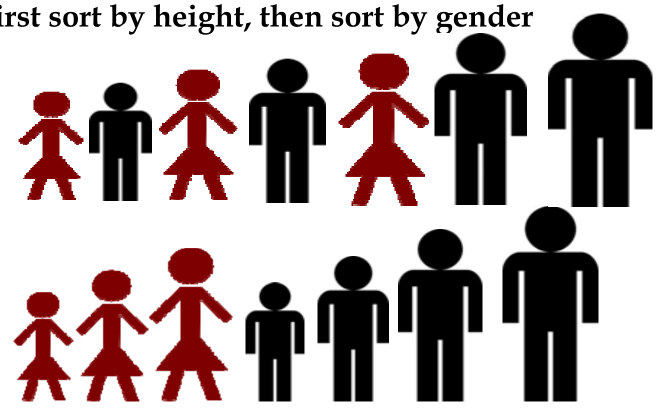
Stable, Stability

- What does the search query 'stable sort' show us?
  - Image search explained
  - First shape, then color: for equal colors?



Stable sorting: respect re-order

- Women before men …
  - First sort by height, then sort by gender



How to import: in general and sorting

- We can write: import operator
  - Then use key=operator.itemgetter(…)
- We can write: from operator import itemgetter
  - Then use key=itemgetter(…)
- From math import pow, From cannon import pow
  - Oops, better not to do that, use dot-qualified names like math.sqrt and operator.itemgetter

TimingSorts.py Questions

http://bit.ly/101fall15-nov17-2