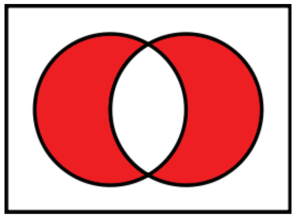
Compsci 101
Sets, Simple Sorting



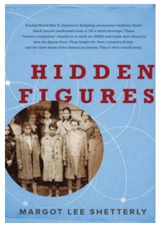
Susan Rodger
Feb 24, 2022

M is for …

- **Machine Learning**
  - Math, Stats, Compsci: learning at scale
- **Microsoft, Mozilla, Macintosh**
  - Software that changed the world?
- **Memory**
  - Storage space in the computer
  - From 64 Kilobytes to 16 Gigobytes!
- **Mouse, Mouse pad**
  - Easier to navigate

Margot Shetterly

- Writer, Author of Hidden Figures
- Black Women NASA Scientists
- Gave a talk at Duke in 2016



Katherine Johnson	Mary Jackson	Dorothy Vaughn	Christine Darden

Announcements

- APT-4 is out and due Thursday March 3
  - Already looked at one in Lab, one in Lecture!
- Assignment 3 due Tuesday, March 1

- Lab 7 Friday, there is a prelab available now!
- No lab on Friday, March 4

- Take APT Quiz 1 – Feb. 24-27
  - Two parts – each part 1.5 hours, 2 APTs
  - Start on Sakai under quizzes

PFTD

- Simple Sorting
- Sets and APTs

Let’s sort lists with sorted() function

- Want list elements in sorted order
  - Example: have list [17 , 7, 13, 3]
  - Want list [3, 7, 13, 17], in order
- Built-in function: sorted(*sequence*)
  - Returns new list of sequence in sorted order
  - Sequence could be list, tuple, string

Example

```
lst = [6, 2, 9, 4, 3]          lst  is [6, 2, 9, 4, 3]
lsta = sorted(lst)
b = [‘ko’, ‘et’, ‘at’, ‘if’]
c = sorted(b)
b.remove(‘et’)
b.append(6)
b.insert(1,5)
c = sorted(b)
```

Example

```
lst = (7, 4, 1, 8, 3, 2)       lst  is    (7, 4, 1, 8, 3, 2)
lsta = sorted(lst)
b = (‘ko’, ‘et’, ‘at’, ‘if’)
c = sorted(b)
d = “word”
e = sorted(d)
f = ‘go far’
g = sorted(f)
f = ‘go far’
h = sorted(f.split())
```

Now, sort lists with .sort() list method

- **Want to "change" list elements to sorted order**
  - lst is [17 , 7, 13, 3]
  - lst.sort()
  - Now **same** list lst is [3, 7, 13, 17], in order
- **List method: *list*.sort()**
  - List is **modified, now in sorted order**
  - There is NO return value
  - Only works with lists, can't modify strings, tuples

Compare sorted() with .sort()

```
lsta = [6, 2, 9, 4, 3]
lstb = sorted(lsta)

lsta.sort()
a = [7, 2, 9, 1]
b = a.sort()

c = (5, 6, 2, 1)
c.sort()
d = "word"
d.sort()
```

lsta is [6, 2, 9, 4, 3]

WOTO-1 Sorting http://bit.ly/101s22-0224-1

Python Sets

- **Set – unordered collection of distinct items**
  - Unordered – can look at them one at a time, but cannot count on any order
  - Distinct - one copy of each

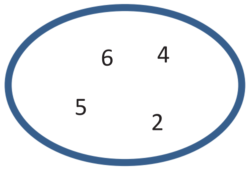
```
x = [5, 3, 4, 3, 5, 1]
y = set(x)

y.add(6)
y.add(4)
```

x is [5, 3, 4, 3, 5, 1]

List vs Set

- List
  - Ordered, 3rd item, can have duplicates
  - Example: `x =[4, 6, 2, 4, 5, 2, 4]`
- Set
  - No duplicates, no ordering
  - Example: `y = set(x)`
- Both
  - Add, remove elements
  - Iterate over all elements



Python Sets

- Can convert list to set, set to list
  - Great to get rid of duplicates in a list

a = [2, 3, 6, 3, 2, 7]   a is [2, 3, 6, 3, 2, 7]
b = set(a)

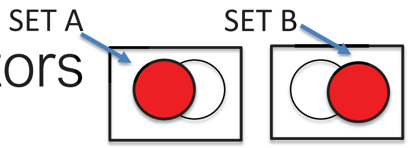
c = list(b)

Python Sets

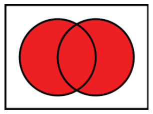
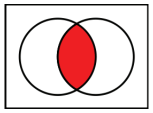
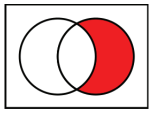
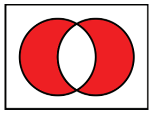
- Operations on sets:
  - Modify:
    - add a.add(7)
    - clear a.clear()
    - remove a.remove(5)
  - Create a new set: a = set([])

  - difference(-), intersection(&), union (|), symmetric_difference(^)
  - Boolean: issubset <=, issuperset >=

Python Set Operators



- Using sets and set operations often useful
- A | B, set union
  - Everything
- A & B, set intersection
  - *Only* in both
- B – A, set difference
  - In B *and* not A
- A ^ B, symmetric diff
  - Only in A *or* only in B

List and Set, Similarities/Differences

	Function for List	Function for Set
Adding element	`x.append(elt)`	`x.add(elt)`
Size of collection	`len(x)`	`len(x)`
Combine collections	`x + y`	`x | y`
Iterate over	`for elt in x:`	`for elt in x:`
Element membership	`elt in x`	`elt in x`
Index of an element	`x.index(elt)`	`CANNOT DO THIS`

- Lists are ordered and indexed, e.g., has a first or last
- Sets are not ordered, very fast, e.g., **`if elt in x`**

Creating and changing a set

```
colorList = ['red', 'blue', 'red', 'red', 'green']
colorSet = set(colorList)
smallList = list(colorSet)
colorSet.clear()
colorSet.add("yellow")
colorSet.add("red")
colorSet.add("blue")
colorSet.add("yellow")
colorSet.add("purple")
colorSet.remove("yellow")
```

smallList is

Set Operations – Union and Intersection

```
UScolors = set(['red', 'white', 'blue'])
dukeColors = set(['blue', 'white', 'black'])

print(dukeColors | UScolors)
print(dukeColors & UScolors)
```

Set Operations - Difference

```
UScolors = set(['red', 'white', 'blue'])
dukeColors = set(['blue', 'white', 'black'])

print( dukeColors - UScolors)
print( UScolors - dukeColors)
```

Set Operations – Symmetric Difference

```
UScolors = set(['red', 'white', 'blue'])
dukeColors = set(['blue', 'white', 'black'])

print(dukeColors ^ UScolors)
print(UScolors ^ dukeColors)
```

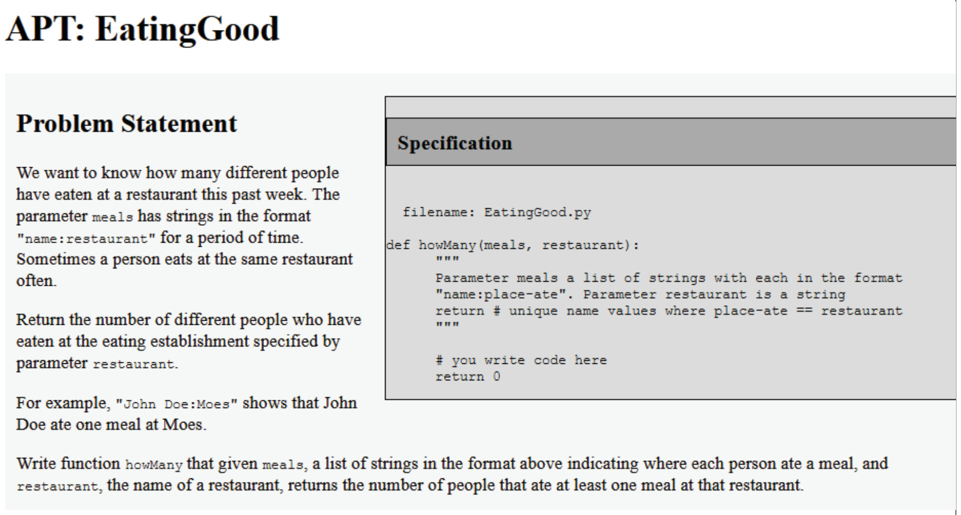
Let's sort lists with sorted() function

- Built-in function: sorted(*sequence*)
  - Returns new list of sequence in sorted order
  - Sequence could be list, tuple, string
  - Sequence could be set!

a = set([3, 5, 2, 1, 7, 2, 5)]
b = sorted(a)

WOTO-2 Sets
http://bit.ly/101s22-0224-2

APT Eating Good



APT: EatingGood

Problem Statement

We want to know how many different people have eaten at a restaurant this past week. The parameter `meals` has strings in the format `"name:restaurant"` for a period of time. Sometimes a person eats at the same restaurant often.

Return the number of different people who have eaten at the eating establishment specified by parameter `restaurant`.

For example, `"John Doe:Moes"` shows that John Doe ate one meal at Moes.

Specification

```
  filename: EatingGood.py

def howMany(meals, restaurant):
      """
      Parameter meals a list of strings with each in the format
      "name:place-ate". Parameter restaurant is a string
      return # unique name values where place-ate == restaurant
      """

      # you write code here
      return 0
```

Write function `howMany` that given `meals`, a list of strings in the format above indicating where each person ate a meal, and `restaurant`, the name of a restaurant, returns the number of people that ate at least one meal at that restaurant.

APT Eating Good Example

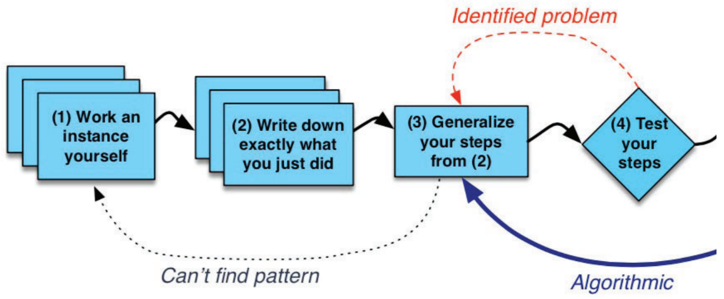
```
meals = ["Sue:Elmos", "Sue:Elmos", "Sue:Elmos"]

restaurant = "Elmos"

returns 1
```

WOTO-3: APT Eating Good http://bit.ly/101s22-0224-3

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


Identified problem
(1) Work an instance yourself
(2) Write down exactly what you just did
(3) Generalize your steps from (2)
(4) Test your steps
Can't find pattern
Algorithmic

APT Eating Code Idea