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 Gigabytes!
• Mouse, Mouse pad
  • Easier to navigate

Margot Shetterly

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

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
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

```python
lst = [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

```python
lst = (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
  - List is `[17, 7, 13, 3]`
  - List.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()`

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

ista = [6, 2, 9, 4, 3]
ista.sort()
a = [7, 2, 9, 1]
b = a.sort()
c = (5, 6, 2, 1)
c.sort()
d = "word"
d.sort()
```

---

WOTO-1 Sorting

---

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

```python
x = [5, 3, 4, 3, 5, 1]
x is [5, 3, 4, 3, 5, 1]
y = set(x)
y = set(x)
y.add(6)
y.add(4)
```
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: \( y = \text{set}(x) \)

- **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

```python
a = [2, 3, 6, 3, 2, 7]

b = set(a)

c = list(b)
```

Python Sets

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

  - 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

<table>
<thead>
<tr>
<th></th>
<th>Function for List</th>
<th>Function for Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding element</td>
<td><code>x.append(elt)</code></td>
<td><code>x.add(elt)</code></td>
</tr>
<tr>
<td>Size of collection</td>
<td><code>len(x)</code></td>
<td><code>len(x)</code></td>
</tr>
<tr>
<td>Combine collections</td>
<td><code>x + y</code></td>
<td>`x</td>
</tr>
<tr>
<td>Iterate over</td>
<td><code>for elt in x:</code></td>
<td><code>for elt in x:</code></td>
</tr>
<tr>
<td>Element membership</td>
<td><code>elt in x</code></td>
<td><code>elt in x</code></td>
</tr>
<tr>
<td>Index of an element</td>
<td><code>x.index(elt)</code></td>
<td>CANNOT DO THIS</td>
</tr>
</tbody>
</table>

- 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

```python
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

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

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

Set Operations - Difference

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

print(dukeColors - UScolors)
p
```
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)
APT Eating Good Example

```python
meals = ["Sue:Elmos", "Sue:Elmos", "Sue:Elmos"]
restaurant = "Elmos"
returns 1
```

WOTO-3: APT Eating Good


- [https://www2.cs.duke.edu/csed/pythonapt/eatinggood.html](https://www2.cs.duke.edu/csed/pythonapt/eatinggood.html)

APT Eating Code Idea