Compsci 101
Stable Sorting, Lambda

\[ f = \text{lambda } x : x[1] \]
\[ \text{sorted(lst, key=f)} \]

Yesenia Velasco
Susan Rodger
March 29, 2022

S is for …

- Software
  - Joy, sorrow, fun, changing the world
- System and sys
  - Connecting to the machine at different levels
- Sorting
  - From hat to tim to more

Shaundra Daily

- Professor of the Practice, Duke University
- B.S. Florida State, Electrical Eng
- PhD Media Arts/Sciences – MIT
- Combines Dance with Robotics
- Focuses on technologies, programs and curricula to support Diversity, Equity and Inclusion in STEM Fields

Announcements

- Assignment 5 due Today!
- APT-6 due Thurs. March 31
- APT-7 out Thursday, Due April 7
- Lab 10 Friday
  - There is a prelab, it is out!
- Coming up…
  - APT Quiz 2 – April 7-10
  - Exam 4 – April 12
PFTD

• Sorting in Python and sorting in general
  • How to use .sort and sorted, differences
  • Key function – change how sorting works
  • Lambda – create anonymous functions

• Stable sorting
  • How to leverage when solving problems
  • Why Timsort is the sort-of-choice (! quicksort)

Review Sort: Items Same Length

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

```python
a = ['red', 'orange', 'green', 'blue', 'indigo', 'violet']
print(sorted(a))
print(sorted(a, key=len))
print(sorted(a, key=len, reverse=True))
```

Review Sort: Items Same Length

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

```python
a = ['red', 'orange', 'green', 'blue', 'indigo', 'violet']
print(sorted(a))
print(sorted(a, key=len))
print(sorted(a, key=len, reverse=True))
```

Sorting Examples

```python
a = [4, 1, 7, 3]
b = sorted(a)
a.sort()
a = ['Q', 'W', 'B', 'F']
b = sorted(a)
c = sorted(a, reverse = True)
a = ['hello', 'blue', 'car']
b = sorted(b, key=len)
```
### Sorting Examples

```
# Example 1
a = [4, 1, 7, 3]
b = sorted(a)
a.sort()
```

```
# Example 2
a = ['Q', 'W', 'B', 'F']
b = sorted(a)
c = sorted(a, reverse=True)
a = ['hello', 'blue', 'car']
b = sorted(b, key=len)
```

### More Sorting Examples

```
# Example 3
a = [4, 1, 7, 3]
b = [1, 3, 4, 7]
```

```
# Example 4
a = ['Q', 'W', 'B', 'F']
b = ['B', 'F', 'Q', 'W']
c = ['W', 'Q', 'F', 'B']
```

```
# Example 5
a = ['Q', 'W', 'B', 'F']
b = [1, 3, 4, 7]
```

```
# Example 6
a = ['hello', 'blue', 'car']
b = ['car', 'blue', 'hello']
```

### More Sorting Examples

```
# Example 7
a = [2, 2, 34], [2, 6, 7, -1], [1, 2, 3]
b = sorted(a)
```

```
# Example 8
a = [2, 2, 34], [2, 6, 7, -1], [1, 2, 3]
c = sorted(a, key=len)
d = sorted(a, key=max)
e = sorted(a, key=min)
```

### WOTO-1 Basic Sorting

WOTO – 1st question

Showing the list and the list sorted

```python
In[14]: a = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet']
In[15]: sorted(a)
Out[15]: ['blue', 'green', 'indigo', 'orange', 'red', 'violet', 'yellow']
```

What’s the list returned by `sorted(a, reverse=True)`? *

- [yellow, violet, red, orange, indigo, green, blue]
- [violet, indigo, blue, green, yellow, orange, red]

WOTO – 2nd question

Showing the list and the list sorted

```python
In[14]: a = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet']
In[15]: sorted(a)
Out[15]: ['blue', 'green', 'indigo', 'orange', 'red', 'violet', 'yellow']
```

What’s the list returned by `sorted(a, key=len)`? *

- ['red', 'blue', 'green', 'orange', 'yellow', 'indigo', 'violet']
- ['red', 'blue', 'orange', 'green', 'yellow', 'indigo', 'violet']
WOTO – 3rd question

Showing the list and the list sorted

In[14]: a = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet']
In[15]: sorted(a)
Out[15]: ['blue', 'green', 'indigo', 'orange', 'red', 'violet', 'yellow']

The function max applied to a string returns the alphabetically greatest character in the string, so max('indigo') == 'o' and max('yellow') == 'y'. What's the list returned by sorted(a, key=max)?

- [indigo, orange, green, red, blue, violet, yellow]
- [indigo, red, orange, green, blue, violet, yellow]

3/29/22
Compsci 101, Spring 2022 17

The power of lambda

- We want to create a function "on-the-fly"
  - aka anonymous function
  - aka "throw-away" function

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? Later today! 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
Why is lambda used?

• It doesn't matter at all could use zeta? iota? ...
  • https://en.wikipedia.org/wiki/Alonzo_Church

• Lisp and Scheme have lambda expressions
• Guido van Rossum, learned to live with lambda

What is a lambda expression?

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

```python
>>> 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

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

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

Syntax and Semantics of Lambda

• Major use: single variable function as key

```python
fruits = ['banana', 'apple', 'lemon', 'kiwi', 'pineapple']
b = sorted(fruits)
c = min(fruits)
d = max(fruits)
```
Syntax and Semantics of Lambda

• Major use: single variable function as key

```
fruits = ['banana', 'apple', 'lemon', 'kiwi', 'pineapple']
b = sorted(fruits)
  b: ['apple', 'banana', 'kiwi', 'lemon', 'pineapple']
c = min(fruits)
c: 'apple'
d = max(fruits)
d: 'pineapple'
```

Syntax and Semantics of Lambda (2)

```
fruits = ['banana', 'apple', 'lemon', 'kiwi', 'pineapple']
e = min(fruits, key=lambda f: len(f))
e: 'kiwi'
g = max(fruits, key=lambda z: z.count('e'))
g: 'pineapple'
h = sorted(fruits, key=lambda z: z.count('e'))
h: ['banana', 'kiwi', 'apple', 'lemon', 'pineapple']
```

Review: CSV and Sort for top artists

• Using two-sorts to get top artists

```
print('
Top 5 artists:
', sortedArtists[-5:])
```

• Reverse tuples to sort
• Reverse tuples to print

```
```
Top 5 Artists

• Instead of intermediary list, use lambda
• Instead of [-5:], use reverse=True

```python
print('Top 5 artists: ')
sortbycount = sorted([(a[1], a[0]) for a in counts.items()])
sortedArtists = [(a[1], a[0]) for a in sortbycount]
for artist in sortedArtists[-5:]:
    print(artist)

print("repeat it")
sortedArtists = sorted(counts.items(), key=lambda item: item[1], reverse=True)
for tup in sortedArtists[:5]:
    print(tup)
```

Output slightly different. Why?