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))
```

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)
```

More Sorting Examples

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

c = sorted(a, key = len)
d = sorted(a, key=max)
e = sorted(a, key=min)
```
We want to create a function "on-the-fly"
  • aka anonymous function
  • aka "throw-away" function

```python
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]
```

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 (2)

```python
fruits = ['banana', 'apple', 'lemon', 'kiwi', 'pineapple']
e = min(fruits, key=lambda f: len(f))
g = max(fruits, key=lambda z: z.count('e'))
h = sorted(fruits, key=lambda z: z.count('e'))
```
Review: CSV and Sort for top artists

- Using two sorts to get top artists

```python
print('nTop 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)
```

- Reverse tuples to sort
- Reverse tuples to print

Top 5 artists:
('John, Elton', 21)
('Who', 24)
('Rolling Stones', 36)
('Led Zeppelin', 38)
('Beatles', 51)

Output slightly different. Why?

Top 5 Artists

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

```python
print('nTop 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)
```

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

repeat it
('Beatles', 51)
('Led Zeppelin', 38)
('Rolling Stones', 36)
('Who', 24)
('Eagles', 21)

How is the sorting happening?

```python
>>> d
{'a': [1, 2, 3], 'b': [4, 7], 'c': [1, 1, 5, 8]}
>>> sorted(d.items())

>>> sorted(d.items(), key=lambda x: x[1])

>>> sorted(d.items(), key=lambda x: x[1][-1])
```
How to do some “fancy” sorting

- **lambda PARAMETER :** EXPRESSION

- Given data: list of tuples: (first name, last name, age)
  
  `[(‘Percival’, ‘Avram’, 51),
   (‘Melete’, ‘Sandip’, 24), ...]

- Think: What is the lambda key to sort the following?

  `sorted(data, key=lambda z : (z[0],z[1],z[2]))`

  - Sort by last name, break ties with first name
  - Sort by last name, break ties with age
  - Alphabetical by last name, then first name, then reverse age order

Creating Tuples with lambda

- Sort by last name, break ties with first name
- Sort by last name, break ties with age
- Alphabetical by last name, then first name, then reverse age order

Leveraging the Algorithm

- Can’t sort by creating a tuple with lambda, use:
  - Pattern: Multiple-pass **stable** sort – first sort with last tie breaker, then next to last tie breaker, etc. until at main criteria

- Sort by index 0, break tie in reverse order with index 1

  `[(‘b’, ‘z’), (‘c’, ‘x’), (‘b’, ‘x’), (‘a’, ‘z’)]`

- **Stable** sort respects original order of "equal" keys

Stable sorting: respect "equal" items

- Women before men, each group height-sorted
  - First sort by height

3/29/22 Compsci 101, Spring 2022 35

3/29/22 Compsci 101, Spring 2022 36

3/29/22 Compsci 101, Spring 2022 38

3/29/22 Compsci 101, Spring 2022 41
Understanding Multiple-Pass Sorting

```python
> data
[('f', 2, 0), ('e', 1, 4), ('a', 2, 0), ('c', 2, 5), ('b', 3, 0), ('d', 2, 4)]
> a0 = sorted(data, key = lambda x: x[0])
> a0

> a1 = sorted(a0, key = lambda x: x[2])
> a1

> a2 = sorted(a1, key = lambda x: x[1])
> a2
```

WOTO-3 Multipass Sorting