Lecture 11: Sorting with Lambda and Plotting

Check-in and Reminders

- Pick up **Homework 4** from the box up front
- Reminder: Lab 3 due tonight/tomorrow night
- My office hours today 12.30 - 2 pm @ CS common room
- Midterm information:
    - Room: **TPL 203**
    - Time: **5.45 pm - 7.45 pm** or **8-10 pm**
    - **Closed book exam**
    - **Review your homework**: best practice with pencil and paper and coding questions
    - Review lecture material including Jupyter notebook examples
- Next week lab will be shorter and be done the day off

Do You Have Any Questions?

Midterm Syllabus and Topics

- Topics included: everything we cover up to to today's lecture
- Homework 1: Expressions & Functions, Return vs Print
- Homework 2: Booleans & Loops over Sequences, Simplifying conditionals, List indexing, etc.
- Homework 3: Strings and Mutability
- Homework 4: Tuples, Dict (get), List comprehension, Lambda sorting

- From labs:
- Writing functions, file reading; strip, split; sorting, strings; len; finding max; counters in loops; doctests, __all__, modules/scripts, if __name__=='__main__', etc.
- **Pretty much everything up to and including Lab 4 & Homework 4**

Do You Have Any Questions?

Today's Highlights

Sorting with Lambda!

- How to sort data in Python in all kinds of ways!
- sorted() function provides an optional key parameter
- We can use that lambda expression to implement cool sorting functionalities!

Plotting Data using matplotlib!

- We will learn how to visualize data using Python's matplotlib library
- We will plot the frequency of the common words in Pride and Prejudice!

Review: Sorted() Function

- sorted() function takes any object like lists, strings, tuples, dictionaries and returns a **new sorted list**
- Sorting list of numbers (sorts in ascending order)

```
In [1]: numbers = [35, -2, 17, -9, 0, 12, 19]
        sorted(numbers)
```

```
Out[1]: [-9, -2, 0, 12, 17, 19, 35]
```

- Sorting strings (sorts in ascending based on ASCII ordering)

```
In [2]: phrase = 'Red Code 1'
        sorted(phrase) # sorted based on ascii ordering
```

```
Out[2]: [' ', ' ', '1', 'C', 'R', 'd', 'd', 'e', 'e', 'o']
```

Review: Sorted() Function

- sorted() function takes any object like lists, strings, tuples, dictionaries and returns a **new sorted list**
- **Sorting a list of tuples:**
    - Tuples are compared element by element (starting with one at index 0), known as lexicographical order

In [3]:
```
triples = [(8, 'a', '$'), (7, 'c', '@'),
           (7, 'b', '+'), (8, 'a', '!')]

sorted(triples)
```

Out[3]:
```
[(7, 'b', '+'), (7, 'c', '@'), (8, 'a', '!'), (8, 'a', '$')]
```

Question. How do we sort based on the second item in tuples?

Sorted() Function: List of Lists

- sorted() function takes any object like lists, strings, tuples, dictionaries and returns a **new sorted list**
- **Sorting a list of lists:**
    - Lists are compared element by element (starting with one at index 0) in lexicographical order

```
In [2]: listOfLists = [[3, 1], [3, 10], [15, 2]]
        sorted(listOfLists)

Out[2]: [[3, 1], [3, 10], [15, 2]]
```

Question. How do we sort based on the second item in the lists?

Sorted() Function: Dictionaries

- **Sorting dictionaries:** what happens if we sort a dictionary (which don't have any inherent order)?

```
In [4]: daysOfMonth = {'Jan': 31, 'Feb': 28, 'Mar': 31, 'Apr': 30,
                       'May': 31, 'Jun': 30, 'Jul': 31, 'Aug': 31,
                       'Sep': 30, 'Oct': 31, 'Nov': 30, 'Dec': 31}
```

```
In [5]: sorted(daysOfMonth)
```

```
Out[5]: ['Apr',
         'Aug',
         'Dec',
         'Feb',
         'Jan',
         'Jul',
         'Jun',
         'Mar',
         'May',
         'Nov',
         'Oct',
         'Sep']
```

Question. **How do we sort based on values?**

Sorted() Function: List of Dicts

- Suppose I wanted to sort the following list of dictionaries with fruit information by say the weight of the fruits
- **List of dictionaries:** what happens if we sort a list of dictionaries?

```
In [13]: fruitDictsList = [{'name': 'apple', 'weight': 20},
                           {'name': 'orange', 'weight': 5},
                           {'name': 'kiwi', 'weight': 15.7}]
         sorted(listOfDicts)
         # by default Python does not know how to compare two dicts
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-3164ca5da260> in <module>
      2                           {'name': 'orange', 'weight': 5},
      3                           {'name': 'kiwi', 'weight': 15.7}]
----> 4 sorted(listOfDicts)
      5 # by default Python does not know how to compare two dicts

TypeError: '<' not supported between instances of 'dict' and 'dict'
```

Question. **How do we sort these based on, e.g., the weights?**

Sorting with the Key Parameter

- Sorted takes several parameters: type `help(sorted)` in interactive python or `pydoc3 sorted` on the terminal to find out more
- First parameter is an “iterable”, meaning, any object over which we can iterate (list, string, tuple).
- We have already seen the parameter **reverse**
- **key** specifies a **function** that for each element determines how it should be compared to other elements.

```
In [3]: help(sorted)
```

```
Help on built-in function sorted in module builtins:

sorted(iterable, /, *, key=None, reverse=False)
    Return a new list containing all items from the iterable in ascending
order.

    A custom key function can be supplied to customize the sort order, an
d the
    reverse flag can be set to request the result in descending order.
```

Sorting with the Key Parameter

- Let's sort the following tuples of NASA mission names and number of days by the number of days key

In [21]:
```
missionTuples = [('Apollo 11', 8), ('Mercury-Redstone 3', 1),
                 ('Apollo 13', 5), ('Gemini 3', 1), ('Little Joe 6', 1)]
```

In [20]:
```
# defining a function that returns item at index 1
def days(missions):
    """Takes a sequence and returns the item at index 1"""
    return missions[1]
```

Sort each tuple by number of days

In [23]:
```
sorted(missionTuples, key=days) # sort by num of days
```

Out[23]:
```
[('Mercury-Redstone 3', 1),
 ('Gemini 3', 1),
 ('Little Joe 6', 1),
 ('Apollo 13', 5),
 ('Apollo 11', 8)]
```

Notice order of items with same number of days

Python Sorting Functions are Stable

- Python's sorting functions are **stable**, which means that items that are equal according to the sorting key have the same relative order as in the original list.

In [21]:
```
missionTuples = [('Apollo 11', 8), ('Mercury-Redstone 3', 1),
                 ('Apollo 13', 5), ('Gemini 3', 1), ('Little Joe 6', 1)]
```

In [20]:
```
# defining a function that returns item at index 1
def days(missions):
    """Takes a sequence and returns the item at index 1"""
    return missions[1]
```

Sort each tuple by number of days

In [23]:
```
sorted(missionTuples, key=days) # sort by num of days
```

Out[23]:
```
[('Mercury-Redstone 3', 1),
 ('Gemini 3', 1),
 ('Little Joe 6', 1),
 ('Apollo 13', 5),
 ('Apollo 11', 8)]
```

Notice order of items with same number of days

Breaking Ties with Sorting Key

- Python's sorting functions are **stable**, which means that items that are equal according to the sorting key have the same relative order as in the original list.

In [28]:
```
# A key function that returns a tuple to specify
# lexicographic ordering by the elements of that tuple.
def daysProgram(missions):
    return (days(missions), programName(missions))

sorted(missionTuples, key=daysProgram)
# sort first by days and then alphabetically by name
```

Out[28]:
```
[('Gemini 3', 1),
 ('Little Joe 6', 1),
 ('Mercury-Redstone 3', 1),
 ('Apollo 13', 5),
 ('Apollo 11', 8)]
```

Sort first by days, then by program name

Notice the order

`lambda` Notation: Anonymous Functions

- It is often inconvenient or unnecessary to define a named function just in order to pass it as the functional argument to higher-order functions (HOFs) like `sorted`
- Python provides `lambda` notation for creating **anonymous functions** (a function with no name that cannot be called elsewhere) that can be used directly in functions like `sorted`

```
In [30]: sorted(missionTuples, key=lambda missions: missions[1])

Out[30]: [('Mercury-Redstone 3', 1),
          ('Gemini 3', 1),
          ('Little Joe 6', 1),
          ('Apollo 13', 5),
          ('Apollo 11', 8)]
```

function takes missions as parameter and returns the item at index 1

Anatomy of a `lambda` Expression

A **`lambda`** expression has the form:

```
lambda param: bodyExpression
```

Keyword meaning "I am a function"

parameter name of this function

expression for result of this function. It does **not** use an explicit **`return`**!

```
lambda num: num*2
```

I am a function

that takes an argument named num

and returns the result of doubling it

```
lambda n: (n%2)==0
```

I am a function

that takes an argument named n

and returns a boolean that indicates whether it's even

Image Source: (http://cs111.wellesley.edu/spring19)

Why Lambda?

- In the 1930s and 40s, **Alonzo Church** developed a model of computation called **the lambda calculus**

- It is a programming language with only three kinds of expressions:
  - variables, e.g. **x**
  - functions expressed in lambda notation, e.g. λ x . x
  - function application, e.g., (λ x . x)(y)
- Remarkably, this simple language can express any computable program – even though it has no built-in numbers, arithmetic, booleans, conditionals, lists, loops, or recursion! (To find out more, take CS 334 or CS 361!)

More Sorting with Lambda

- Let's use sorting with lambda to sort the missionTuples first by the number of days and then by the length of the program Names

```
In [21]: missionTuples = [('Apollo 11', 8), ('Mercury-Redstone 3', 1),
                          ('Apollo 13', 5), ('Gemini 3', 1), ('Little Joe 6', 1)]
```

```
In [30]: sorted(missionTuples, key=lambda missions: missions[1])
```

```
Out[30]: [('Mercury-Redstone 3', 1),
          ('Gemini 3', 1),
          ('Little Joe 6', 1),
          ('Apollo 13', 5),
          ('Apollo 11', 8)]
```

sorting by days

```
In [31]: sorted(missionTuples,
                key=lambda mTup: (mTup[1], len(mTup[0])))
```

```
Out[31]: [('Gemini 3', 1),
          ('Little Joe 6', 1),
          ('Mercury-Redstone 3', 1),
          ('Apollo 13', 5),
          ('Apollo 11', 8)]
```

sorting by days and length of name

Sorting Dictionaries by Value

- Recall our earlier question: how do we sort a dictionary by its values?

In [32]:
```
daysOfMonth = {'Jan': 31, 'Feb': 28, 'Mar': 31, 'Apr': 30,
               'May': 31, 'Jun': 30, 'Jul': 31, 'Aug': 31,
               'Sep': 30, 'Oct': 31, 'Nov': 30, 'Dec': 31}
```

In [33]:
```
sorted(daysOfMonth, key=daysOfMonth.get) # sort by values
```

Out[33]:
```
['Feb',
 'Apr',
 'Jun',
 'Sep',
 'Nov',
 'Jan',
 'Mar',
 'May',
 'Jul',
 'Aug',
 'Oct',
 'Dec']
```

Use the predefined get method for dictionaries!

Sorting a List of Dictionaries

- Recall the the following lists of dictionaries where we want to sort them by their weights

```
In [34]: fruitDicts = [{'name': 'apple', 'weight': 20},
                       {'name': 'orange', 'weight': 15},
                       {'name': 'kiwi', 'weight': 15}]
         # by default Python does not know how to compare two dicts
```

```
In [35]: sorted(fruitDicts, key=lambda fruitDict: fruitDict["weight"])
```

```
Out[35]: [{'name': 'orange', 'weight': 15},
          {'name': 'kiwi', 'weight': 15},
          {'name': 'apple', 'weight': 20}]
```

```
In [36]: sorted(fruitDicts, key=lambda fDict: (fDict["weight"], len(fDict["name"])))
```

```
Out[36]: [{'name': 'kiwi', 'weight': 15},
          {'name': 'orange', 'weight': 15},
          {'name': 'apple', 'weight': 20}]
```

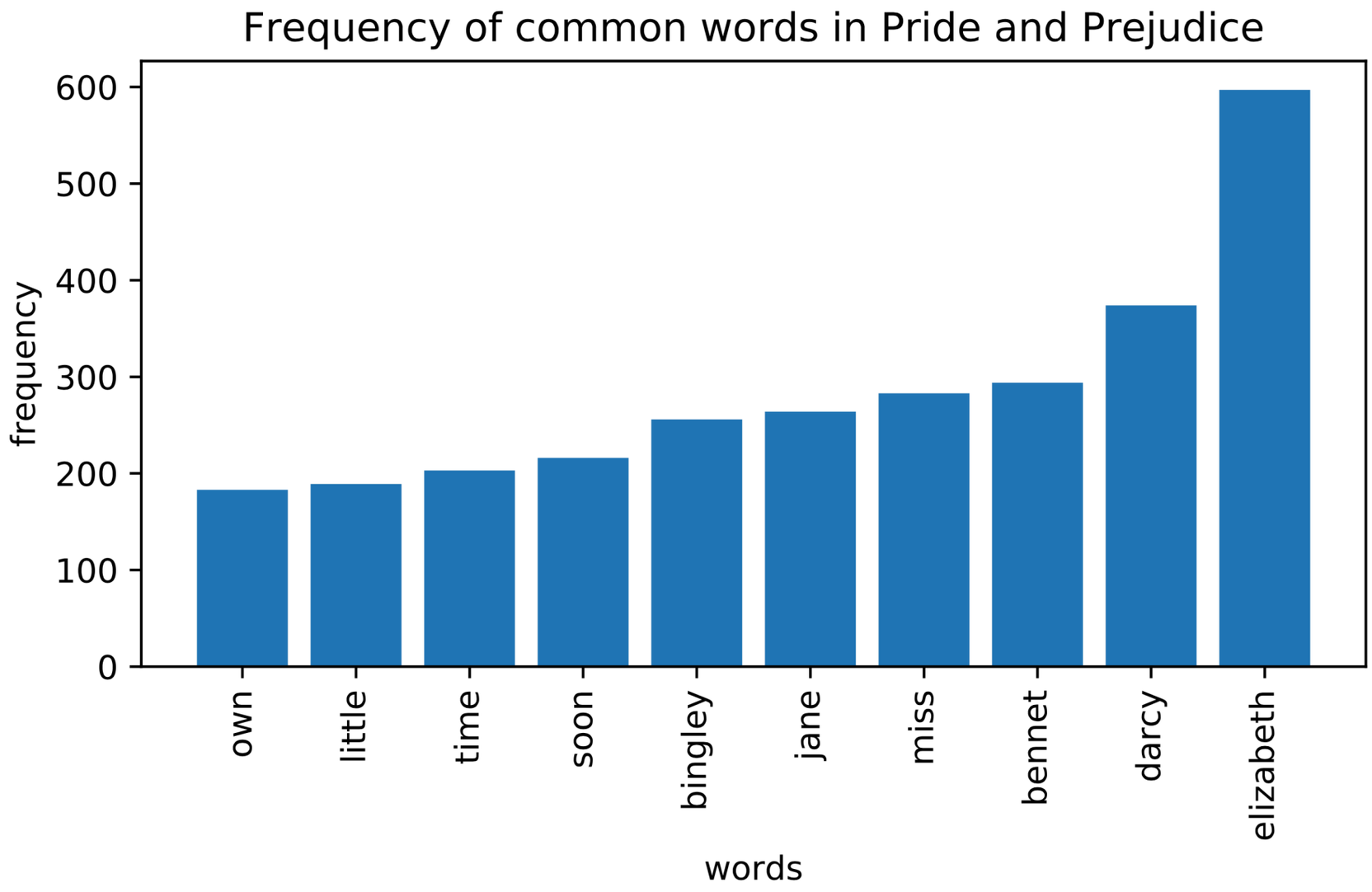
sort by weight and then by length of name

Plotting with `matplotlib`

- A plot is a graphical technique for representing a data set, usually as a graph showing the relationship between two or more variables
- We'll be using Python's `matplotlib` library to make plots/graphs/charts
- The best way to learn how to plot different types of graphs is to read the documentation and see examples
- Resources
  - **matplotlib examples**: http://matplotlib.org/examples
  - **pyplot documentation**: http://matplotlib.org/api/pyplot_summary.html
  - cool plots: https://matplotlib.org/gallery.html

`matplotlib` Examples: See Notebook

- See lecture Jupyter notebook for variable matplotlib examples
- We will plot the bar chart for the common words in Pride and Prejudice and their frequency !



Acknowledgments

These slides have been adapted from:

- http://cs111.wellesley.edu/spring19 and
- https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/