Lambdas and Custom Sort

CS106AP Lecture 25
Roadmap

- Programming Basics
- The Console
- Images
- Data structures
- Midterm
- Graphics
- Object-Oriented Programming
- Everyday Python
- Life after CS106AP!
Roadmap

- Programming Basics
- The Console
- Data structures
- Images
- Life after CS106AP!

Day 1!

- Graphics
- Midterm
- Object-Oriented Programming

Everyday Python

- Tuples
- List Comp.
- Lambdas
- Internet
- Computers
- Life After Final Exam
Today's questions

How can we write operations that help us better organize and process information inside data structures?

How can we visualize our data?
Today’s topics

1. Review
2. Lambdas
   Map, Filter
   Sorted, Min, Max
3. Matplotlib
4. What’s next?
Review
List Comprehensions
List Comprehensions

Definition

A way to create a list based on existing lists
List Comprehensions

\[ n ** 2 \text{ for } n \text{ in nums} \]

- Reuses syntax from other features:
  - `[]` to create new list
  - `foreach` loop over other list

**Definition**

List Comprehension
A way to create a list based on existing lists
Combining functions with list comprehensions

def name_case(s):
    return s[0].upper() + s[1:].lower()

strings = [‘SONja’, ‘nicHOLAs’, ‘KYLiE’]
name_strings = [name_case(s) for s in strings]
Conditions in list comprehensions

- You can add a condition for additional “filtering”

```python
[n for n in nums if n % 2 == 0]
```
Why list comprehensions?

- They’re more concise
- They’re faster
- They’re Pythonic
When to **not** use list comprehensions

- When you need more than one condition

- When the expression is complex
  - Break it out into a separate function!
Dictionary
Comprehensions
Dict Comprehensions

d = {'a': 1, 'b': 2, 'c': 3, 'd': 4}

flipped = {v:k for (k, v) in d.items()}
Dict Comprehensions

d = {'a': 1, 'b': 2, 'c': 3, 'd': 4}

flipped = {v:k for (k, v) in d.items()}

Two differences:

{} instead of []

key expression:val expression
Jupyter Notebooks
Jupyter Notebook

- Interactive “notebook” where you can run parts of code
  - Can develop code step-by-step
  - Great for data analysis
- Built on top of regular Python code
Jupyter Notebook Setup

$ python3 -m pip install jupyter
How can we write operations that help us better organize and process information inside data structures?
Recall: sorting lists with tuples

```python
>>> fruit = [('mango', 3), ('apple', 6), ('lychee', 1), ('apricot', 10)]
>>> sorted(fruit)
[('apple', 6), ('apricot', 10), ('lychee', 1), ('mango', 3)]
```

sorts by the first element in each tuple
Recall: sorting lists with tuples

>>> fruit = [('mango', 3), ('apple', 6), ('lychee', 1), ('apricot', 10)]

>>> sorted(fruit)
[('apple', 6), ('apricot', 10), ('lychee', 1), ('mango', 3)]

what if we want to sort by the second element in the tuple?
Lambda Functions
Lambda Functions

\(\lambda n: n \times 2\)

\(\lambda x, y: x \times y\)

\(\lambda\) tup: tup[0]
Lambda Functions

lambda n: n * 2

lambda x, y: x ** y

Definition

Lambda
A one-line, unnamed function

Note:
no def, no return
Lambdas vs. Regular Functions

lambda x: x * 2

def double(x):
    return x * 2
Lambdas vs. Regular Functions

\[
\text{lambda } x: x \times 2
\]

\[
\text{def double(x):}
    \text{return } x \times 2
\]
Lambdas vs. Regular Functions

\[
\text{lambda } x: x * 2
\]

\[
\text{def double}(x):
\]

\[
\text{return } x * 2
\]
Lambdas vs. Regular Functions

lambda x: x * 2

def double(x):
    return x * 2
**Lambdas**

```python
lambda x: x * 2
```

*this expression is automatically returned*

**Regular Functions**

```python
def double(x):
    return x * 2
```
**Lambdas** vs. **Regular Functions**

Lambdas:

\[
\text{lambda } x: x \times 2
\]

This expression is automatically returned.

Regular Functions:

```python
def double(x):
    return x \times 2
```

We need `return` in order to return!
How can I use a lambda function? - map()

- **map(function, list)**
  - calls (lambda) function once for each element in the list
  - returns a list containing the output of each function
  - like `[function(x) for x in list]`
  - but returns an *iterable*
    - use `list(map(fn, lst))` to get a list
How can I use a lambda function? - map()

```python
# usage: map(function, list)
>>> nums = [1, 3, 6, 7]
>>> squared = map(lambda n: n ** 2, nums)
>>> list(squared)
[1, 9, 36, 29]
```

we have to use `list()` because `map` returns an iterable

lambda function
How can I use a lambda function? - map()

- Say we have a list of strings, and we want a list of the strings’ lengths.

  in: ['i', 'rly', 'love', 'breakout']

  out: [1, 3, 4, 8]

Think/Pair/Share:
How would you produce the output list using map()?
How can I use a lambda function? - map()

# usage: map(function, list)

```python
>>> lst = ['i', 'rly', 'love', 'breakout']
>>> lengths = map(len, lst)
>>> list(lengths)
[1, 3, 4, 8]
```
How can I use a lambda function? - filter()

- **filter**(*function*, *list*)
  - calls (lambda) function once for each element in the list
  - function is a boolean that acts as a filter
    - if it doesn’t evaluate to True, exclude the element
  - like [*x for x in list if function(x)*]
How can I use a lambda function? - filter()

# usage: filter(function, list)

```python
>>> nums = [4, 23, 9, 18, 63, 42]
```

```python
>>> even = filter(lambda n: n % 2 == 0, nums)
```

```python
>>> list(even)
```

```python
[4, 18, 42]
```
Why lambdas?

- Powerful in the context of custom sort and min/max
- Great for when you need a tiny function
- Use less memory than regular functions in Python
How can I use a lambda function? - sorted()

- `sorted(iterable, key, reverse)`
  - `key` is where you can pass in a lambda
  - `key` function transforms each element before sorting
    - it outputs the value to use for comparison when sorting

key and reverse are optional arguments
Recall: sorting lists with tuples

```python
>>> fruit = [('mango', 3), ('apple', 6), ('lychee', 1), ('apricot', 10)]
>>> sorted(fruit)
[('apple', 6), ('apricot', 10), ('lychee', 1), ('mango', 3)]
```

what if we want to sort by the second element in the tuple?
Recall: sorting lists with tuples

```python
>>> fruit = [('mango', 3), ('apple', 6), ('lychee', 1), ('apricot', 10)]
>>> sorted(fruit)
[('apple', 6), ('apricot', 10), ('lychee', 1), ('mango', 3)]

# get the second value from the tuple and sort on it
>>> sorted(fruit, key=lambda elem: elem[1])
[('lychee', 1), ('mango', 3), ('apple', 6), ('apricot', 10)]
```
How can I use a lambda function? - sorted()

- Say we have a list of strings, and we want to sort them alphabetically by the last character in the string.

  in: ['llamas', 'love', 'my', 'lambda']

  out: ['lambda', 'love', 'llamas', 'my']

Think/Pair/Share: How would you produce the output list using sorted()?
How can I use a lambda function? - sorted()

```python
>>> lst = ['llamas', 'love', 'my', 'lambda']
>>> sorted(lst, key=lambda s: s[len(s)-1])
['lambda', 'love', 'llamas', 'my']
```
How can I use a lambda function? - sorted()

- Say we have a list of strings, and we want to sort them by length.

  in: ['lambdas', 'are', 'so', 'cool!']
  out: ['so', 'are', 'cool!', 'lambdas']

Think/Pair/Share:
How would you produce the output list using sorted()?
How can I use a lambda function? - sorted()

```python
>>> lst = ['lambdas', 'are', 'so', 'cool!']

>>> sorted(lst, key=len)

['so', 'are', 'cool!', 'lambdas']
```
How can I use a lambda function? - min()/max()

- `min(iterable, key)`
  - if you just care about min/max, less costly than sorting a list
    - faster!
  - key function transforms each element before comparing
How can I use a lambda function? - min()/max()

- Say you have a list of tuples containing ints. You want to find the tuple whose ints add up to the greatest value.

  in: [(23, 4, 5), (9, 1, 3), (-27, 3, 300)]
  out: (-27, 3, 300)
How can I use a lambda function? - min() / max()

```python
>>> nums = [(23, 4, 5), (9, 1, 3), (-27, 3, 300)]

>>> max(nums, key=lambda elem: elem[0] + elem[1] + elem[2])
(-27, 3, 300)

>>> max(nums, key=sum)
(-27, 3, 300)
```
New Function: `sum()`

- `sum(iterable)`
- Returns the sum of the elements contained in a list, dict, or tuple
How can I use a lambda function? - min() / max()

- Back to the zoo-ture! We want to find our hungriest and least hungry animals.
  - Find the animal that eats the fewest times per day and the animal that eats the most times per day.

Think/Pair/Share: How would you find the animals with min/max feedings?
When to use lambdas

- `map()`, `filter()`  
  - actually not used that frequently
- `sorted()`
- `min()`, `max()`
How can we visualize our data?
Matplotlib

● A library for creating plots
  ○ especially useful inside of Jupyter notebooks

● To install:

```bash
$ python3 -m pip install matplotlib
```
Using Matplotlib

```python
import matplotlib.pyplot as plt

# x = list of x vals, y = list of y vals
plt.plot(x, y)  # line or scatter plot
plt.scatter(x, y)  # scatter plot
plt.title(text)  # adds a title
plt.show()  # display
```
Using Matplotlib

- There are many, many more features!
- You read the docs [here](#).
  - Here’s a useful [tutorial](#)!
Jupyter Notebook: Investigating California Air Quality
What’s next?