UC Berkeley EECS
Lecturer
Michael Ball

Computational Structures in Data Science

Lambda

Announcements

- Maps Project
  - October 1 due Oct 15
  - Checkpoint 1 week 1
  - Focus: Higher Order Functions, some Abstract Data Types (next week!)
- Midterm: Oct 26, 7 – 9pm
  - Remote, proctored option
  - Details for signups next week or so.

Learning Objectives

- Lambda are anonymous functions, which use expressions
  - We don't use return, they always return the value.
  - They are typically short and concise
  - They don't have an "intrinsic" name when using an environment diagram.

lambda

- Function expression
  - "anonymous" function creation
  - Expression, not a statement, no return or any other statement

lambda <arg or arg_tuple> : <expression using args>

```
add_one = lambda v : v + 1
```

```
def add_one(v):
    return v + 1
```

Lambdas

```
>>> def inc_maker(i):
...     return lambda x: x+i
...
>>> inc_maker(3)
<function inc_maker.<locals>.<lambda> at 0x10073c510>

>>> inc_maker(3)(4)
7
>>> map(inc_maker(3), [1,2,3,4])
<map object at 0x1020950b8>

>>> list(map(inc_maker(3), [1,2,3,4]))
[4, 5, 6, 7]
>>>
```

Lambda with HOFs

- **A function that returns (makes) a function**

```
def leq_maker(c):
    return lambda val: val <= c
```

```
>>> leq_maker(3)
<function leq_maker.<locals>.<lambda> at 0x1019d8c80>

>>> leq_maker(3)(4)
False

>>> filter(leq_maker(3), [0,1,2,3,4,5,6,7])
[0, 1, 2, 3]
```

Lambda Examples

```
>>> sorted([1,2,3,4,5], key = lambda x: x)
    [1, 2, 3, 4, 5]

>>> sorted([1,2,3,4,5], key = lambda x: -x)
    [5, 4, 3, 2, 1]

>>> sorted([(2, "hi"), (1, "how"), (5, "goes"), (7, "I")],
           key = lambda x:x[0])
[(1, 'how'), (2, 'hi'), (5, 'goes'), (7, 'I')]

>>> sorted([(2, "hi"), (1, "how"), (5, "goes"), (7, "I")],
         key = lambda x:x[1])
    [(7, 'I'), (5, 'goes'), (2, 'hi'), (1, 'how')]

>>> sorted([(2,"hi"),(1,"how"),(5,"goes"),(7,"I")],
         key =lambda x: len(x[1]))
    [(7, 'I'), (2, 'hi'), (1, 'how'), (5, 'goes')]
```

http://cs88-website.github.io/assets/slides/adt/mersort.py

UC Berkeley EECS
Lecturer
Michael Ball

Computational Structures in Data Science

Dictionaries

Learning Objectives

- Dictionaries are a new type in Python
- Lists let us index a value by a number, or position.
- Dictionaries let us index data by other kinds of data.

Dictionaries

- Constructors:
  - »`dict( <list of 2-tuples> )`
  - »`dict( <key>=<val>, ...) # like kwargs`
  - »`{ <key exp>:<val exp>, … }`
  - »`{ <key>:<val> for <iteration expression> }`
    - `>>> {x:y for x,y in zip(["a","b"],[1,2])}`
    - `{'a': 1, 'b': 2}`
- Selectors: `<dict> [ <key> ]`
  - »<dict>.keys(), .items(), .values()
  - »<dict>.get(key [, default])
- Operations:
  - » Key in, not in, len, min, max
  - » <dict>[<key>] = <val>

Dictionary Example

```
In [1]: text = "Once upon a time"
        d = {word : len(word) for word in text.split()}
        d
Out[1]: {'Once': 4, 'a': 1, 'time': 4, 'upon': 4}

In [2]: d['Once']
Out[2]: 4

In [3]: d.items()
Out[3]: [('a', 1), ('time', 4), ('upon', 4), ('Once', 4)]

In [4]: for (k,v) in d.items():
            print(k,"=>",v)
        ('a', '=>', 1)
        ('time', '=>', 4)
        ('upon', '=>', 4)
        ('Once', '=>', 4)

In [5]: d.keys()
Out[5]: ['a', 'time', 'upon', 'Once']

In [6]: d.values()
Out[6]: [1, 4, 4, 4]
```

Dictionary Example

```
In [1]: text = "Once upon a time"
        d = {word : len(word) for word in text.split()}
        d
Out[1]: {'Once': 4, 'a': 1, 'time': 4, 'upon': 4}

In [2]: d['Once']
Out[2]: 4

In [3]: d.items()
Out[3]: [('a', 1), ('time', 4), ('upon', 4), ('Once', 4)]

In [4]: for (k,v) in d.items():
            print(k,"=>",v)
        ('a', '=>', 1)
        ('time', '=>', 4)
        ('upon', '=>', 4)
        ('Once', '=>', 4)

In [5]: d.keys()
Out[5]: ['a', 'time', 'upon', 'Once']

In [6]: d.values()
Out[6]: [1, 4, 4, 4]
```