Python Basics!

dictionaries, mutable arguments

CS101 Lecture #11

2016-09-28

Administrivia

Administrivia

- Homework #5 is due Friday Sep. 30.

Administrivia

- Homework #5 is due Friday Sep. 30.
- Midterm #1 will be Monday Oct. 3. (7 p.m.)

Administrivia

- Homework #5 is due Friday Sep. 30.
- Midterm #1 will be Monday Oct. 3. (7 p.m.)
  No class on Monday,
  Labs WILL be held all week.
  Contact `cs101admin@cs.illinois.edu` for conflict exam.

Administrivia

AYA AYB AYC AYD AYE	Gregory Hall 112
AYF AYG AYH AYI	Wohlers Hall 141
AYJ AYK AYL	Main Library 66
AYM AYN AYO	Siebel Center 1404
AYP AYQ AYR	David Kinley Hall 114

Midterm Instructions

- 30 multiple-choice questions
- 60 minutes

Midterm Instructions

- 30 multiple-choice questions
- 60 minutes
- Requires NetID and University I-Card.

Midterm Instructions

- 30 multiple-choice questions
- 60 minutes
- Requires NetID and University I-Card.
- Exams are unique—omitting the exam code will dock one letter grade (10%).

Warmup Quiz

Question #1

```
a = [ [1,2,3], [4,5,6], [7,8,9] ]
```

How would you refer to the value 8?

A `a[2][3]`
B `a[1][2]`
C `a[2,3]`
D `a[2][1]`

Question #2

```
x = [ 'a', 'b' ]
y = [ 'c', 'd' ]
def add_it( x,y ):
    y.append( x )
add_it( y,x )
```

What is the final value of x?

A `[ 'a', 'b', 'c', 'd' ]`

B `[ 'a', 'b' ]`

C `[ 'a', 'b', [ 'c', 'd' ] ]`

D `None`

Question #3

```
x = [ 'a', 'b' ]
y = [ 'c', 'd' ]
def add_it( x,y ):
    y.append( x )
add_it( y,x )
```

What is the final value of x?

A `[ 'a', 'b', 'c', 'd' ]`

B `[ 'a', 'b' ]`

C `[ 'a', 'b', [ 'c', 'd' ] ]`

D `None`

Question #1 (Worked)

```
a = 1
def fun(c,b):
    return c + b
a = fun( a,a ) + a
```

Dictionaries

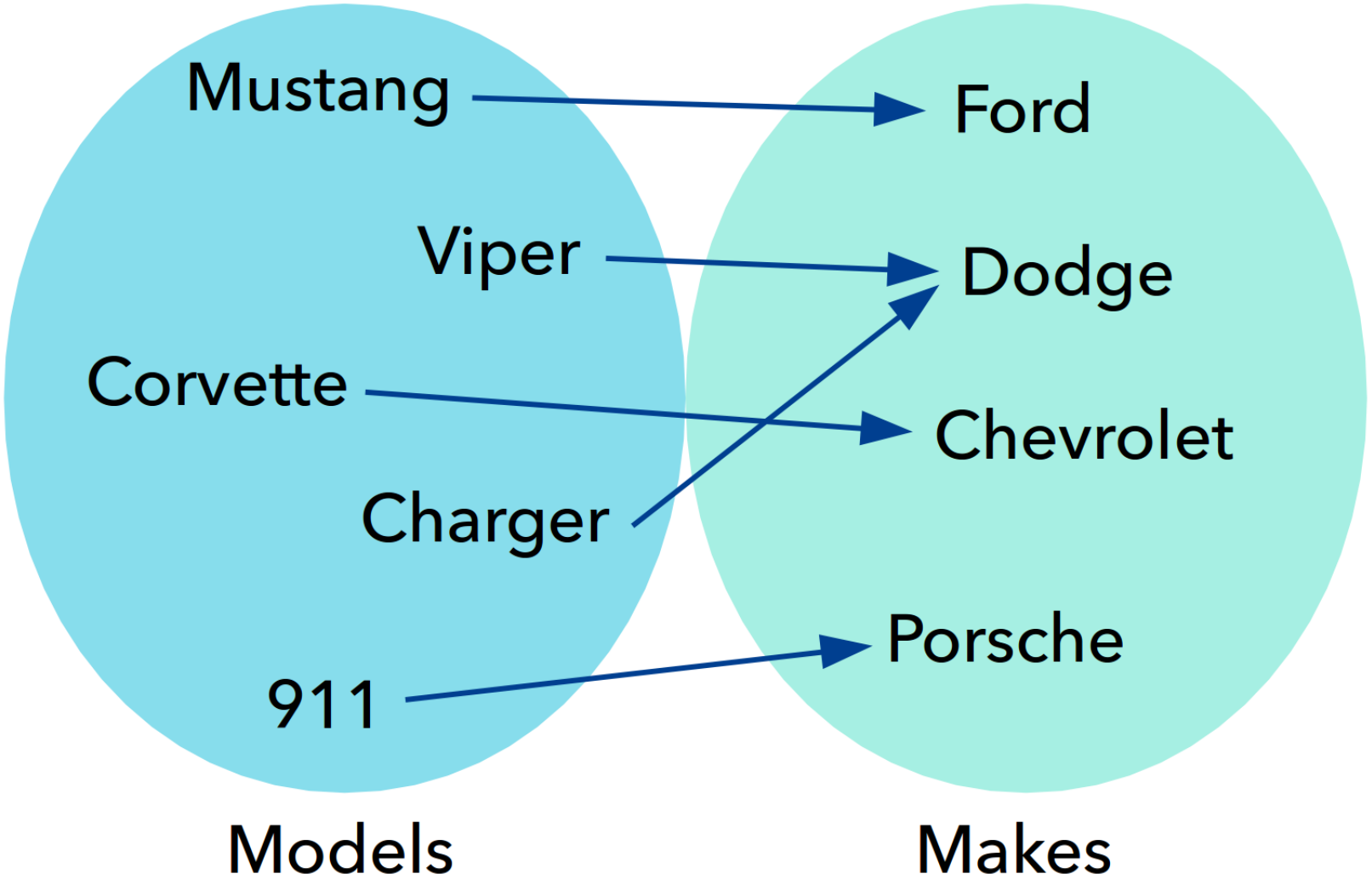
`dict` data type

- How do we index a `list`?

`dict` data type

- How do we index a `list`?
- `lists` and `tuples` are ordered.
- What else may make sense–how else could you organize data?

Example


Mustang
Viper
Corvette
Charger
911
Ford
Dodge
Chevrolet
Porsche
Models
Makes

`dict` data type

- The `dict` indexes data by any value (unordered).

`dict` data type

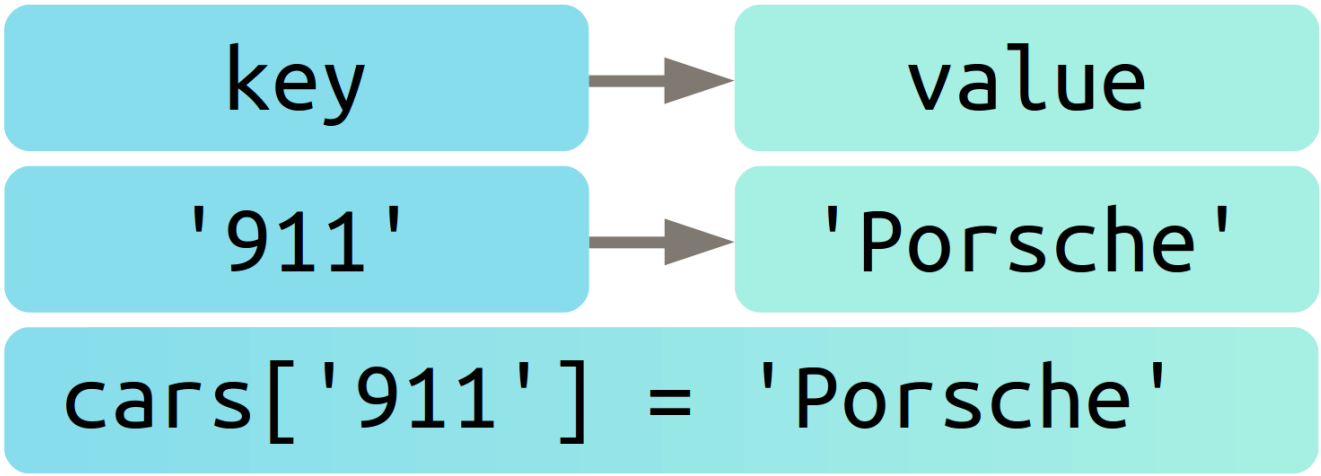
- The `dict` indexes data by any value (unordered).
- Easy to think of as dictionary, but can use lots besides strings.

`dict` data type

- The `dict` indexes data by any value (unordered).
- Easy to think of as dictionary, but can use lots besides strings.
- This container maps keys to values.

dict data type

- The **dict** indexes data by any **value** (unordered).
- Easy to think of as dictionary, but can use lots besides strings.
- This container maps keys to values.



dict data type

```
cars = {}
cars[ 'Mustang' ]  = 'Ford'
cars[ 'Viper' ]    = 'Dodge'
cars[ 'Corvette' ] = 'Chevrolet'
cars[ 'Charger' ]  = 'Dodge'
cars[ '911' ]      = 'Porsche'
```

`dict` literals

- We create a `dict` as follows:
  - opening brace `{`
  - `key : value` pairs, separated by commas
  - closing brace `}`

`dict` literals

- We create a `dict` as follows:
  - opening brace `{`
  - `key : value` pairs, separated by commas
  - closing brace `}`

```
model = {
   'Civic': 'Honda',
   'Mustang': 'Ford',
   'Model S': 'Tesla',
   'Model T': 'Ford'
}
```

dict operations & methods

```
d = { 'one':1, 'two':2, 'three':3 }
print( d['one'] )
d[ 'four' ] = 4
del d[ 'four' ]
'five' in d
for key in d:  # no guarantee on order
    print( key, d[key] )
d.keys()
d.values()
```

Example

```
d = { 'a':2, 'c':3, 'b':1 }
x = d[ 'a' ] + d[ 'c' ]
```

What is the final value of x?

A 4
B 'ac'
C '5'
D 5

Example

```
d = { }
words = [ 'red', 'orange', 'yellow' ]
for word in words:
    d[ word ] = words.index( word )
```

What is the final value of d?

A `{ 'red':3, 'orange':6, 'yellow':6 }`

B `{ 'red':0, 'orange':2, 'yellow':2 }`

C `None`

D `{'orange': 1, 'red': 0, 'yellow': 2}`

`dict` applications

- Dictionaries can encode/decode data, or translate from one representation to another.

dict applications

- Dictionaries can encode/decode data, or translate from one representation to another.

```
x = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
y = 'BCDEFGHIJKLMNOPQRSTUVWXYZA'
e = { }
for i in range( len(x) ):
    e[ x[i] ] = y[i]
encoded = ”
for c in 'HELLO':
    encoded += e[c]
```

- How would you reverse (decode) this?

dict applications

```
x = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
y = 'BCDEFGHIJKLMNOPQRSTUVWXYZA'
d = { }
for i in range( len(x) ):
    d [y[i] ] = x[i]
decoded = ”
for c in encoded:
    decoded += d[c]
```

Exercise

- Encode all of the words in a file using a Caesar cipher.
- Decode all of the words in the file.

dict applications

- Dictionaries can also function as accumulators.

```
x = 'ABBACAB'
d = { }
for c in x:
    if c not in d:
        d[c] = 0
        d[c] += 1
```

- How would you reverse (decode) this?

Exercise

- Count category frequencies in Jeopardy questions.
- Count bigram frequencies in Jeopardy clues.

dict applications

- We can link data based on a common field.

```
zipcode = { 'Bill': 60644,
            'Jill': 41073,
            'Tony': 63103 }
city = { 60644: 'Chicago',
         41073: 'Cincinnati',
         63103: 'St. Louis' }
for name in zipcode:
    print( name,city[zipcode[name]] )
```

Mutable Arguments

Exercise: mutability

```
x = [ 3,2,1 ]
y = x
y.sort()
x.append( 0 )
```

What is the final value of x?

A `[ 3,2,1 ]`

B `[ 1,2,3 ]`

C `[ 1,2,3,0 ]`

D `[ 0,1,2,3 ]`

Mutable arguments

- Mutability causes `lists` to work differently in functions.

Mutable arguments

- Mutability causes `lists` to work differently in functions.
- `lists` used as arguments can be changed by the function.

Mutable arguments

- Mutability causes `lists` to work differently in functions.
- `lists` used as arguments can be changed by the function.
- This is very useful!

Mutable arguments

- Mutability causes `lists` to work differently in functions.
- `lists` used as arguments can be changed by the function.
- This is very useful!

```
def fun(q):
    q.append(3)

a = [ ]
for i in range(3):
    fun(a)
print(a)
```

Mutable arguments

```
def readfile(fname,a):
    for line in open(fname):
        a.append(line)

all_lines = []
readfile( 'file1.txt', all_lines )
readfile( 'file2.txt', all_lines )
```

Mutable arguments

```
def readfile(fname,a):
    for line in open(fname):
        a.append(line)

all_lines = []
for f in open("filenames.txt"):
    readfile(f,all_lines)
```

Copying mutable values

- What if we want a copy of a `list` (not an alias)?

Copying mutable values

- What if we want a copy of a `list` (not an alias)?
- Slice everything!

Copying mutable values

- What if we want a copy of a `list` (not an alias)?
- Slice everything!

```
x = [ 3,2,1 ]
y = x[ : ]
y.sort()
print( x )
```

Copying mutable values

```
x = [ 1,2,3 ]
y = x[ : ]
y.append( 4 )
print( x == y )
```

String/List Methods

string.split method

- `split` returns a `list`.

`string.split` method

- `split` returns a `list`.
- Takes a single string argument, the delimiter.

string.split method

- `split` returns a `list`.
- Takes a single string argument, the delimiter.

```
name = 'Oliver Wendell Holmes'
names = name.split(' ')
print(m[-1])
```

Example

```
x = 'A+B+C'
y = x.split('+')
```

What is the final value of y?

A `'ABC'`

B `[ 'A','B','C' ]`

C `'A','B','C'`

D `None`

Example

```
x = 'A+B+C'
y = x.split('-')
```

What is the final value of y?

A `'A+B+C'`

B `[ 'A+B+C' ]`

C `( 'A+B+C' )`

D `None`

string.join method

- `join` returns a `str`.

`string.join` method

- `join` returns a `str`.
- Takes a single `list` argument.

`string.join` method

- `join` returns a `str`.
- Takes a single `list` argument.
- Returns the `list` elements joined as a string.

`string.join` method

- `join` returns a `str`.
- Takes a single `list` argument.
- Returns the `list` elements joined as a string.

```
names = [ ”Geoffrey”, ”Richard”, ”Aloysius”, ”Jo
’,’.join(names)      # note the odd syntax!
```

Example

```
a = [ 'X', 'A', 'G' ]
b = a[:]
a.sort()
x = ','.join(b)
```

What is the final value of x?

A `'XAG'`

B `[ 'X,A,G' ]`

C `'A,G,X'`

D `',A,G,X,'`

Reminders

Reminders

- Homework #5 is due Friday Sep. 30.
- Midterm #1 will be Monday Oct. 3. (7 p.m.)
  No class on Monday,
  Labs WILL be held all week.
  Contact `cs101admin@cs.illinois.edu` for conflict exam.