CS101: Intro to Computing
Summer 2016

Week 4
DICTIONARIES
Types we’ve learned

• Lists and tuples are ordered
• Lists and tuples are indexed using a integers
• It’s natural to associate data with an identifier:
  – Person → birthday, gender, parents
  – Country → flag, median income
Mapping

X
---
1
2
3
---
Y
D
B
C
A
Dictionaries

• Represents an **unordered** collection of items or elements

• A **container** type
  – Contains other values of **any type**
  – **NOTE**: elements don’t have to be the same type

• Can be indexed with **any type**

• Map **keys** to **values**
Dictionary literals

- We create an *dictionary* by typing:
  1. an open curly bracket `{`
  2. a key, a colon, and its associated value
  3. key:value pairs separated by commas
  4. a closing curly bracket `}`
Dictionary

model={"Civic":"Honda",
       "Mustang":"Ford",
       "Corvette":"Chevy",
       "Accord":"Honda",
       "Accent":"Hyundai"}
Dictionary Operations

d={'one':1,"two":2,"three":3}
print(d['one'])
d['four']=4
del d['four']
"five" in d
for key in d:
    print(key,d[key])
d={"a":2,"c":3,"b":1}
x=d["a"]+d["c"]

What is the final value of x?

a) 3  
b) "ac"  
c) 4  
d) 5
d = {}
for i, j in enumerate("ABC"):  
    d[j] = i

What is the final value of d?

a) {"A":0,"B":1,"C":2}
b) {0:"A",1:"B",2:"C"}
c) {"A":1,"B":2,"C":3}
d) {1:"A",2:"B",3:"C"}
d = {}
for i, j in zip("WAT", "NEY"):  
    d[(i, j)] = j

Which of these expressions evaluates to "E"?

a) d["A"]
b) d["E"]
c) d["A", "E"]
d) d["EA"]
s = "ABACAB"
d = {}
for a, b in enumerate(s):
    if b not in d:
        d[b] = 0
    d[b] += a

a) d = {1: "A", 3: "C", 2: "B"}
b) d = {"A": 3, "B": 2, "C": 1}
c) d = {"A": 6, "B": 6, "C": 3}
d) d = {"A": 0, "B": 0, "C": 0}
s="ABACAB"
d={}
for a,b in enumerate(s):
    if b not in d:
        d[b]=a
    else:
        d[b]+=a
s=[4,8,15,16,23,42]
d={0:0,1:0}
for a,b in enumerate(s):
    d[a%2]+=1
x=d[1]

What is the final value of x?
a) 0
b) 1
c) 2
d) 3
USES FOR DICTIONARIES
Dictionaries to Encode

• We can use dictionaries to encode/decode data
• We can use dictionaries to translate from one representation to another
Exercise

- Encipher all of the words in a file with the Caesar cypher
- Substitute each letter with the next in the alphabet

HELLO → IFMMP
Dictionaries to Encode

```python
x = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
y = "BCDEFGHIJKLMNOPQRSTUVWXYZA"
e = {}
for i in range(len(x)):
    e[x[i]] = y[i]
encoded = ""
for c in "HELLO":
    encoded += e[c]
```
Dictionaries to Encode

```
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 with the Caesar cypher
• Decode all of the words in the file
def encypher(word):
    word=word.upper()
    x="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    y=x[1:]+x[0]
    e={}
    for a,b in zip(x,y):
        e[a]=b
    encoded=""
    for c in word:
        if c in encoded:
            encoded+=e[c]
        else:
            encoded+=c
    return encoded

for line in open("words.txt"):
    line=line.strip()
    print(encypher(line))
Dictionaries as Accumulators

• We can use dictionaries as a collection of counters for many things at once

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

• Count category frequencies in Jeopardy questions
• Count bigram frequencies in Jeopardy clues
Counting bigrams

counter={}
for word in open("words.txt"):  
    word=word.strip().upper()
    for i in range(len(word)-1):
        bi=word[i:i+2]
        if bi not in counter:
            counter[bi]=1
        else:
            counter[bi]+=1
Dictionaries to Join/Merge Data

• We can link data based on a common field

```python
zip={"Bill":60644,
    "Jim":41073,"Beth":63103}
city={60644:"Chicago",
    41073:"Cincinnati",
    63103:"St. Louis"}
for name in zip:
    print(name,city[zip[name]])
```