CS101: Intro to Computing
Fall 2015

Lecture 13

Administrivia

- Homework 10 is due today
- Midterm results coming today
- New homework coming
- Install Anaconda

https://www.continuum.io/downloads

REVIEW

```
s="WTE"
t="ANY"
u={}
for a,b in zip(s,t):
  u[a]=b
x=u["T"]
```

What is the final value of x?

a) "A"

b) "Y"

c) "T"

d) "N"

```
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?

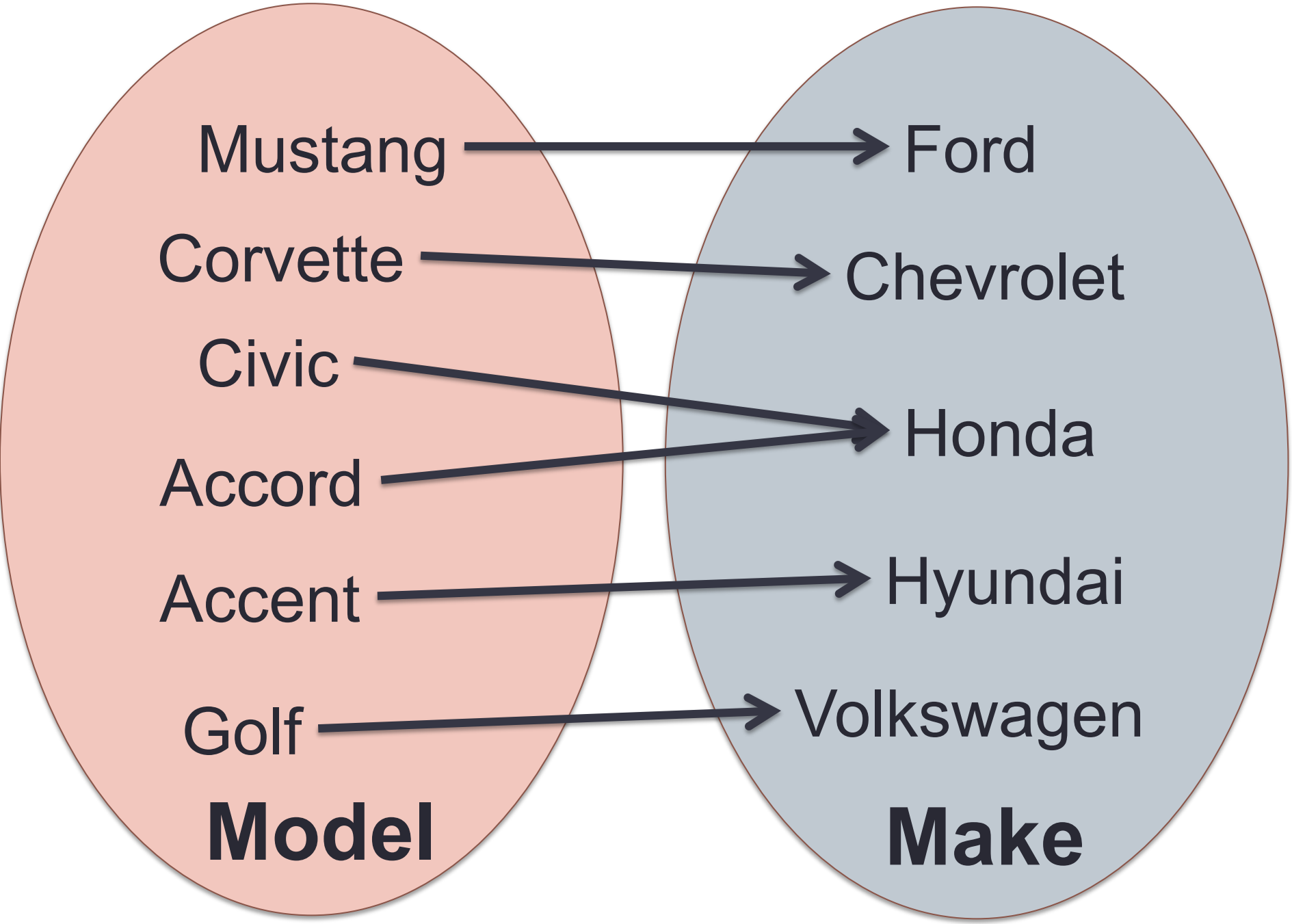
a) 0

b) 1

c) 2

d) 3

DICTIONARIES


Mustang
Corvette
Civic
Accord
Accent
Golf
Model
Ford
Chevrolet
Honda
Hyundai
Volkswagen
Make

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

```
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)
```

Exercise

- Count category frequencies in Jeopardy questions

```
category_counts={}
for line in open("jeopardy.txt"):
        if line[0]!="#":
                split=line.split("\t”)
                category=split[2]
                if category not in category_counts:
                        category_counts[category]=0
                category_counts[category]+=1

category_tuples=[]
for c in category_counts:
        n=category_counts[c]
        category_tuples.append((n,c))
category_tuples.sort()
print category_tuples
```

Dictionaries to Join/Merge Data

- We can link data based on a common field

```
zip={“Bill”:60644,
  ”Jim”:41073,”Beth”:63103}
city={60644:”Chicago”,
  41073:”Cincinnati”,
  63103:”St. Louis”}
for name in zipcode:
  print name,city[zipcode[name]]
```

MODULES

Modules

- A collection of Python specialized functions, variables, and types
- We need to ***import*** the module

```
import math
```

- Can then access things within the module using ***attribute operator***

```
math.sqrt(math.pi)
```

From

- Can choose what to import with ***from***

```
from cmath import phase
phase(1+1j)
```

- This means we don't have to type the module name all the time.
- Import multiple items with a comma

```
from cmath import phase, rect
```

```
???
exp(pi)-pi
```

What should replace the ???

a)
```
from math import pi
  import math
```

b)
```
from math import pi,exp
```

c)
```
import pi,exp
  import math
```

d)
```
import math
```

Useful Python Modules

- math, cmath
- random
- csv
- sys, os
- time, datetime
- itertools
- logging
- NumPy
- SciPy
- matplotlib

WRITING CODE

Writing readable code

- We should always strive to write code that is easy to read.
  - Our variables should have ***descriptive*** names.
  - We should also ***annotate*** our code.
- **REMEMBER**: A program is set of instructions a computer executes ***to achieve a goal***.

Commenting

- ***Comments*** are text that the interpreter ignores
- Comments help ***a person*** read a program
- The # symbol indicates a comment
  - Anything after that symbol is ignored

```
# Hello, I am a comment
```

Docstring

- A string literal that behaves like a comment
- Use triple quotes
- Especially useful after function definition

```
"""Hello, I am a docstring."""
```

```
s="ABCD"
s+="1"
#s+="2"
"""s+=3"""
```

What is the final value of x?

a) ABCD

b) ABCD1

c) ABCD12

d) ABCD123

Why comment/document?

- Allows us to ***explain*** our code to others.
- But mostly… to ourselves.
- Yes, ***ourselves.***

Documenting Modules

- Every script (.py) file you write is a module.
- Your modules should have a docstring at the beginning describing them and you.

```
"""
CS101 class demonstration
Author: Ryan Cunningham
"""
```

Documenting Functions

- Use doc string and describe what function does.
- Describe all parameters by name.
- Describe all return values.

```
def sqrt(n):
    """
    Computes square root of a number.
    n: an integer or float
    returns: the square root of n
    """
    return n**.5
```

```
category_counts={} # accumulator for counting categories
for line in open("jeopardy.txt"):
        if line[0]!="#": # ignore comment lines
                split=line.split("\t”) # split on tab
                category=split[2] #category is 3rd column
                #increment count of this category
                if category not in category_counts:
                        category_counts[category]=0
                category_counts[category]+=1

# sort category counts for display
category_tuples=[]
for c in category_counts:
        n=category_counts[c]
        category_tuples.append((n,c))
category_tuples.sort() # tuples are sorted by first item
print category_tuples
```

DEBUGGING

“My code doesn’t work.”

“My code doesn’t work.”

“My code doesn’t work.”

1. How do I know it isn’t working?
2. What do I expect it to do?
3. What is my code doing instead? Why?

Eliminate all other factors, and the one which remains must be the truth.

Errors = Clues

1. Read message and ***think*** about it.

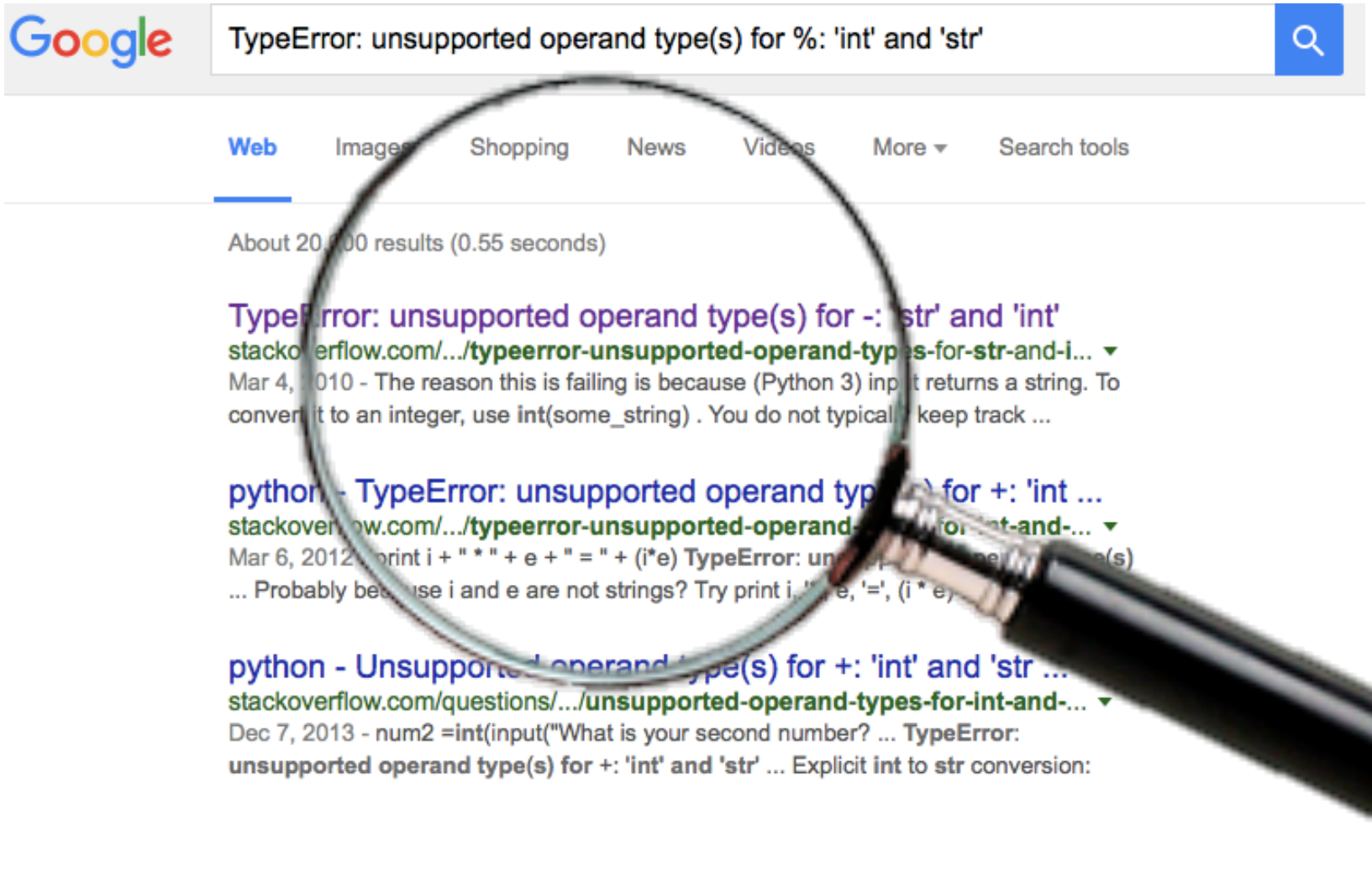
```
AttributeError: 'int' object has no attribute 'append'
```

```
IndexError: list index out of range
```

2. Google the error message

```
TypeError: unsupported operand type(s) for %: 'int' and 'str'
```

Error messages



Don’t just stare at your code!

- Make the ***computer*** work, not you!
- Add print statements (especially in loops!)

Data! Data!
Data!
I can't make
bricks without
clay.

```
# Find all words whose first two letters are
# the same
for line in open("words.txt"):
        line=line.strip()
        if len(line)>=2:
                a,b=line[1:3]
                if a==b:
                        print line
```

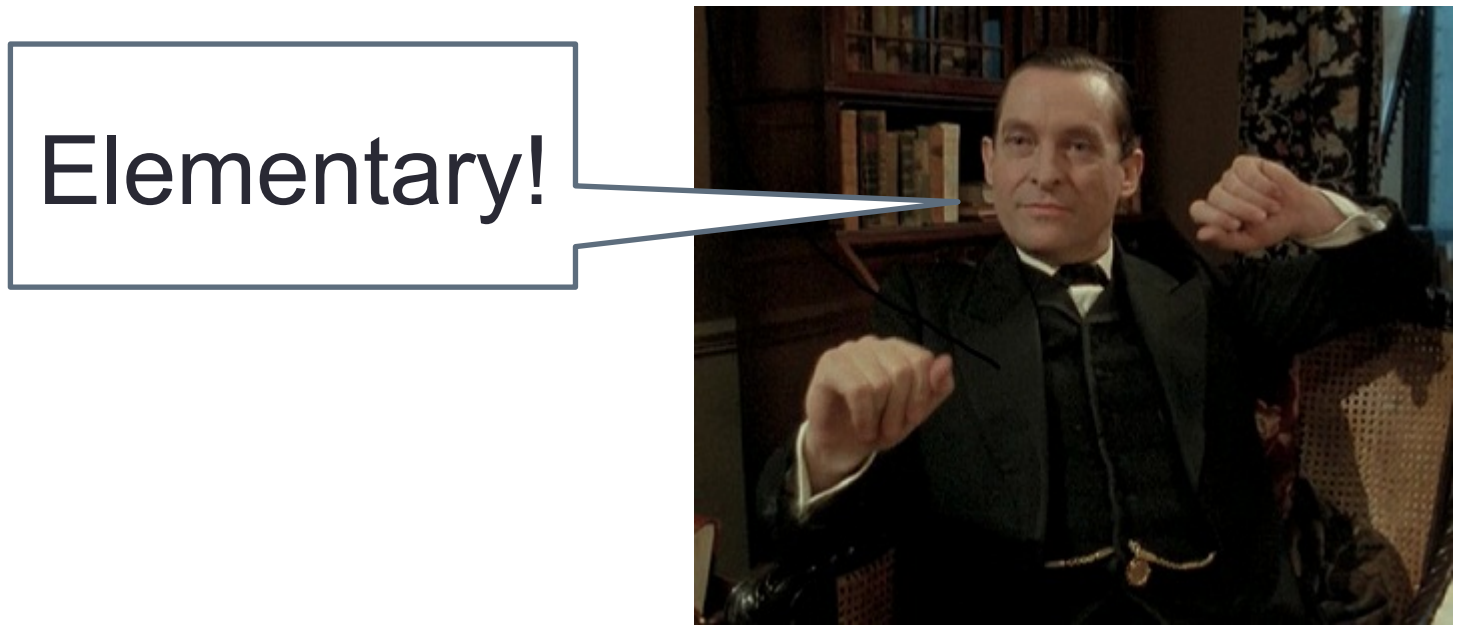
```
ValueError: need more than 1 value to unpack
```


The game
is afoot!

```
# Find all words whose first two letters are
# the same
for line in open("words.txt"):
        line=line.strip()
        if len(line)>=2:
                print line
                a,b=line[1:3]
                print a,b
                if a==b:
                        print line
```

It is a capital mistake to theorize in advance of the facts.

```
# Find all words whose first two letters are
# the same
for line in open("words.txt"):
        line=line.strip()
        if len(line)>=2:
                a,b=line[0:2]
                if a==b:
                        print line
```



Debugging Tips

1. Comment out lines to narrow down your search for the problem.
2. Add print statements.
3. Trace troublesome input through your code line by line.
4. Explain the problem to someone else.
5. Take a break.

Coding tips

1. **START EARLY!**
2. Break the problem down and write functions.
3. Document your functions ***before*** writing them.
4. Test every function (even every ***line***) you write to make sure it works (write then execute, write then execute).

Ultimate Coding Secret

- If you're confused, frustrated, and can't make progress…

Always approach a case with an absolutely blank mind.

- … start over.

You know my methods. Apply them.