ECS 89

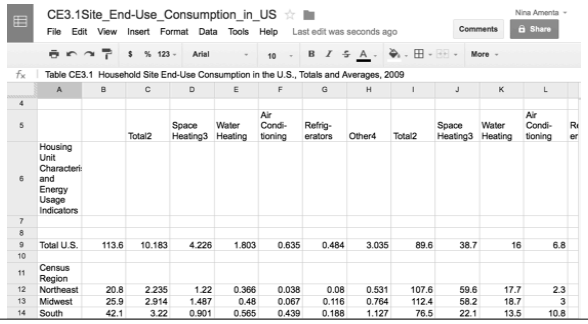
4/2 and 4/4

Announcements

- Lecture tomorrow at 12, here (regular discussion time)
- Discussion Fri at 10, here (regular lecture time)

First Assignment

- Write a Python program that helps the user turn this dataset into…

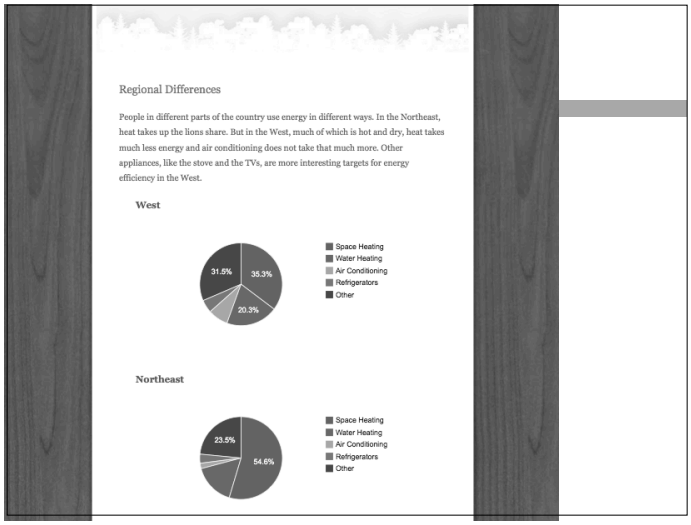


CE3.1Site_End-Use_Consumption_in_US

File Edit View Insert Format Data Tools Help Last edit was seconds ago — Comments — Share — Nina Amenta

Table CE3.1 Household Site End-Use Consumption in the U.S., Totals and Averages, 2009

	A	B	C	D	E	F	G	H	I	J	K	L
4												
5			Total2	Space Heating3	Water Heating	Air Condi-tioning	Refrig-erators	Other4	Total2	Space Heating3	Water Heating	Air Condi-tioning
6	Housing Unit Characteri and Energy Usage Indicators											
7												
8												
9	Total U.S.	113.6	10.183	4.226	1.803	0.635	0.484	3.035	89.6	38.7	16	6.8
10												
11	Census Region											
12	Northeast	20.8	2.235	1.22	0.366	0.038	0.08	0.531	107.6	59.6	17.7	2.3
13	Midwest	25.9	2.914	1.487	0.48	0.067	0.116	0.764	112.4	58.2	18.7	3
14	South	42.1	3.22	0.901	0.565	0.439	0.188	1.127	76.5	22.1	13.5	10.8



Regional Differences

People in different parts of the country use energy in different ways. In the Northeast, heat takes up the lions share. But in the West, much of which is hot and dry, heat takes much less energy and air conditioning does not take that much more. Other appliances, like the stove and the TVs, are more interesting targets for energy efficiency in the West.

West

Northeast

Details

- Object-oriented program
- Program uses a bunch of datatypes that come from a module
- Your job is to write the eiadata module

Objects

- Let's make an object that represents a deck of cards.

Objects

- Let's make an object that represents a deck of cards.

```
class Deck:
def __init__(self):
        self.cards = []
        for num in range(1,14):
            for suit in ["h","s","d","c"]:
                card = str(num)+suit
                self.cards.append(card)
```

Classes and __init__

- The class defines a data type, eg. class Deck
- Usually start with a capital letter
- A piece of data – an object – can have this class
- To make an object of this type, use the initialization method (aka a constructor)

```
D = Deck()
```

D is a variable containing a Deck

- There might be lots of objects of the same class.

Methods

- Methods are functions that belong to a class
- Eg. string methods like split() work on strings, list methods like append() work on lists...
- You can make up methods that work on the data in your class

attributes

- A class includes some code, maybe a lot of code.
- Usually a class also contains some data (in this example, the list of cards). The data are called attributes.
- Access the attributes with the dot:

```
print  D.cards
```

- Not all the variables used in a class are attributes.

self

- The code inside the class has no idea what the object is called. There might be lots of objects of this class.
- The word ***self*** refers to the object ***itself***.
- To access the object's own data using the code in the class, use self instead of the variable name.

```
self.cards.append(card)
```

printing

- Printing out the class just gives nonsense
- Attributes might be lists or more complicated data structures
- Nice to have things print out pretty

```
def __str__(self):
        s = ""
        for card in self.cards:
            s += card+" "
        return s
```

Local variables

- s is local to the function __str__
  - Invisible outside the class
  - Invisible to other functions in the class
- self.cards is global to the whole class
  - Visible to other functions in the class
  - Visible outside the class, with variable containing an object replacing "self"

Classes in their own module

- Tidier to put the classes into their own module.
- We could use these cards in a poker program, or in a bridge program, or for a magic trick…
- Only thing the main program has to know is the classes, attributes, and methods, not how they are implemented.

Project 1 program structure

- Project 1 is a common programming problem:
  - get data in,
  - select, reformat, compute…
  - put data out
- Crucial design choice: how to store data within the program. Ask yourself two questions:
  - What is the data?
  - What are the outputs going to be?

Name decoder data

- .csv file – data fields separated by commas
- First field – name
- Second field – meaning
- In EIA data for Project 1, you have 15 or so fields.

- Output?

Name decoder data

- .csv file – data fields separated by commas
- First field – name
- Second field – meaning
- In EIA data for Project 1, you have lots more fields.

- Output?
- Need to report meaning when given name

- SO…DICTIONARY!

Reading a file

```
def __init__(self):
        f = open("names.csv","rU")
            for line in f:
                    words = line.split(",")
                    print words
```

Using a dictionary

```
def __init__(self):
        self.nameD = {}
        f = open("names.csv","rU")
        for line in f:
                words = line.split(",")
                name = words[0]
                meaning = words[1]
                self.nameD[name] = meaning
```

Get data out of dictionary

```
def define(self,name):
        if name in self.nameD:
                return self.nameD[name]
        else:
                return "nothing that I know of"
```