Chapter 1

Examples and Exercises from Think Stats, 2nd Edition

http://thinkstats2.com

In [3]:
```
from os.path import basename, exists


def download(url):
    filename = basename(url)
    if not exists(filename):
        from urllib.request import urlretrieve

        local, _ = urlretrieve(url, filename)
        print("Downloaded " + local)


download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkstats2.py
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkplot.py")
```

In [4]:
```
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/nsfg.py")

download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/2002FemPreg.dc
download(
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/2002FemPreg.dat.gz"
)
```

Examples from Chapter 1

Read NSFG data into a Pandas DataFrame.

In [5]:
```
import nsfg
```

In [6]:
```
preg = nsfg.ReadFemPreg()
preg.head()
```

Out[6]:

	caseid	pregordr	howpreg_n	howpreg_p	moscurrp	nowprgdk	pregend1	pregend2	nbrnaliv
0	1	1	NaN	NaN	NaN	NaN	6.0	NaN	1.0
1	1	2	NaN	NaN	NaN	NaN	6.0	NaN	1.0
2	2	1	NaN	NaN	NaN	NaN	5.0	NaN	3.0
3	2	2	NaN	NaN	NaN	NaN	6.0	NaN	1.0
4	2	3	NaN	NaN	NaN	NaN	6.0	NaN	1.0

5 rows × 244 columns

Print the column names.

In [7]:
```
preg.columns
```

Out[7]:
```
Index(['caseid', 'pregordr', 'howpreg_n', 'howpreg_p', 'moscurrp', 'nowprgdk',
       'pregend1', 'pregend2', 'nbrnaliv', 'multbrth',
       ...
       'laborfor_i', 'religion_i', 'metro_i', 'basewgt', 'adj_mod_basewgt',
       'finalwgt', 'secu_p', 'sest', 'cmintvw', 'totalwgt_lb'],
      dtype='object', length=244)
```

Select a single column name.

In [8]:
```
preg.columns[1]
```

Out[8]:
```
'pregordr'
```

Select a column and check what type it is.

In [9]:
```
pregordr = preg['pregordr']
type(pregordr)
```

Out[9]:
```
pandas.core.series.Series
```

Print a column.

In [10]:
```
pregordr
```

Out[10]:
```
0        1
1        2
2        1
3        2
4        3
        ..
13588    1
13589    2
13590    3
13591    4
13592    5
Name: pregordr, Length: 13593, dtype: int64
```

Select a single element from a column.

In [11]:
```
pregordr[0]
```

Out[11]:
```
1
```

Select a slice from a column.

In [12]:
```
pregordr[2:5]
```

Out[12]:
```
2    1
3    2
4    3
Name: pregordr, dtype: int64
```

Select a column using dot notation.

In [13]:
```
pregordr = preg.pregordr
```

Count the number of times each value occurs.

In [15]:
```
preg.outcome.value_counts().sort_index()
```

Out[15]:
```
1    9148
2    1862
3     120
4    1921
5     190
6     352
Name: outcome, dtype: int64
```

Check the values of another variable.

In [16]:
```
preg.birthwgt_lb.value_counts().sort_index()
```

Out[16]:
```
0.0        8
1.0       40
2.0       53
3.0       98
4.0      229
5.0      697
6.0     2223
7.0     3049
8.0     1889
9.0      623
10.0     132
11.0      26
12.0      10
13.0       3
14.0       3
15.0       1
Name: birthwgt_lb, dtype: int64
```

Make a dictionary that maps from each respondent's `caseid` to a list of indices into the

pregnancy `DataFrame`. Use it to select the pregnancy outcomes for a single respondent.

In [17]:
```
caseid = 10229
preg_map = nsfg.MakePregMap(preg)
indices = preg_map[caseid]
preg.outcome[indices].values
```

Out[17]:
```
array([4, 4, 4, 4, 4, 4, 1], dtype=int64)
```

Exercises

Select the `birthord` column, print the value counts, and compare to results published in the codebook

In [18]:
```
preg.birthord.value_counts().sort_index()
```

Out[18]:
```
1.0     4413
2.0     2874
3.0     1234
4.0      421
5.0      126
6.0       50
7.0       20
8.0        7
9.0        2
10.0       1
Name: birthord, dtype: int64
```

We can also use `isnull` to count the number of nans.

In [19]:
```
preg.birthord.isnull().sum()
```

Out[19]:
```
4445
```

Select the `prglngth` column, print the value counts, and compare to results published in the codebook

In [20]:
```
preg.prglngth.value_counts().sort_index()
```

Out[20]:
```
0       15
1        9
2       78
3      151
4      412
5      181
6      543
7      175
8      409
9      594
10     137
11     202
12     170
13     446
14      29
15      39
16      44
17     253
18      17
19      34
20      18
21      37
22     147
23      12
24      31
25      15
26     117
27       8
28      38
29      23
30     198
31      29
32     122
33      50
34      60
35     357
36     329
37     457
38     609
39    4744
40    1120
41     591
42     328
43     148
44      46
45      10
46       1
47       1
48       7
50       2
Name: prglngth, dtype: int64
```

To compute the mean of a column, you can invoke the `mean` method on a Series. For example, here is the mean birthweight in pounds:

In [21]:
```
preg.totalwgt_lb.mean()
```

Out[21]:
```
7.265628457623368
```

Create a new column named `totalwgt_kg` that contains birth weight in kilograms. Compute its mean. Remember that when you create a new column, you have to use dictionary syntax, not dot notation.

In [22]:
```
preg["totalwgt_kg"] = preg.totalwgt_lb/2.2
preg.totalwgt_kg.mean()
```

Out[22]:
```
3.302558389828807
```

`nsfg.py` also provides `ReadFemResp`, which reads the female respondents file and returns a `DataFrame`:

In [23]:
```
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/2002FemResp.dc
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/2002FemResp.da
```

In [24]:
```
resp = nsfg.ReadFemResp()
```

`DataFrame` provides a method `head` that displays the first five rows:

In [25]:
```
resp.head()
```

Out[25]:

	caseid	rscrinf	rdormres	rostscrn	rscreenhisp	rscreenrace	age_a	age_r	cmbirth	agescrn	.
0	2298	1	5	5	1	5.0	27	27	902	27	
1	5012	1	5	1	5	5.0	42	42	718	42	
2	11586	1	5	1	5	5.0	43	43	708	43	
3	6794	5	5	4	1	5.0	15	15	1042	15	
4	616	1	5	4	1	5.0	20	20	991	20	

5 rows × 3087 columns

Select the `age_r` column from `resp` and print the value counts. How old are the youngest and oldest respondents?

In [26]:
```
resp.age_r.value_counts().sort_index()
```

Out[26]:
```
15    217
16    223
17    234
18    235
19    241
20    258
21    267
22    287
23    282
24    269
25    267
26    260
27    255
28    252
29    262
30    292
31    278
32    273
33    257
34    255
35    262
36    266
37    271
38    256
39    215
40    256
41    250
42    215
43    253
44    235
Name: age_r, dtype: int64
```

In [27]:
```
'Youngest : ',  resp.age_r.min()
```

Out[27]:
```
('Youngest : ', 15)
```

In [66]:
```
'Oldest : ' , resp.age_r.max()
```

Out[66]:
```
('Oldest : ', 44)
```

We can use the `caseid` to match up rows from `resp` and `preg`. For example, we can select the row from `resp` for `caseid` 2298 like this:

In [32]:
```
resp[resp.caseid==2298]
```

Out[32]:

	caseid	rscrinf	rdormres	rostscrn	rscreenhisp	rscreenrace	age_a	age_r	cmbirth	agescrn	.
0	2298	1	5	5	1	5.0	27	27	902	27	

1 rows × 3087 columns

And we can get the corresponding rows from `preg` like this:

In [29]:
```
preg[preg.caseid==2298]
```

Out[29]:

	caseid	pregordr	howpreg_n	howpreg_p	moscurrp	nowprgdk	pregend1	pregend2	nbr
2610	2298	1	NaN	NaN	NaN	NaN	6.0	NaN	
2611	2298	2	NaN	NaN	NaN	NaN	6.0	NaN	
2612	2298	3	NaN	NaN	NaN	NaN	6.0	NaN	
2613	2298	4	NaN	NaN	NaN	NaN	6.0	NaN	

4 rows × 245 columns

How old is the respondent with `caseid` 1?

In [30]:
```
resp[resp.caseid==1].age_r
```

Out[30]:
```
1069    44
Name: age_r, dtype: int64
```

What are the pregnancy lengths for the respondent with `caseid` 2298?

In [31]:
```
preg[preg.caseid==2298].prglngth
```

Out[31]:
```
2610    40
2611    36
2612    30
2613    40
Name: prglngth, dtype: int64
```

What was the birthweight of the first baby born to the respondent with `caseid` 5012?

In [34]:
```
preg[preg.caseid==5012].pregordr.apply(lambda x:  preg[preg.caseid==5012].totalwgt_
```

Out[34]:

	5515
5515	6.0