In [27]:
```python
import sys
import string
import logging


logging.basicConfig(filename="logger.log", format='%(message)s',
                    level=logging.INFO, filemode='w')

def mapper():
    """
    The input to this mapper will be the final Subway-MTA dataset, the same as
    in the previous exercise.  You can check out the csv and its structure below:
    https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/turnstile_data_master_with_weather.csv

    For each line of input, the mapper output should PRINT (not return) the UNIT as
    the key, the number of ENTRIESn_hourly as the value, and separate the key and
    the value by a tab. For example: 'R002\t105105.0'

    Since you are printing the output of your program, printing a debug
    statement will interfere with the operation of the grader. Instead,
    use the logging module, which we've configured to log to a file printed
    when you click "Test Run". For example:
    logging.info("My debugging message")
    Note that, unlike print, logging.info will take only a single argument.
    So logging.info("my message") will work, but logging.info("my","message") will not.

    The logging module can be used to give you more control over your debugging
    or other messages than you can get by printing them. In this exercise, print
    statements from your mapper will go to your reducer, and print statements
    from your reducer will be considered your final output. By contrast, messages
    logged via the loggers we configured will be saved to two files, one
    for the mapper and one for the reducer. If you click "Test Run", then we
    will show the contents of those files once your program has finished running.
    The logging module also has other capabilities; see
    https://docs.python.org/2/library/logging.html for more information.
    """

    file_input = open("turnstile_data_master_with_weather.csv",'r')
    for line in file_input:
        data = line.strip().split(',')
        if len(data) != 22 or data[1] == 'UNIT':
            continue
        print("{0}\t{1}".format(data[1], data[6]))
        logging.info(len(data))
mapper()
```

```
R001    0.0
R001    217.0
R001    890.0
R001    2451.0
R001    4400.0
R001    3372.0
R002    0.0
R002    42.0
R002    50.0
R002    316.0
R002    633.0
R002    639.0
R003    0.0
R003    0.0
R003    0.0
R003    0.0
R003    0.0
R004    0.0
R004    0.0
R004    0.0
```

In [28]:
```python
import logging
import sys

logging.basicConfig(filename="logger.log", format='%(message)s',
                    level=logging.INFO, filemode='w')

def mapper():
    '''
    For this exercise, compute the average value of the ENTRIESn_hourly column
    for different weather types. Weather type will be defined based on the
    combination of the columns fog and rain (which are boolean values).
    For example, one output of our reducer would be the average hourly entries
    across all hours when it was raining but not foggy.

    Each line of input will be a row from our final Subway-MTA dataset in csv format.
    You can check out the input csv file and its structure below:
    https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/turnstile_data_master_with_weather.csv

    Note that this is a comma-separated file.

    This mapper should PRINT (not return) the weather type as the key (use the
    given helper function to format the weather type correctly) and the number in
    the ENTRIESn_hourly column as the value. They should be separated by a tab.
    For example: 'fog-norain\t12345'

    Since you are printing the output of your program, printing a debug
    statement will interfere with the operation of the grader. Instead,
    use the logging module, which we've configured to log to a file printed
    when you click "Test Run". For example:
    logging.info("My debugging message")
    Note that, unlike print, logging.info will take only a single argument.
    So logging.info("my message") will work, but logging.info("my","message") will not.
    '''

    # Takes in variables indicating whether it is foggy and/or rainy and
    # returns a formatted key that you should output.  The variables passed in
    # can be booleans, ints (0 for false and 1 for true) or floats (0.0 for
    # false and 1.0 for true), but the strings '0.0' and '1.0' will not work,
    # so make sure you convert these values to an appropriate type before
    # calling the function.
    def format_key(fog, rain):
        return '{}fog-{}rain'.format(
            '' if fog else 'no',
            '' if rain else 'no'
        )
    file_input = open("turnstile_data_master_with_weather.csv",'r')
    for line in file_input:
        data = line.strip().split(',')
        if len(data) != 22 or data[1] == 'UNIT':
            continue
        print("{0}\t{1}".format(format_key(float(data[14]), float(data[15])), data[6]))
        logging.info(len(data))

mapper()
```

```
nofog-norain    0.0
nofog-norain    217.0
nofog-norain    890.0
nofog-norain    2451.0
nofog-norain    4400.0
nofog-norain    3372.0
nofog-norain    0.0
nofog-norain    42.0
nofog-norain    50.0
nofog-norain    316.0
nofog-norain    633.0
nofog-norain    639.0
nofog-norain    0.0
nofog-norain    0.0
nofog-norain    0.0
nofog-norain    0.0
nofog-norain    0.0
nofog-norain    0.0
nofog-norain    0.0
nofog-norain    0.0
```

In [29]:
```python
import sys
import logging


logging.basicConfig(filename="logger.log", format='%(message)s',
                    level=logging.INFO, filemode='w')

def mapper():
    """
    In this exercise, for each turnstile unit, you will determine the date and time
    (in the span of this data set) at which the most people entered through the unit.

    The input to the mapper will be the final Subway-MTA dataset, the same as
    in the previous exercise. You can check out the csv and its structure below:
    https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/turnstile_data_master_with_weather.csv

    For each line, the mapper should return the UNIT, ENTRIESn_hourly, DATEn, and
    TIMEn columns, separated by tabs. For example:
    'R001\t100000.0\t2011-05-01\t01:00:00'

    Since you are printing the output of your program, printing a debug
    statement will interfere with the operation of the grader. Instead,
    use the logging module, which we've configured to log to a file printed
    when you click "Test Run". For example:
    logging.info("My debugging message")
    Note that, unlike print, logging.info will take only a single argument.
    So logging.info("my message") will work, but logging.info("my","message") will not.
    """
    file_input = open("turnstile_data_master_with_weather.csv",'r')
    for line in file_input:
        data = line.strip().split(',')
        if len(data) != 22 or data[1] == 'UNIT':
            continue
        print("{0}\t{1}\t{2}\t{3}".format(data[1], data[6], data[2], data[3]))
        logging.info(len(data))

mapper()
```

```
R001    0.0     2011-05-01      01:00:00
R001    217.0   2011-05-01      05:00:00
R001    890.0   2011-05-01      09:00:00
R001    2451.0  2011-05-01      13:00:00
R001    4400.0  2011-05-01      17:00:00
R001    3372.0  2011-05-01      21:00:00
R002    0.0     2011-05-01      01:00:00
R002    42.0    2011-05-01      05:00:00
R002    50.0    2011-05-01      09:00:00
R002    316.0   2011-05-01      13:00:00
R002    633.0   2011-05-01      17:00:00
R002    639.0   2011-05-01      21:00:00
R003    0.0     2011-05-01      00:00:00
R003    0.0     2011-05-01      04:00:00
R003    0.0     2011-05-01      12:00:00
R003    0.0     2011-05-01      16:00:00
R003    0.0     2011-05-01      20:00:00
R004    0.0     2011-05-01      00:00:00
R004    0.0     2011-05-01      04:00:00
R004    0.0     2011-05-01      08:00:00
```

In [29]: