In [38]:
```
import sqlite3
from sqlalchemy import create_engine
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

conn = sqlite3.connect('consumer_database.sqlite')

#Using sqlite from sqlalchemy, import the tables as dataframes
disk_engine = create_engine('sqlite:///consumer_database.sqlite')
```

In [13]:
```
df_requests = pd.read_sql_query('SELECT * FROM requests', disk_engi
ne)
df_requests.head()
```

Out[13]:

	request_id	user_id	category_id	location_id	creation_time
0	1	1001	46	35	2013-07-01 07:48:54.000000
1	2	1002	83	19	2013-07-01 04:55:25.000000
2	3	1003	63	91	2013-07-01 09:34:53.000000
3	4	1004	56	2	2013-07-01 10:16:40.000000
4	5	1005	64	11	2013-07-01 03:45:47.000000

In [14]:
```
df_invites = pd.read_sql_query('SELECT * FROM invites', disk_engin
e)
df_invites.head()
```

Out[14]:

	invite_id	request_id	user_id	sent_time
0	1	1	312	2013-07-01 13:20:05.072029
1	2	1	850	2013-07-01 15:49:33.110849
2	3	1	555	2013-07-01 13:39:18.608330
3	4	1	917	2013-07-01 08:56:11.751781
4	5	1	215	2013-07-01 08:40:24.151670

In [15]:
```
df_quotes = pd.read_sql_query('SELECT * FROM quotes', disk_engine)
df_quotes.head()
```

Out[15]:

	quote_id	invite_id	sent_time
0	1	4	2013-07-01 11:04:44.204874
1	2	5	2013-07-01 10:39:30.083032
2	3	6	2013-07-01 16:43:37.668191
3	4	8	2013-07-01 22:10:35.168437
4	5	9	2013-07-01 13:02:03.174618

In [16]:
```
df_locations = pd.read_sql_query('SELECT * FROM locations', disk_en
gine)
df_locations.head()
```

Out[16]:

	location_id	name
0	1	New York-Newark-Jersey City, NY-NJ-PA
1	2	Los Angeles-Long Beach-Anaheim, CA
2	3	Chicago-Naperville-Elgin, IL-IN-WI
3	4	Dallas-Fort Worth-Arlington, TX
4	5	Houston-The Woodlands-Sugar Land, TX

In [17]:
```
df_categories = pd.read_sql_query('SELECT * FROM categories', dis
k_engine)
df_categories.head()
```

Out[17]:

	category_id	name
0	1	Photography
1	2	Window Installation
2	3	Portrait Photography
3	4	Wedding Band
4	5	Home Security and Alarms

In [18]:
```
df_users = pd.read_sql_query('SELECT * FROM users', disk_engine)
df_users.head()
```

Out[18]:

	user_id	email
0	1	william@idxydp.com
1	2	william@dhgtae.com
2	3	liam@aqpvfh.com
3	4	elizabeth@hpgruv.com
4	5	isabella@omwtoj.com

In [20]:
```
#The Quote table is related to the Invites Table using the invite_i
d key which also provides the time the
#vendor/contractor has taken to respond to the Invite.
df_quotes[df_quotes.invite_id == 4]
```

Out[20]:

	quote_id	invite_id	sent_time
0	1	4	2013-07-01 11:04:44.204874

In [21]:
```
df_invites[df_invites.invite_id == 4]
#As we can see, the invite table provides the request_id and the us
er_id and the Invite sent_time
#of the Request to Vendor.
```

Out[21]:

	invite_id	request_id	user_id	sent_time
3	4	1	917	2013-07-01 08:56:11.751781

In [22]:
```
df_requests[df_requests.request_id == 1]
#The Request Table is linked to the category and the location and a
lso provides a creation time by the customer
```

Out[22]:

	request_id	user_id	category_id	location_id	creation_time
0	1	1001	46	35	2013-07-01 07:48:54.000000

In [23]:
```
df_locations[df_locations.location_id == 35]
#As we can see this location was made in Austin, Texas
```

Out[23]:

	location_id	name
34	35	Austin-Round Rock, TX

In [24]:
```
df_categories[df_categories.category_id == 46]
#for the request of a DJ
```

Out[24]:

	category_id	name
45	46	DJ

In [25]:
```
#Now lets explore the data, as we can see the tables are related by
keys, so now we will combine the tables
#to create a larger dataframe that we can apply our analysis.

#Now lets Combine the tables using the Merge function in Pandas

#Merge dataframes using inner join on the Location dataframe and th
e request_dataframe
frame = pd.DataFrame.merge(df_requests, df_locations, left_on='loca
tion_id', right_on='location_id')

#Merge Categories and Location Dataframe into one dataframe using c
ategory_id as the key
frame1 = pd.DataFrame.merge(frame, df_categories, left_on='categor
y_id', right_on='category_id')

#Merge Invite Dataframe with the Request Dataframe using the reques
t_id as the key do an inner join
frame2 = pd.DataFrame.merge(frame1, df_invites, left_on='request_i
d', right_on='request_id')

#Merge users Dataframe with the Invite Dataframe using the user_id
as the key do an inner join
frame3 = pd.DataFrame.merge(frame2, df_users, left_on='user_id_y',
right_on='user_id')

#Merge Quotes dataframe to the Final Dataframe including Requests,
Invites and time
#using the invite_id as the Key for the inner join.
final_df = pd.DataFrame.merge(frame3, df_quotes, left_on='invite_i
d', right_on='invite_id')
final_df.head()
```

Out[25]:

	request_id	user_id_x	category_id	location_id	creation_time	name_x	nan
0	3630	4630	46	4	2013-08-14 18:32:50.000000	Dallas-Fort Worth-Arlington, TX	DJ
1	2038	3038	87	15	2013-07-25 01:44:30.000000	Seattle-Tacoma-Bellevue, WA	Con Serv
2	802	1802	87	56	2013-07-10 08:38:02.000000	Fresno, CA	Con Serv
3	61	1061	48	5	2013-07-01 16:46:37.000000	Houston-The Woodlands-Sugar Land, TX	Hou Clea
4	1783	2783	48	5	2013-07-22 05:57:40.000000	Houston-The Woodlands-Sugar Land, TX	Hou Clea

In [26]:

```
#Rename the columns since the inner join, created names with variab
les name_x, name_y, sent_time_x, sent_time_y
final_df.rename(columns={'sent_time_x':'sent_time_invite',
                         'sent_time_y':'sent_time_quote',
                         'name_x':'location_name',
                         'name_y':'category_name',
                         'user_id_x':'customer_id'}, inplace=True)
final_df.columns
```

Out[26]:

```
Index([u'request_id', u'customer_id', u'category_id', u'location_i
d', u'creation_time', u'location_name', u'category_name', u'invit
e_id', u'user_id_y', u'sent_time_invite', u'user_id', u'email', u'q
uote_id', u'sent_time_quote'], dtype='object')
```

In [32]:
```
#Sort the data by creation_time
final_df.sort_index(axis=0, by='creation_time').head()
```

Out[32]:

	request_id	customer_id	category_id	location_id	creation_time	location_
5389	90	1090	87	36	2013-07-01 00:02:41.000000	Nashville-Davidson Murfreesb Franklin, T
11734	90	1090	87	36	2013-07-01 00:02:41.000000	Nashville-Davidson Murfreesb Franklin, T
3154	14	1014	53	52	2013-07-01 00:19:39.000000	Grand Ra Wyoming
11765	14	1014	53	52	2013-07-01 00:19:39.000000	Grand Ra Wyoming
9391	14	1014	53	52	2013-07-01 00:19:39.000000	Grand Ra Wyoming

In [33]:
```
##-------------------------------------------------------------------
---------------------
# Convert the time the Request was created by the User and entered
into the database
# final_df['creation_time_test'] = pd.to_datetime(final_df.creatio
n_time, "%Y-%M-%D")
##-------------------------------------------------------------------
---------------------

#convert the invite sent_time & quote sent_time to datetime format
final_df['sent_time_invite'] = pd.to_datetime(final_df['sent_time_i
nvite'])
final_df['sent_time_quote'] = pd.to_datetime(final_df['sent_time_qu
ote'])
```

In [34]:
```
#Define a function get_minutes to convert the hours and minutes int
o total minutes
#All plot are determined in minutes
def get_minutes(row):
    return (row['sent_time_quote'] - row['sent_time_invite']).tota
l_seconds()/60
```

In [35]:
```
#Apply the get_minutes function to all the rows of the dataframe wh
ich we will use to plot the data
final_df['time_taken'] = final_df.apply(get_minutes, axis=1)
final_df['time_taken'].head(10)
```

Out[35]:
```
0    582.703773
1    151.333102
2    203.248777
3    139.626905
4    229.123087
5     76.358164
6     49.772543
7    226.651886
8     23.220268
9    161.601691
Name: time_taken, dtype: float64
```

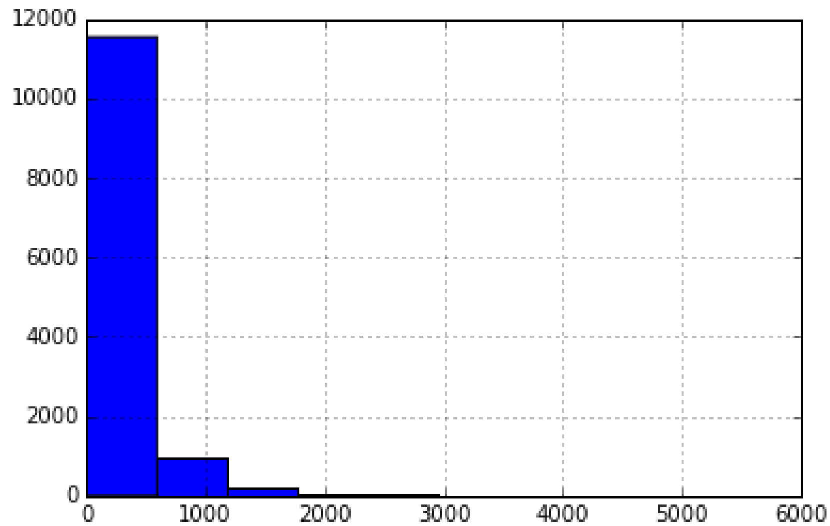
In [36]:
```
#-------------------------------------------------------------------
----
# Now we can start visualizing and start making some sense of the d
ata
# Let us interpret the dataset
#-------------------------------------------------------------------
----

#Lets plot some basic statistics of the data
final_df.time_taken.describe()
```

Out[36]:
```
count    12819.000000
mean       269.018999
std        334.432133
min          3.683707
25%         83.725640
50%        163.494942
75%        321.977402
max       5911.546642
Name: time_taken, dtype: float64
```

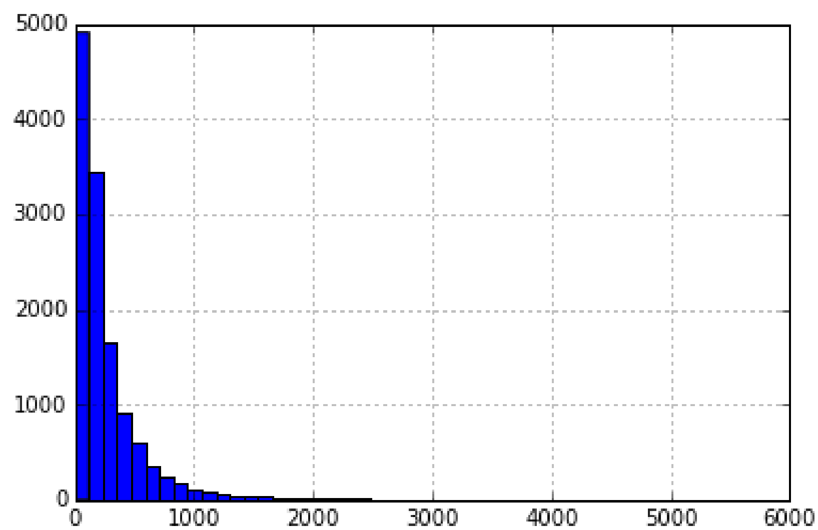
In [41]:
```
final_df['time_taken'].hist()
```

Out[41]: <matplotlib.axes._subplots.AxesSubplot at 0x10c8b2f10>



In [40]:
```
#lets see the distribution of the data
final_df['time_taken'].hist(bins=50)
```

Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x10c4e18d0>

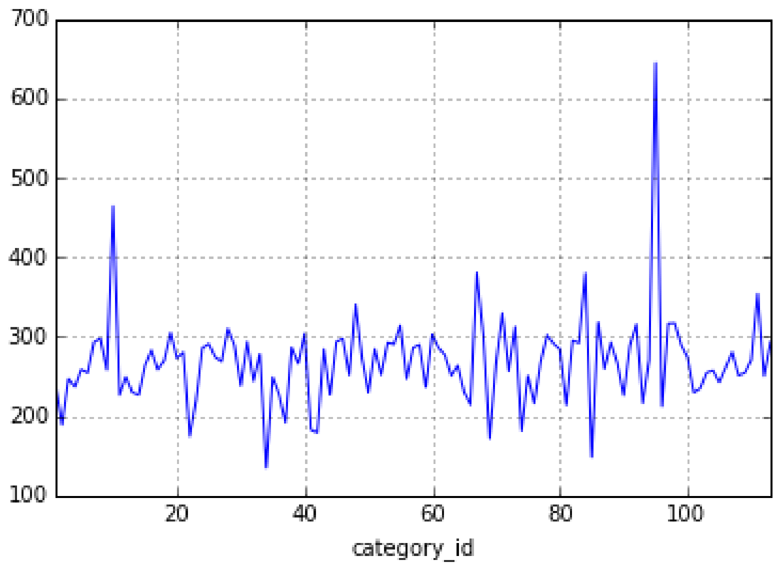


In [43]:
```
#Group by category_id and plot the mean
final_df.groupby('category_id').time_taken.mean().head()
```

Out[43]:
```
category_id
1              241.476854
2              189.189453
3              247.140482
4              237.160914
5              259.125132
Name: time_taken, dtype: float64
```

In [44]:
```
final_df.groupby('category_id').time_taken.mean().plot()
```

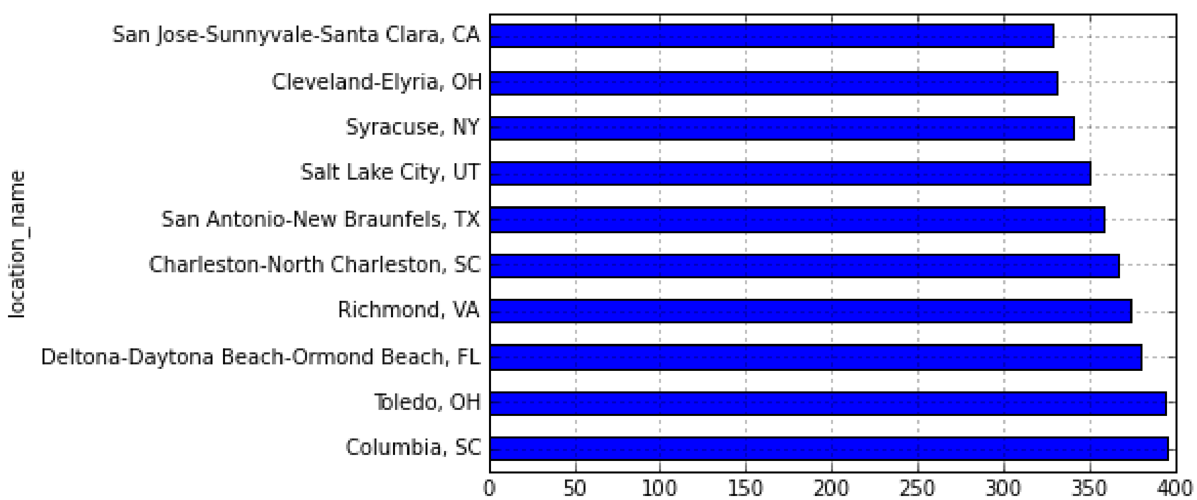
Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x10cb6dd50>



In [45]:
```
#Group by location and Category and plot the time taken to Invite t
o Quote rate
final_df.groupby(['location_name']).time_taken.mean().order(ascendi
ng=False)[:10].plot(kind='barh')

#As plotted below, we can see that SC, OH, FL take > 300 minutes (5
Hours) to respond to a Invite
```

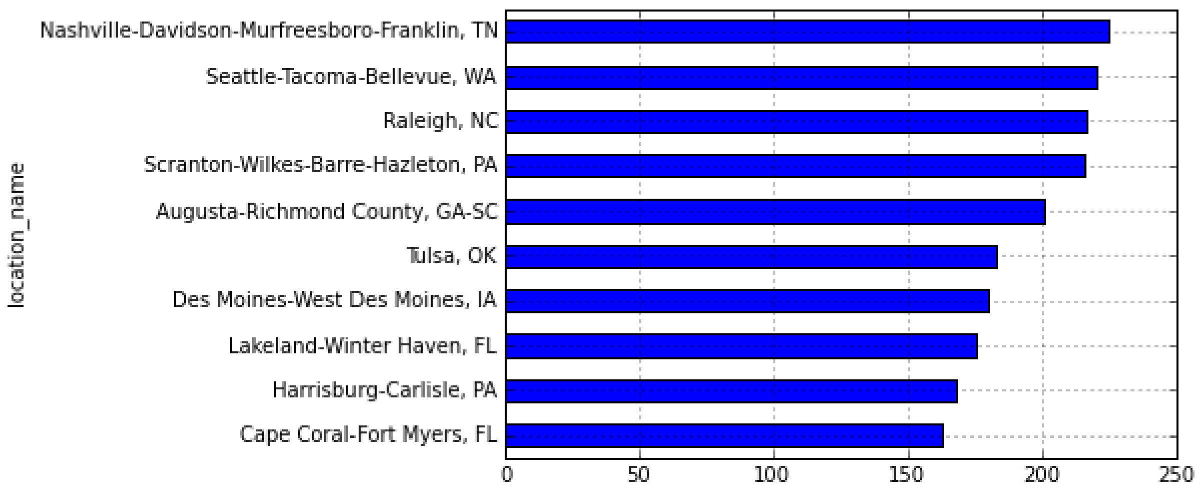
Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x10cbdb110>



In [46]:
```
final_df.groupby(['location_name']).time_taken.mean().order(ascendi
ng=True)[:10].plot(kind='barh')

#The Bar Graph below, shows the cities, states that take the least
time to respond to an Invite.
```

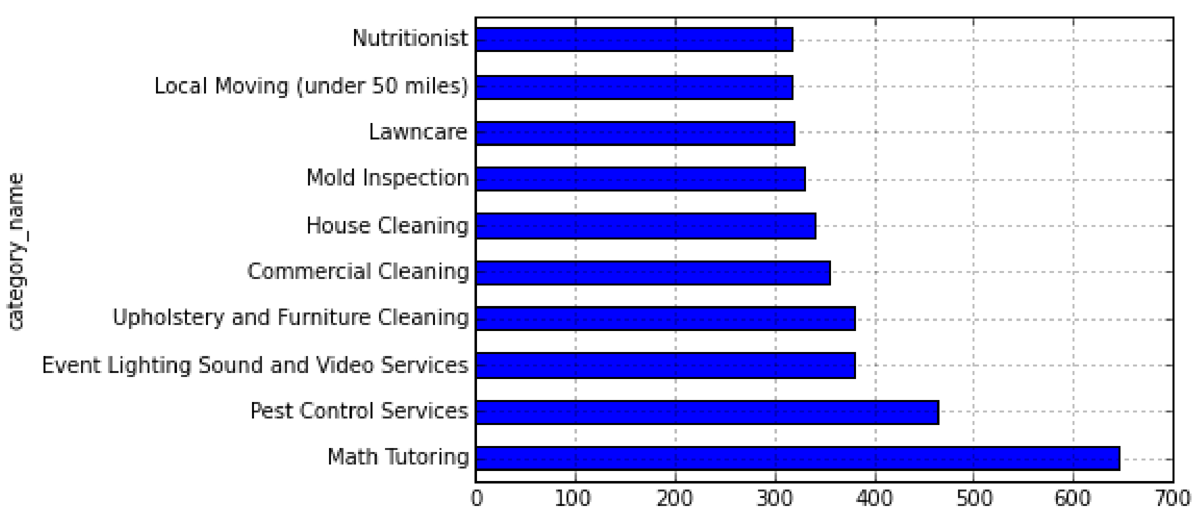
Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x10cd045d0>



In [47]:
```
final_df.groupby(['category_name']).time_taken.mean().order(ascendi
ng=False)[:10].plot(kind='barh')

#Categories not dependent on state, show that Math Tutoring, Pest C
ontrol Services take more than 6 hours to respond
# to a Invite.
```

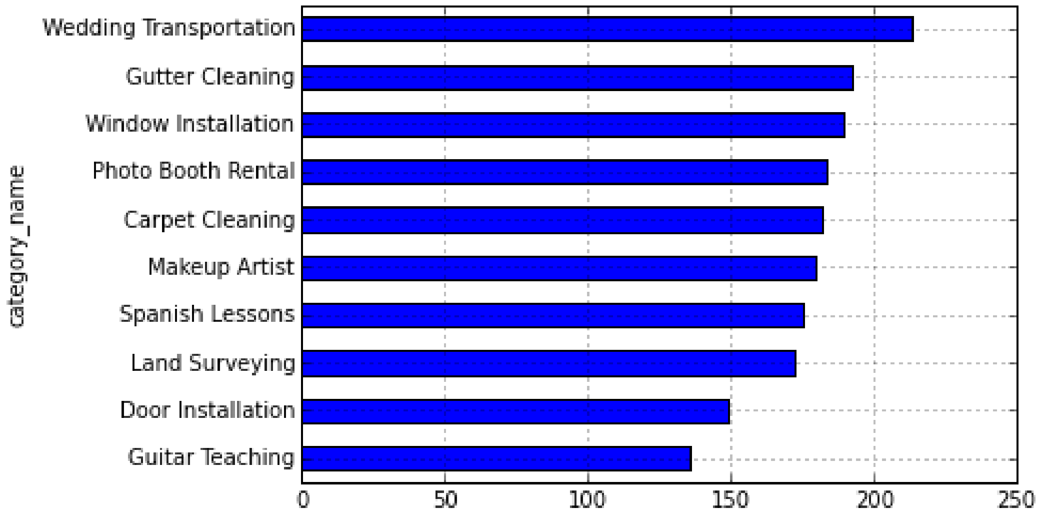
Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x10cf0a350>



In [48]:
```
final_df.groupby(['category_name']).time_taken.mean().order(ascendi
ng=True)[:10].plot(kind='barh')

#In contrast Guitar Teaching, independent of the state, takes less
than 2.5 hours to respond to an invite.
```

Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x10d12d210>

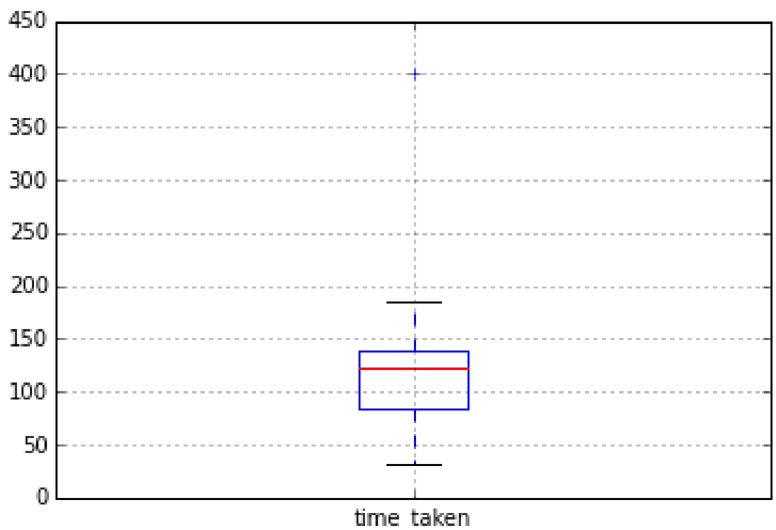


In [49]:
```
#Box plots are useful in seeing the mean and seeing how far we are
from the mean
#In this case we plot box plots for Guitar Teaching (34), Math Tuto
ring

df_categories[df_categories.name == 'Guitar Teaching']
final_df[final_df['category_id'] == 34]
final_df[final_df['category_id'] == 34].boxplot('time_taken')
```

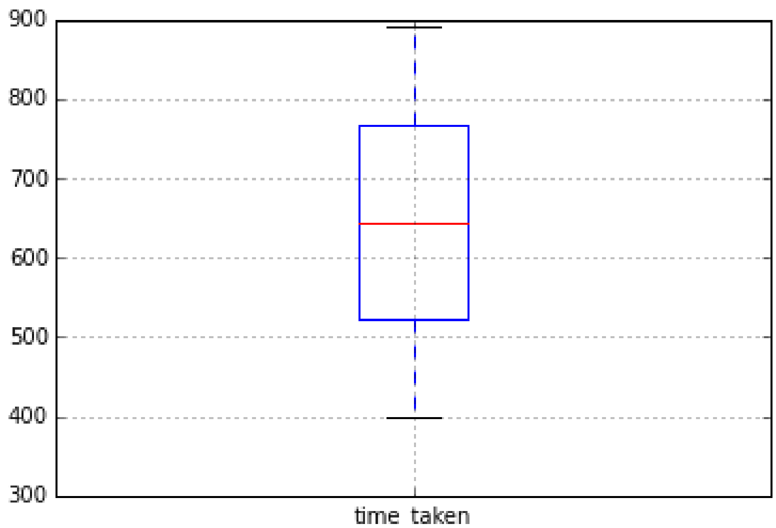
```
/Users/karunsiddana/anaconda/lib/python2.7/site-packages/pandas/too
ls/plotting.py:2625: FutureWarning:
The default value for 'return_type' will change to 'axes' in a futu
re release.
 To use the future behavior now, set return_type='axes'.
 To keep the previous behavior and silence this warning, set retur
n_type='dict'.
  warnings.warn(msg, FutureWarning)
```

Out[49]:
```
{'boxes': [<matplotlib.lines.Line2D at 0x10d429e10>],
 'caps': [<matplotlib.lines.Line2D at 0x10d436dd0>,
  <matplotlib.lines.Line2D at 0x10d443350>],
 'fliers': [<matplotlib.lines.Line2D at 0x10d443fd0>],
 'means': [],
 'medians': [<matplotlib.lines.Line2D at 0x10d443990>],
 'whiskers': [<matplotlib.lines.Line2D at 0x10d4360d0>,
  <matplotlib.lines.Line2D at 0x10d436790>]}
```



In [50]:
```
df_categories[df_categories.name == 'Math Tutoring']
final_df[final_df['category_id'] == 95]
final_df[final_df['category_id'] == 95].boxplot('time_taken')
```

Out[50]:
```
{'boxes': [<matplotlib.lines.Line2D at 0x10d55f810>],
 'caps': [<matplotlib.lines.Line2D at 0x10d56e7d0>,
  <matplotlib.lines.Line2D at 0x10d56ed10>],
 'fliers': [<matplotlib.lines.Line2D at 0x10d5779d0>],
 'means': [],
 'medians': [<matplotlib.lines.Line2D at 0x10d577390>],
 'whiskers': [<matplotlib.lines.Line2D at 0x10d55fa90>,
  <matplotlib.lines.Line2D at 0x10d56e190>]}
```



In [51]:
```
#This shows us the number of quotes per categories, popular demands made by customers.
final_df.category_name.value_counts()
```

Out[51]:
```
Balloon Artistry                      603
Personal Training                     433
Tennis Instruction                    385
Tree and Shrub Service                369
Bartending                            364
Window Repair                         362
Carpet Installation or Replacement    349
Wedding Videography                   332
TV Mounting                           312
Wiring                                304
Landscaping                           303
Wedding Planning                      296
Power Washing                         289
Algebra Tutoring                      286
Personal Chef Services                251
...
Carpentry                        19
Door Installation                19
Event Decorator and Designing    16
Wedding Decorating               16
Spanish Lessons                  15
Commercial Photography           13
Window Installation              12
Band Entertainment               10
Guitar Teaching                  10
Moon Bounce Rental Services       9
Lawn Mowing                       8
Pest Control Services             8
Handyman                          7
Mold Remediation                  5
Math Tutoring                     2
Length: 113, dtype: int64
```

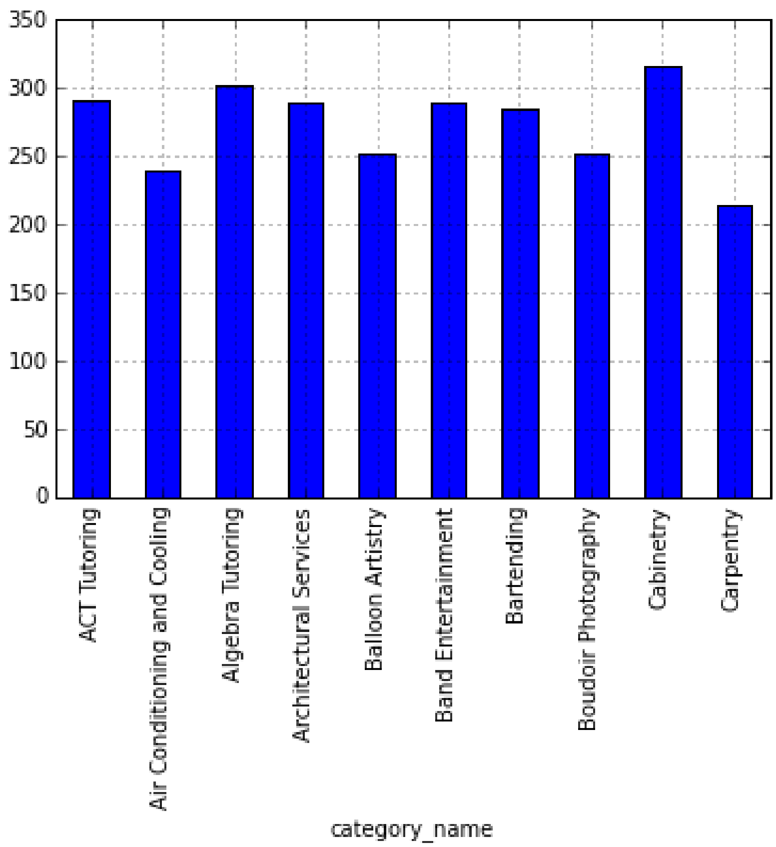
In [53]:
```
final_df.groupby(['category_name', 'location_name']).time_taken.mea
n()
```

```
Out[53]: category_name                    location_name
         ACT Tutoring                     Chicago-Naperville-Elgin, IL-IN-WI
         82.653711
                                          Cleveland-Elyria, OH
         298.456209
                                          Lakeland-Winter Haven, FL
         290.790624
                                          Los Angeles-Long Beach-Anaheim, CA
         237.090981
                                          Miami-Fort Lauderdale-West Palm Beac
         h, FL       114.163497
                                          New York-Newark-Jersey City, NY-NJ-PA
         197.572272
                                          Philadelphia-Camden-Wilmington, PA-N
         J-DE-MD     171.545887
                                          Virginia Beach-Norfolk-Newport News,
         VA-NC     1259.270699
         Air Conditioning and Cooling     Albany-Schenectady-Troy, NY
         82.918300
                                          Albuquerque, NM
         492.189298
                                          Atlanta-Sandy Springs-Roswell, GA
         25.959638
                                          Augusta-Richmond County, GA-SC
         190.505172
                                          Boston-Cambridge-Newton, MA-NH
         144.528911
                                          Buffalo-Cheektowaga-Niagara Falls, NY
         403.059220
                                          Chicago-Naperville-Elgin, IL-IN-WI
         239.418051
         ...
         Yoga Lessons   Los Angeles-Long Beach-Anaheim, CA                35
         2.188411
                        Memphis, TN-MS-AR                                 39
         8.145703
                        Miami-Fort Lauderdale-West Palm Beach, FL         17
         5.443689
                        Minneapolis-St. Paul-Bloomington, MN-WI           64
         3.881966
                        Nashville-Davidson-Murfreesboro-Franklin, TN       5
         8.956417
                        Philadelphia-Camden-Wilmington, PA-NJ-DE-MD        9
         3.649433
                        Providence-Warwick, RI-MA                         21
         2.008718
                        Richmond, VA                                      32
         0.942188
                        Rochester, NY                                     24
         3.219847
                        Salt Lake City, UT                                51
         9.901620
                        San Diego-Carlsbad, CA                            20
```

```
1.770593
                 San Francisco-Oakland-Fremont, CA              27
8.838941
                 Seattle-Tacoma-Bellevue, WA                    13
0.042730
                 St. Louis, MO-IL                               13
0.617393
                 Virginia Beach-Norfolk-Newport News, VA-NC     26
5.144510
Name: time_taken, Length: 2255, dtype: float64
```

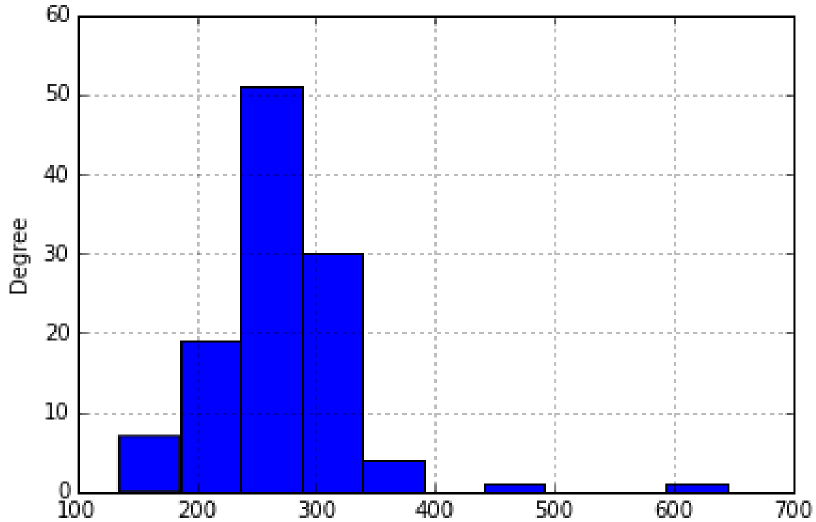
In [54]:
```
final_df.groupby('category_name').time_taken.mean()[:10].plot(kin
d='bar')
```

Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x10d4adfd0>



In [55]:
```
final_df.groupby('category_name').time_taken.mean().plot(kind='his
t')
```

Out[55]: <matplotlib.axes._subplots.AxesSubplot at 0x10e00af10>



In [56]:
```
final_df.groupby(['category_name', 'location_name']).agg({'time_taken': 'mean'})
```

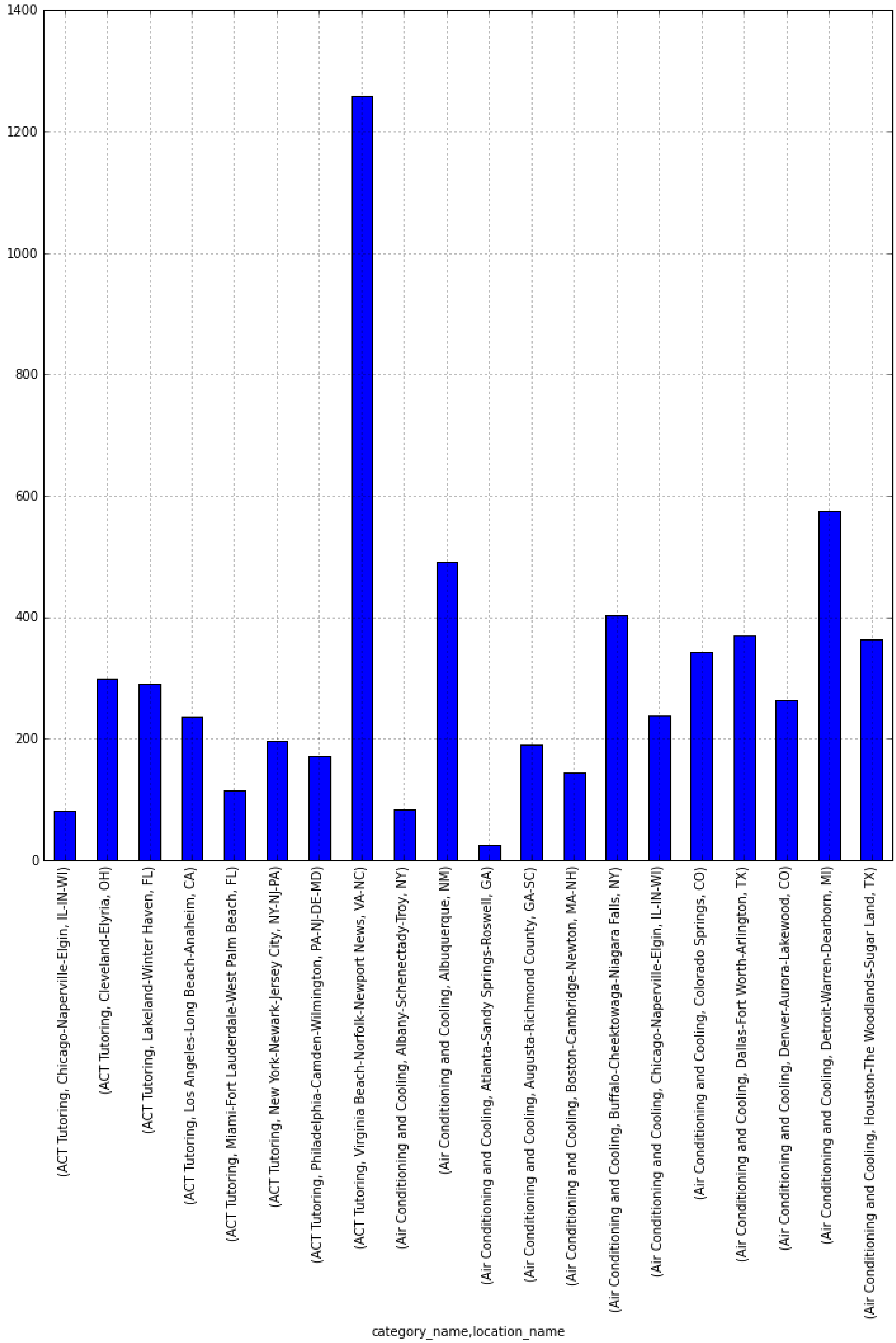
Out[56]:

		time_taken
category_name	**location_name**	
ACT Tutoring	**Chicago-Naperville-Elgin, IL-IN-WI**	82.653711
	Cleveland-Elyria, OH	298.456209
	Lakeland-Winter Haven, FL	290.790624
	Los Angeles-Long Beach-Anaheim, CA	237.090981
	Miami-Fort Lauderdale-West Palm Beach, FL	114.163497
	New York-Newark-Jersey City, NY-NJ-PA	197.572272
	Philadelphia-Camden-Wilmington, PA-NJ-DE-MD	171.545887
	Virginia Beach-Norfolk-Newport News, VA-NC	1259.270699
Air Conditioning and Cooling	**Albany-Schenectady-Troy, NY**	82.918300
	Albuquerque, NM	492.189298
	Atlanta-Sandy Springs-Roswell, GA	25.959638
	Augusta-Richmond County, GA-SC	190.505172
	Boston-Cambridge-Newton, MA-NH	144.528911
	Buffalo-Cheektowaga-Niagara Falls, NY	403.059220
	Chicago-Naperville-Elgin, IL-IN-WI	239.418051
	Colorado Springs, CO	343.791524
	Dallas-Fort Worth-Arlington, TX	370.936954
	Denver-Aurora-Lakewood, CO	263.998112
	Detroit-Warren-Dearborn, MI	574.658408
	Houston-The Woodlands-Sugar Land, TX	363.866179
	Las Vegas-Henderson-Paradise, NV	192.748420
	Los Angeles-Long Beach-Anaheim, CA	68.624431
	Memphis, TN-MS-AR	367.961726
	Nashville-Davidson-Murfreesboro-Franklin, TN	276.653748
	New Haven-Milford, CT	364.262052
	New York-Newark-Jersey City, NY-NJ-PA	207.625105
	Philadelphia-Camden-Wilmington, PA-	[illegible]

NJ-DE-MD	213.314805
Pittsburgh, PA	202.742702
San Diego-Carlsbad, CA	612.568571

In [62]:
```
final_df.groupby(['category_name', 'location_name']).time_taken.mea
n()[:20].plot(kind='bar', figsize=(12,12))
```

```
Out[62]: <matplotlib.axes._subplots.AxesSubplot at 0x121c5fd90>
```



In [ ]: