Appendix (A)

In [1]:
```
# 3250 Foundations of Data Science
# Project: City of Toronto Web Page Analytics
# Author: Shaikh Asif
```

In [3]:
```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pylab import *
import calendar
import os.path

# Render plots inline
%matplotlib inline

# Chart aesthetics
pd.set_option('display.mpl_style', 'default')
plt.rcParams['figure.figsize'] = (15, 5)
plt.rcParams['font.family'] = 'sans-serif'
```

In [4]:
```
# Assign Data Path
data_parent_path = 'C:\Temp\cot_web_analytics_monthly'
```

Overall Summary Metrics

In [5]:
```
# Import past 12 month data

# Assign data import date range
start_month, end_month = 7, 7
start_year, end_year = 2016, 2017

# Temp variables for multiple csv import
month, year = start_month, start_year

# Dataframe to hold consolidated monthly data
monthly_summary = pd.DataFrame()

# Loop over data dir and import all Key Metrics-monthlyYYYYMM.csv files
while (year, month) <= (end_year, end_month):
    monthly_file = pd.read_csv(os.path.join(data_parent_path, 'Summary Metrics\Key Metrics-monthly'+
                                            str(year)+'{:02d}'.format(month)+
                                            '.csv'), usecols = [0,2,3,4,6,7,8,9])

    # Replace existing Date field values with MonthName-Year format
    monthly_file['Date Range'] = calendar.month_name[month] + '-' + str(year)

    # Rename existing Date field
    monthly_file.rename(columns = {'Date Range': 'Month'}, inplace = True)

    # Append monthly data to master dataframe
    monthly_summary = monthly_summary.append(monthly_file[:1])

    # Update import loop vars
    month += 1
    if month > 12:
        month = 1
        year += 1
```

In [6]:
```
# Review imported Data
monthly_summary.head(3)
```

Out[6]:

	Month	Page Views	Visits	Visitors	Avg. Time on Site	Avg. Visitors per Day	Page Views per Visit	New Visitors
0	July-2016	8379194.0	3278023.0	2050481.0	365.86	91547.55	2.56	1441149.0
0	August-2016	7050982.0	2701205.0	1753525.0	360.48	76165.58	2.61	1230150.0
0	September-2016	9303078.0	2507596.0	1586047.0	398.72	72432.97	3.71	1096576.0

In [7]:
```
# Get Summary Stats on the 13 months of website traffic data
monthly_summary.describe()
```

Out[7]:

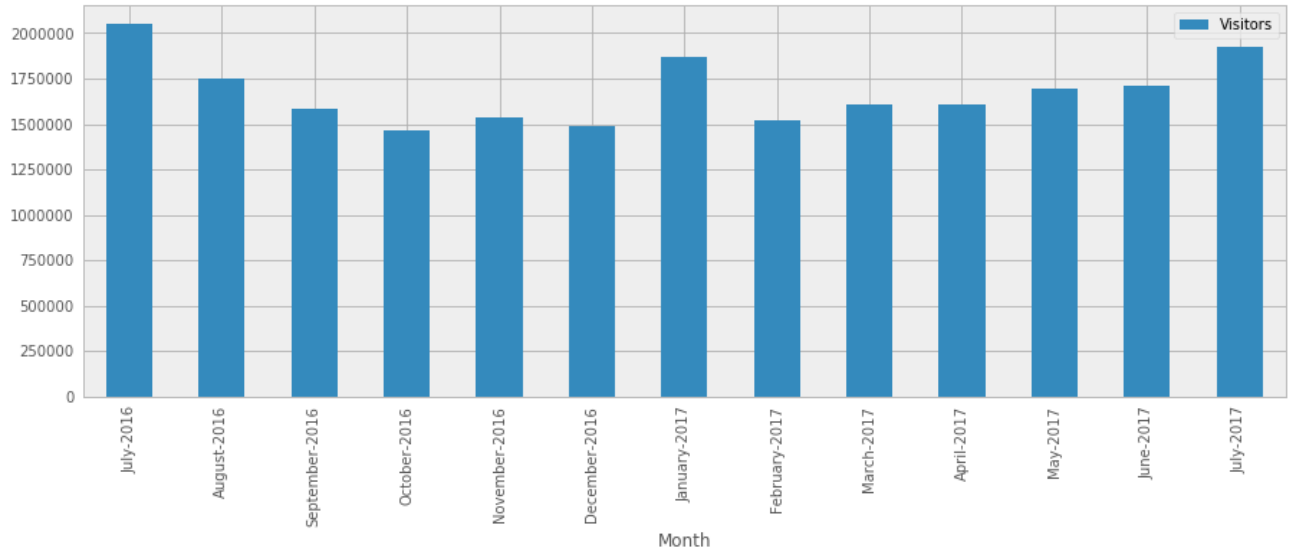
	Page Views	Visits	Visitors	Avg. Time on Site	Avg. Visitors per Day	Page Views per Visit	New Visitors
count	1.300000e+01	1.300000e+01	1.300000e+01	13.000000	13.000000	13.000000	1.300000e+01
mean	8.005108e+06	2.674856e+06	1.677946e+06	384.096154	75790.623846	3.015385	1.164496e+06
std	8.191758e+05	3.302027e+05	1.799571e+05	15.111715	8401.100791	0.328370	1.267872e+05
min	6.806835e+06	2.263357e+06	1.465352e+06	360.480000	63498.450000	2.560000	1.021281e+06
25%	7.378961e+06	2.434474e+06	1.536217e+06	377.590000	72287.870000	2.840000	1.094248e+06
50%	7.943067e+06	2.658008e+06	1.609145e+06	386.300000	74472.610000	2.950000	1.132136e+06
75%	8.503839e+06	2.752698e+06	1.753525e+06	390.420000	79048.100000	3.180000	1.230150e+06
max	9.303078e+06	3.278023e+06	2.050481e+06	417.870000	91547.550000	3.710000	1.441149e+06

In [8]:
```
# Majority of the visits seem to be generated from new visitors
# Determine % of new visitors for each month
monthly_summary['Percent_New_Visitors'] = monthly_summary['New Visitors']/monthly_summary['Visitors'] * 100

# Determine Avg number of visits per visitor
monthly_summary['Avg. Visits per Visitor'] = monthly_summary['Visits']/monthly_summary['Visitors']
monthly_summary.head(3)
```

Out[8]:

	Month	Page Views	Visits	Visitors	Avg. Time on Site	Avg. Visitors per Day	Page Views per Visit	New Visitors	Percent_New_Visitors	Avg. Visits per Visitor
0	July-2016	8379194.0	3278023.0	2050481.0	365.86	91547.55	2.56	1441149.0	70.283460	1.598661
0	August-2016	7050982.0	2701205.0	1753525.0	360.48	76165.58	2.61	1230150.0	70.152978	1.540443
0	September-2016	9303078.0	2507596.0	1586047.0	398.72	72432.97	3.71	1096576.0	69.138935	1.581035

In [9]:
```
# 12 month visitor traffic trend for the website
monthly_summary.plot(kind = 'bar', x = 'Month', y = 'Visitors');
```



Monthly Traffic by Country

In [10]:
```
# Import past 12 month data

# Assign data import date range
start_month, end_month = 7, 7
start_year, end_year = 2016, 2017

# Temp variables for multiple csv import
month, year = start_month, start_year

# Dataframe to hold consolidated monthly data
monthly_visits_by_country=pd.DataFrame()

# Loop over data dir and import all Countries-monthlyYYYYMM.csv files
while (year, month) <= (end_year, end_month):
    monthly_file = pd.read_csv(os.path.join(data_parent_path, 'Global Visits\Countries-monthly'+
                                            str(year)+'{:02d}'.format(month)+
                                            '.csv'), usecols=[0,1,2])

    # The Countries field value in the first record for each file is always empty
    # The first record indicates the number of visits for all countries for the month
    # Filter out the first record in each file as total visits can be derived
    # using the individual country visit counts for each month
    monthly_file.dropna(thresh=3, inplace=True)

    # Some Countries field values have % characters around the country name
    # Remove the % characters
    monthly_file['Countries'] = monthly_file['Countries'].str.replace('%', '')

    # Each monthly file also has ** as the value in the Countries field
    # This will be assumed to be not available and grouped with the
    # 'Unknown Country' value
    monthly_file['Countries'] = monthly_file['Countries'].str.replace('*', '_')
    monthly_file['Countries'] = monthly_file['Countries'].str.replace('__', 'Unknown Country')

    # Append monthly data to master dataframe
    monthly_visits_by_country = monthly_visits_by_country.append(monthly_file)

    # Update import loop vars
    month += 1
    if month > 12:
        month = 1
        year += 1
```

In [11]:
```
# There are now two individual visit counts for 'Unkown Country' per month
# Aggregate the Visit counts grouped by each Country value over the duration of the year
visits_by_country = monthly_visits_by_country.groupby('Countries')['Visits'].sum().reset_index(name = 'Total_Visits

# Review the data
visits_by_country.head(3)
```

Out[11]:

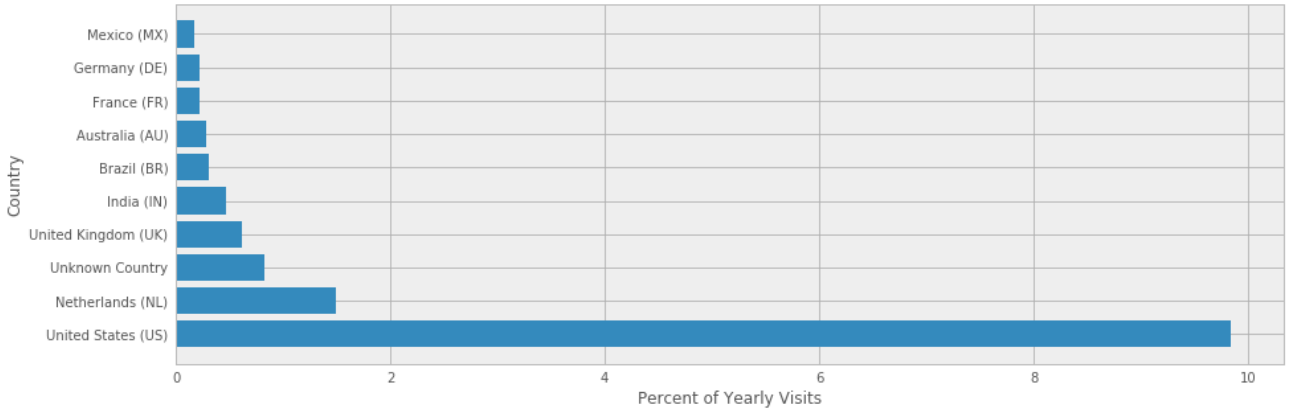
	Countries	Total_Visits
0	AP	1
1	Afghanistan (AF)	402
2	Albania (AL)	1454

In [12]:
```
# Determine % of Visits for the year for each country
visits_by_country['Percent_Visits'] = visits_by_country.Total_Visits/visits_by_country.Total_Visits.sum() *100

# Sort by top countries
visits_by_country.sort_values(by = 'Percent_Visits', ascending = False).head(3)
```

Out[12]:

	Countries	Total_Visits	Percent_Visits
39	Canada (CA)	28921901	83.173138
221	United States (US)	3422540	9.842486
152	Netherlands (NL)	517864	1.489265

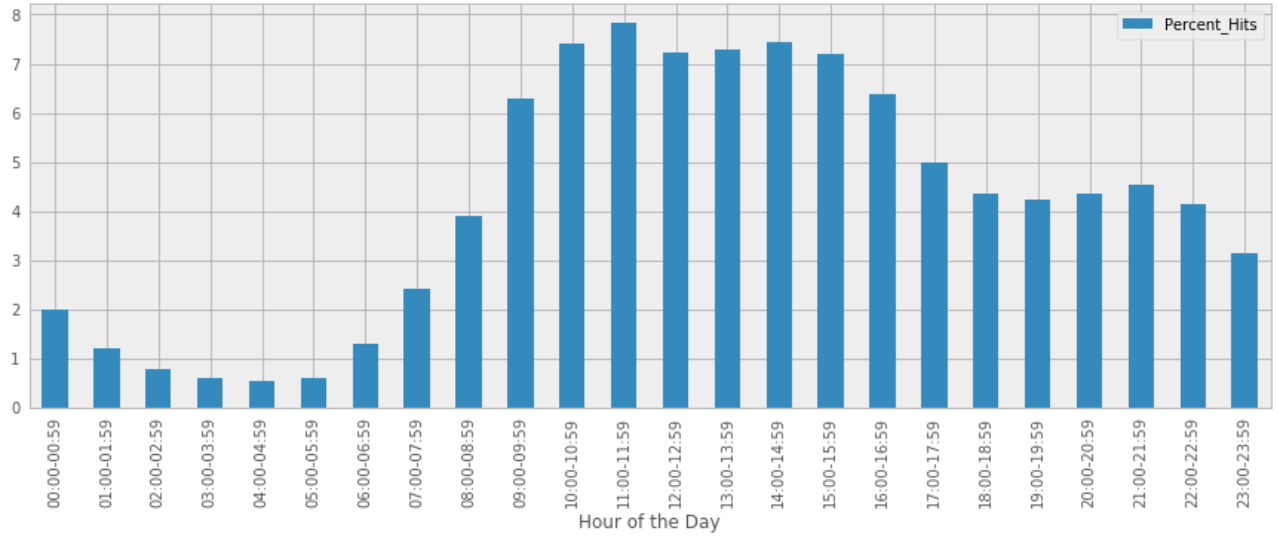
In [13]:
```
# Majority of the visits are from Canada
# Chart the top 10 international countries
bar_chart_data = visits_by_country.sort_values(by ='Percent_Visits', ascending = False).head(11)
bar_chart_data = bar_chart_data.query("Countries != 'Canada (CA)'")

# the bar lengths
val = bar_chart_data['Percent_Visits'].tolist()
# the bar centers on the y axis
pos = arange(10)+.5

barh(pos, val, align = 'center')
yticks(pos, bar_chart_data['Countries'].tolist())
ylabel('Country')
xlabel('Percent of Yearly Visits')
grid(True)
show()
```



Traffic by Time of Day

In [14]:
```
# The data for 2016 is not available
# Traffic patterns by time of day can be observed through 2017 data as well
# Not having 2016 data will not remove any value from or skew the analysis as
# majority of people stay consistent with their routines

# Import 2017 monthly data

# Assign data import date range
start_month, end_month = 1, 7
start_year, end_year = 2017, 2017

# Temp variables for multiple csv import
month, year = start_month, start_year

# Dataframe to hold consolidated monthly data
monthly_hits_by_hour = pd.DataFrame()

# Loop over data dir and import all Hits by Hour of Day-monthlyYYYYMM.csv files
while (year, month) <= (end_year, end_month):
    monthly_file = pd.read_csv(os.path.join(data_parent_path, 'Hourly Hits\Hits by Hour of Day-monthly'+
                                            str(year)+'{:02d}'.format(month)+
                                            '.csv'), usecols=[0,1,2])

    # The Hour of the Day field value in the first record for each file is always empty
    # The first record indicates the number of hits for all hours of the day in the month
    # Filter out the first record in each file as total hits can be derived
    # using the individual hourly hit counts for each month
    monthly_file.dropna(thresh = 3, inplace = True)

    # Append monthly data to master dataframe
    monthly_hits_by_hour = monthly_hits_by_hour.append(monthly_file)

    # Update import loop vars
    month += 1
    if month > 12:
        month = 1
        year += 1
```

In [15]:
```
# Aggregate the hit counts grouped by hour of the day over the duration of the year
hourly_hits_2017 = monthly_hits_by_hour.groupby('Hour of the Day')['Hits'].sum().reset_index(name='Total_Hits')

# Determine % of Visits for the year for each country
hourly_hits_2017['Percent_Hits'] = hourly_hits_2017.Total_Hits/hourly_hits_2017.Total_Hits.sum() *100

# Chart percent of hits for 2017 by hour of day
hourly_hits_2017.plot(kind='bar', x = 'Hour of the Day', y = 'Percent_Hits');
```

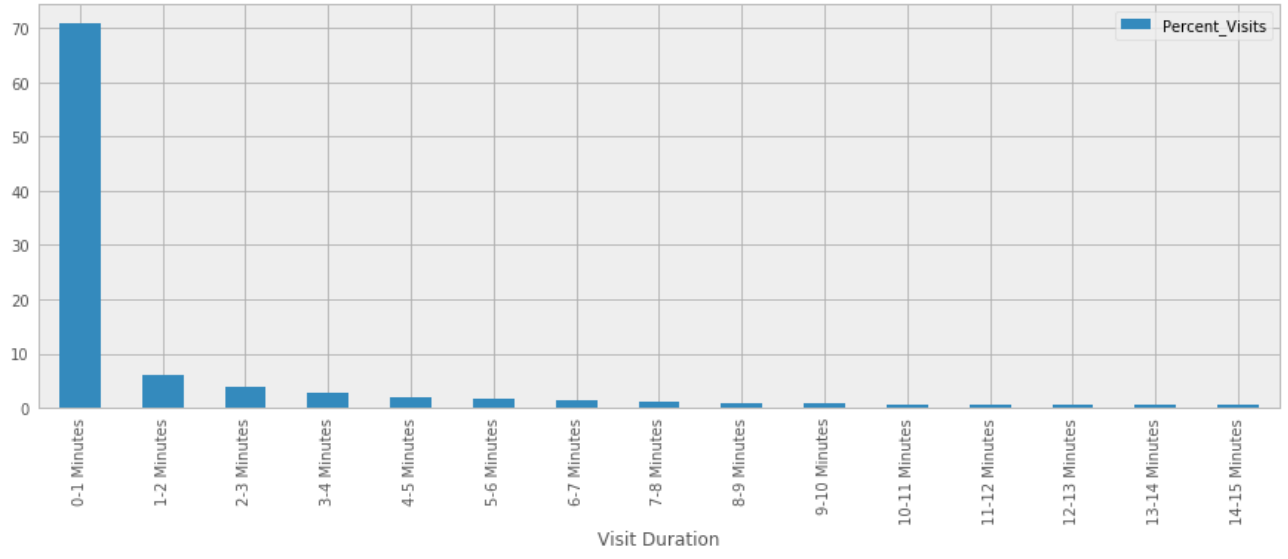


Visit Duration

In [16]:
```
# Import 2017 monthly data

# Assign data import date range
start_month, end_month = 1, 7
start_year, end_year = 2017, 2017

# Temp variables for multiple csv import
month, year = start_month, start_year

# Dataframe to hold consolidated monthly data
monthly_visit_by_duration = pd.DataFrame()

# Loop over data dir and import all Visit Duration Activity-monthlyYYYYMM.csv files
while (year, month) <= (end_year, end_month):
    monthly_file = pd.read_csv(os.path.join(data_parent_path, 'Visit Duration\Visit Duration Activity-monthly'+
                                            str(year)+'{:02d}'.format(month)+
                                            '.csv'), usecols=[0,1,2])

    # The visit duration field value in the first record for each file is always empty
    # The first record indicates the number of visits overall in the month
    # Filter out the first record in each file as total visits can be derived
    # using the individual duration counts for each month
    monthly_file.dropna(thresh = 3, inplace=True)

    # Clean up the duration value of 60+ mins from '&gt; 60 Minutes' to '60+ Minutes'
    monthly_file['Visit Duration'] = monthly_file['Visit Duration'].str.replace('&gt; 60 Minutes', '60+ Minutes')

    # Append monthly data to master dataframe
    monthly_visit_by_duration = monthly_visit_by_duration.append(monthly_file)

    # Update import loop vars
    month += 1
    if month > 12:
        month = 1
        year += 1
```

In [17]:
```
# Aggregate the visits grouped by duration for 2017
visit_durations_2017 = monthly_visit_by_duration.groupby('Visit Duration')['Visits'].sum().reset_index(name='Total_

# Determine % of Visits for the year for each duration bucket
visit_durations_2017['Percent_Visits'] = visit_durations_2017.Total_Visits/visit_durations_2017.Total_Visits.sum()

# Chart percent of visits for 2017 by the top 15 duration buckets
visit_durations_2017 = visit_durations_2017.sort_values(by = 'Percent_Visits', ascending = False).head(15)
visit_durations_2017.plot(kind = 'bar', x = 'Visit Duration', y = 'Percent_Visits')
```

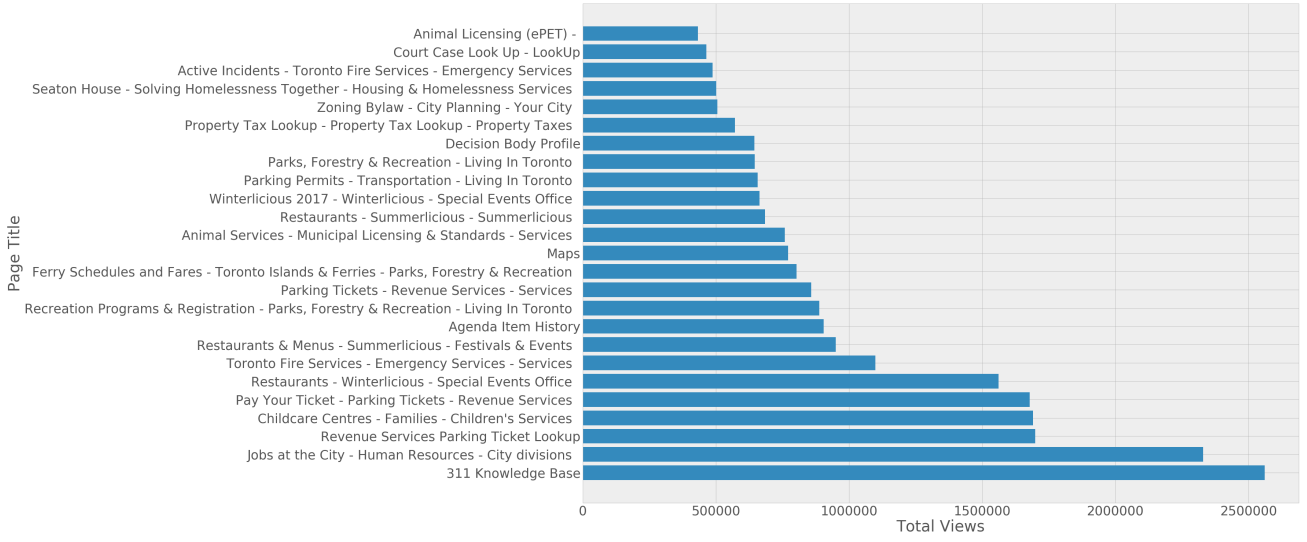
Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xa102400>
```



Top Pages

In [18]:
```
# Import past 12 month data

# Assign data import date range
start_month, end_month = 7, 7
start_year, end_year = 2016, 2017

# Temp variables for multiple csv import
month, year = start_month, start_year

# Dataframe to hold consolidated monthly data
monthly_top_pages = pd.DataFrame()

# Loop over data dir and import all Top Pages-monthlyYYYYMM.csv files
while (year, month) <= (end_year, end_month):
    monthly_file = pd.read_csv(os.path.join(data_parent_path, 'Top Pages\Top Pages-monthly'+
                                            str(year)+'{:02d}'.format(month)+
                                            '.csv'), usecols=[2,5])

    # The Title field value in the first record for each file is always empty
    # The first record indicates the number of views for all webpages for the month
    # Filter out the first record in each file as total views can be derived
    # using the individual webpage counts for each month
    monthly_file.dropna(thresh=2, inplace=True)

    # Append monthly data to master dataframe
    monthly_top_pages = monthly_top_pages.append(monthly_file)

    # Update import loop vars
    month += 1
    if month > 12:
        month = 1
        year += 1
```

In [19]:
```
# Aggregate the views grouped by webpage for past 12 months
top_pages = monthly_top_pages.groupby('Title')['Views'].sum().reset_index(name='Total_Views')

# Filter out the home page and the search results page
top_pages = top_pages.query("Title != 'City of Toronto | City of Toronto' and Title != 'Search Results | City of To

# Get top 25 web pages
top_pages = top_pages.sort_values(by='Total_Views', ascending=False).head(25)

# Some web page titles end with the | City of Toronto string, remove the substring
top_pages['Title'] = top_pages['Title'].str.replace('City of Toronto', '').str.replace('|', '')

# Chart the top 25 web pages for the past 12 months
plt.rcParams['figure.figsize'] = (35, 25)
plt.rcParams['font.family'] = 'sans-serif'
plt.rcParams['font.size'] = '35'

# the bar lengths
val = top_pages['Total_Views'].tolist()
# the bar centers on the y axis
pos = arange(25)+10.5

barh(pos, val, align = 'center')
yticks(pos, top_pages['Title'].tolist())
ylabel('Page Title')
xlabel('Total Views')
grid(True)
show()
```



In [ ]:
```
```