Date 04-07-2021

Line Charts Part-2

In [1]:
```
import pandas as pd
import numpy as np
import plotly.offline as pyo
import plotly.graph_objs as go
```

In [2]:
```
data_solar_power_csv=pd.read_csv("Solar_Power_Generation_Data.csv",
                                 usecols=["DATE_TIME", "AC_POWER"])
data_solar_power_csv
```

Out[2]:

	DATE_TIME	AC_POWER
0	15-05-2020 00:00	0.0
1	15-05-2020 00:00	0.0
2	15-05-2020 00:00	0.0
3	15-05-2020 00:00	0.0
4	15-05-2020 00:00	0.0
...	...	...
67693	17-06-2020 23:45	0.0
67694	17-06-2020 23:45	0.0
67695	17-06-2020 23:45	0.0
67696	17-06-2020 23:45	0.0
67697	17-06-2020 23:45	0.0

67698 rows × 2 columns

In [3]:
```
data_solar_power_time_numpy_array = data_solar_power_csv["DATE_TIME"].to_numpy()
data_solar_power_time_numpy_array
```

Out[3]:
```
array(['15-05-2020 00:00', '15-05-2020 00:00', '15-05-2020 00:00', ...,
       '17-06-2020 23:45', '17-06-2020 23:45', '17-06-2020 23:45'],
      dtype=object)
```

In [4]:
```
data_solar_power_time_numpy_array[data_solar_power_time_numpy_array.size-1] = str(data_solar_power_time_numpy_array[data_solar_power_time_numpy_array.size-1]).split(' ')[1]
data_solar_power_time_numpy_array[data_solar_power_time_numpy_array.size-1]
```

Out[4]:
```
'23:45'
```

In [5]:
```
type(data_solar_power_time_numpy_array[0])
```

Out[5]:
```
str
```

In [6]:
```
for i in range(len(data_solar_power_time_numpy_array)-1):
    data_solar_power_time_numpy_array[i] = str(data_solar_power_time_numpy_array[i]).split(' ')[1]
```

In [7]:
```
data_solar_power_time_numpy_array
```

Out[7]:
```
array(['00:00', '00:00', '00:00', ..., '23:45', '23:45', '23:45'],
      dtype=object)
```

In [8]:
```
data_solar_power_csv["DATE_TIME"] = data_solar_power_time_numpy_array
```

In [9]:
```
data_solar_power_csv.columns=["TIME", "AC_POWER"]
```

In [10]:
```
data_solar_power_csv.set_index("TIME",inplace=True)
data_solar_power_csv
```

Out[10]:

	AC_POWER
TIME	
00:00	0.0
00:00	0.0
00:00	0.0
00:00	0.0
00:00	0.0
...	...
23:45	0.0
23:45	0.0
23:45	0.0
23:45	0.0
23:45	0.0

67698 rows × 1 columns

In [11]:
```
data_solar_power_csv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 67698 entries, 00:00 to 23:45
Data columns (total 1 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   AC_POWER  67698 non-null  float64
dtypes: float64(1)
memory usage: 1.0+ MB
```

In [12]:
```
data_solar_power_csv.describe()
```

Out[12]:

	AC_POWER
count	67698.000000
mean	241.277825
std	362.112118
min	0.000000
25%	0.000000
50%	0.000000
75%	438.215000
max	1385.420000

In [13]:
```
data_solar_power_csv.index.values
```

Out[13]:
```
array(['00:00', '00:00', '00:00', ..., '23:45', '23:45', '23:45'],
      dtype=object)
```

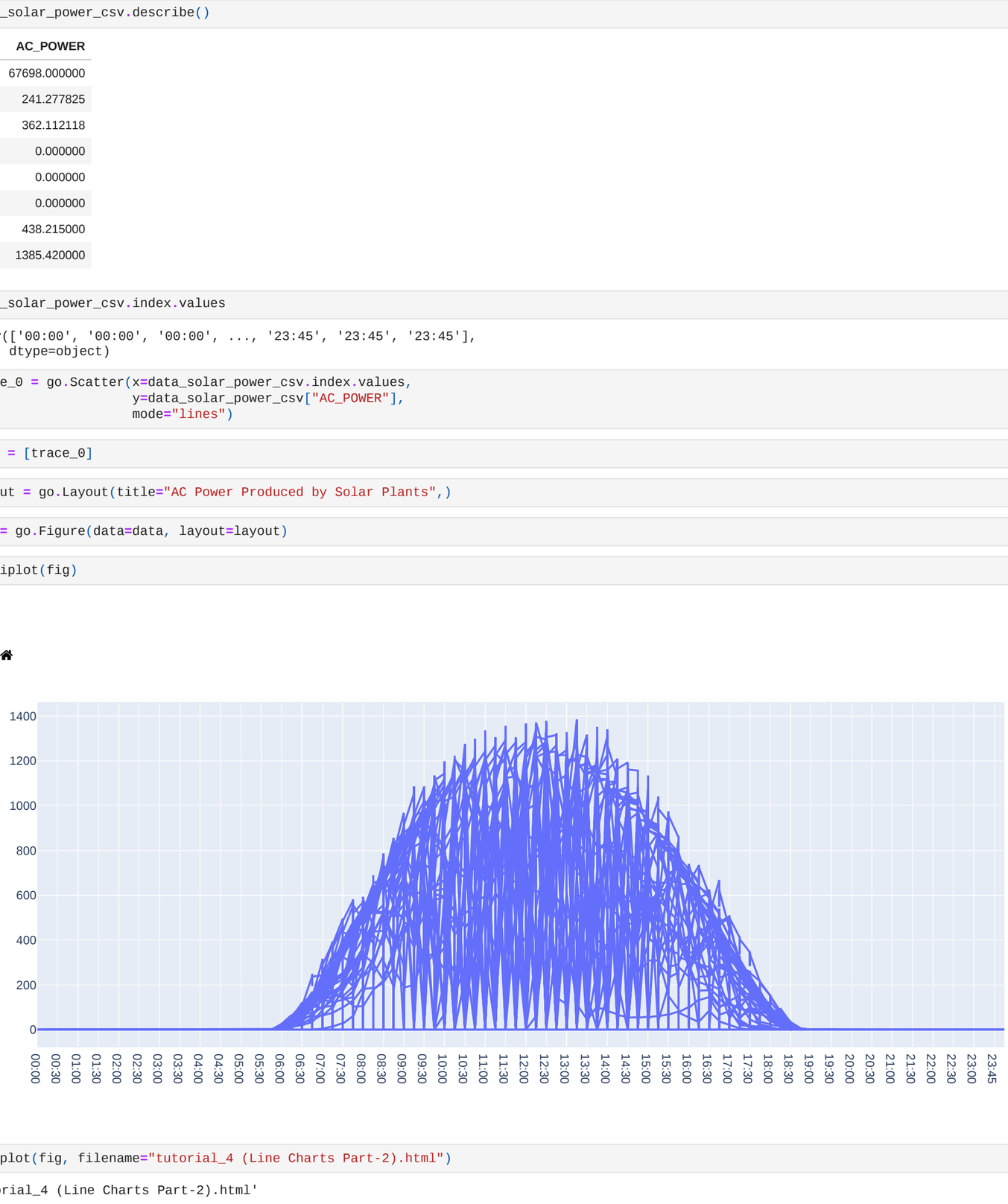
In [14]:
```
trace_0 = go.Scatter(x=data_solar_power_csv.index.values,
                     y=data_solar_power_csv["AC_POWER"],
                     mode="lines")
```

In [15]:
```
data = [trace_0]
```

In [16]:
```
layout = go.Layout(title="AC Power Produced by Solar Plants",)
```

In [17]:
```
fig = go.Figure(data=data, layout=layout)
```

In [18]:
```
pyo.iplot(fig)
```



In [19]:
```
pyo.plot(fig, filename="tutorial_4 (Line Charts Part-2).html")
```

Out[19]:
```
'tutorial_4 (Line Charts Part-2).html'
```

Date 05-07-2021

Line Charts Exercise

In [1]:
```
import pandas as pd
import numpy as np
import plotly.offline as pyo
import plotly.graph_objs as go
```

In [2]:
```
temperature_data = pd.read_csv("2010YumaAZ.csv" ,usecols=["DAY", "LST_TIME", "T_HR_AVG"])
temperature_data
```

Out[2]:

	DAY	LST_TIME	T_HR_AVG
0	TUESDAY	0:00	25.2
1	TUESDAY	1:00	24.1
2	TUESDAY	2:00	24.4
3	TUESDAY	3:00	24.9
4	TUESDAY	4:00	22.8
...	...	...	...
163	MONDAY	19:00	39.4
164	MONDAY	20:00	38.5
165	MONDAY	21:00	37.0
166	MONDAY	22:00	34.7
167	MONDAY	23:00	32.6

168 rows × 3 columns

In [3]:
```
temperature_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 168 entries, 0 to 167
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   DAY       168 non-null    object 
 1   LST_TIME  168 non-null    object 
 2   T_HR_AVG  168 non-null    float64
dtypes: float64(1), object(2)
memory usage: 4.1+ KB
```

In [4]:
```
temperature_data.describe()
```

Out[4]:

	T_HR_AVG
count	168.000000
mean	30.402976
std	6.288974
min	17.900000
25%	25.125000
50%	30.900000
75%	35.525000
max	40.600000

In [5]:
```
days = np.array(['TUESDAY', 'WEDNESDAY', 'THURSDAY', 'FRIDAY', 'SATURDAY','SUNDAY', 'MONDAY'])
```

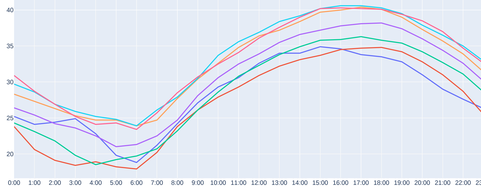
In [6]:
```
data = []
```

In [7]:
```
for day in days:
    trace_0 = go.Scatter(x=temperature_data["LST_TIME"],
                         y=temperature_data[temperature_data["DAY"]==day]['T_HR_AVG'],
                         mode='lines',
                         name=day)
    data.append(trace_0)
```

In [8]:
```
layout = go.Layout(title="Daily Temperature from June 1-7, 2010 in Yuma, Arizona",)
```

In [9]:
```
fig = go.Figure(data=data, layout=layout)
```

In [10]:
```
pyo.iplot(fig)
```

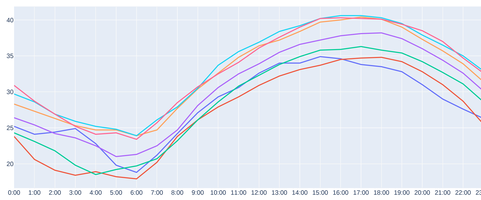


In [11]:
```
pyo.plot(fig, filename="tutorial_5 (Line Charts Exercise).html",
         image_width=1600,
         image_height=900,)
```

Out[11]:
```
'tutorial_5 (Line Charts Exercise).html'
```

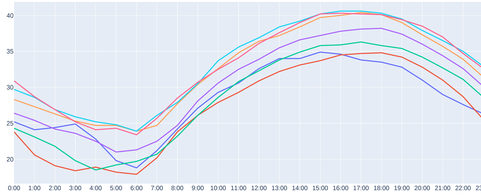
Instructor First Solution is Down

In [12]:
```
######
# Objective: Using the file 2010YumaAZ.csv, develop a Line Chart
# that plots seven days worth of temperature data on one graph.
# You can use a for loop to assign each day to its own trace.
######

# Perform imports here:
import plotly.offline as pyo
import plotly.graph_objs as go
import pandas as pd

# Create a pandas DataFrame from 2010YumaAZ.csv
df = pd.read_csv('2010YumaAZ.csv')
days = ['TUESDAY','WEDNESDAY','THURSDAY','FRIDAY','SATURDAY','SUNDAY','MONDAY']

# Use a for loop to create the traces for the seven days
# There are many ways to do this! Could also do this with a
# list comprehension.

data = []

for day in days:
    trace = go.Scatter(x=df['LST_TIME'],
                       y=df[df['DAY']==day]['T_HR_AVG'],
                       mode='lines',
                       name=day)
    data.append(trace)

# Define the layout
layout = go.Layout(
    title='Daily temperatures from June 1-7, 2010 in Yuma, Arizona',
    hovermode='closest'
)

# Create a fig from data and layout, and plot the fig
fig = go.Figure(data=data, layout=layout)

pyo.iplot(fig)
```



Instructor Second Solution is Down

In [13]:
```
###################
## NOTE: ADVANCED SOLUTION THAT USES ONLY PURE DF CALLS
## THIS IS FOR MORE ADVANCED PANDAS USERS TO TAKE A LOOK AT! :)

######
# Objective: Using the file 2010YumaAZ.csv, develop a Line Chart
# that plots seven days worth of temperature data on one graph.
# You can use a for loop to assign each day to its own trace.
######

# Perform imports here:
import plotly.offline as pyo
import plotly.graph_objs as go
import pandas as pd

# Create a pandas DataFrame from 2010YumaAZ.csv
df = pd.read_csv('2010YumaAZ.csv')

# Define a data variable
data = [{
    'x': df['LST_TIME'],
    'y': df[df['DAY']==day]['T_HR_AVG'],
    'name': day
} for day in df['DAY'].unique()]

# Define the layout
layout = go.Layout(
    title='Daily temperatures from June 1-7, 2010 in Yuma, Arizona',
    hovermode='closest'
)

# Create a fig from data and layout, and plot the fig
fig = go.Figure(data=data, layout=layout)

pyo.iplot(fig)
```



Date 05-07-2021

Bar Charts

In [1]:
```
import pandas as pd
import numpy as np
import plotly.offline as pyo
import plotly.graph_objs as go
```

In [2]:
```
winter_olympics_2018_data = pd.read_csv("2018WinterOlympics.csv")
winter_olympics_2018_data.set_index("Rank", inplace=True)
winter_olympics_2018_data
```

Out[2]:

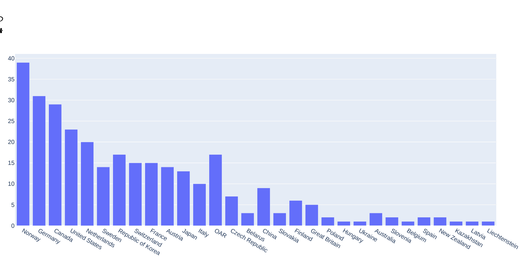
Rank	NOC	Gold	Silver	Bronze	Total
1	Norway	14	14	11	39
2	Germany	14	10	7	31
3	Canada	11	8	10	29
4	United States	9	8	6	23
5	Netherlands	8	6	6	20
6	Sweden	7	6	1	14
7	Republic of Korea	5	8	4	17
8	Switzerland	5	6	4	15
9	France	5	4	6	15
10	Austria	5	3	6	14
11	Japan	4	5	4	13
12	Italy	3	2	5	10
13	OAR	2	6	9	17
14	Czech Republic	2	2	3	7
15	Belarus	2	1	0	3
16	China	1	6	2	9
17	Slovakia	1	2	0	3
18	Finland	1	1	4	6
19	Great Britain	1	0	4	5
20	Poland	1	0	1	2
21	Hungary	1	0	0	1
21	Ukraine	1	0	0	1
23	Australia	0	2	1	3
24	Slovenia	0	1	1	2
25	Belgium	0	1	0	1
26	Spain	0	0	2	2
26	New Zealand	0	0	2	2
28	Kazakhstan	0	0	1	1
28	Latvia	0	0	1	1
28	Liechtenstein	0	0	1	1

In [3]:
```
data = [go.Bar(x = winter_olympics_2018_data["NOC"],
               y = winter_olympics_2018_data["Total"])]
```

In [4]:
```
layout = go.Layout(title = "Medals Won By Contries")
```

In [5]:
```
fig = go.Figure(data, layout)
```

In [6]:
```
pyo.iplot(fig)
```



In [7]:
```
pyo.plot(fig, filename="tutorial_7 (Bar Charts)[Part-1].html",
         image_width=1600,
         image_height=900,)
```

Out[7]:
```
'tutorial_7 (Bar Charts)[Part-1].html'
```

In [8]:
```
trace_1 = go.Bar(x = winter_olympics_2018_data["NOC"],
                 y = winter_olympics_2018_data["Gold"],
                 marker=dict(color="#FFD700"),
                 name="Gold")
```

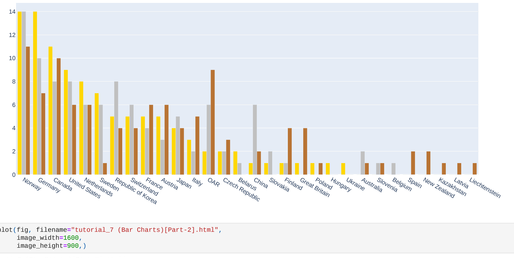
In [9]:
```
trace_2 = go.Bar(x = winter_olympics_2018_data["NOC"],
                 y = winter_olympics_2018_data["Silver"],
                 marker=dict(color="#C0C0C0"),
                 name="Silver")
```

In [10]:
```
trace_3 = go.Bar(x = winter_olympics_2018_data["NOC"],
                 y = winter_olympics_2018_data["Bronze"],
                 marker=dict(color="#B87333"),
                 name="Bronze")
```

In [11]:
```
data = [trace_1, trace_2, trace_3]
```

In [12]:
```
fig = go.Figure(data, layout)
```

In [13]:
```
pyo.iplot(fig)
```



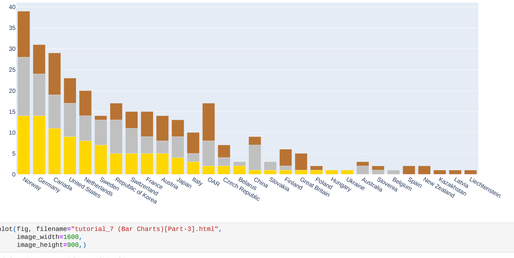
In [14]:
```
pyo.plot(fig, filename="tutorial_7 (Bar Charts)[Part-2].html",
         image_width=1600,
         image_height=900,)
```

Out[14]:
```
'tutorial_7 (Bar Charts)[Part-2].html'
```

In [15]:
```
layout = go.Layout(title = "Medals Won By Contries", barmode="stack")
```

In [16]:
```
fig = go.Figure(data, layout)
```

In [17]:
```
pyo.iplot(fig)
```



In [18]:
```
pyo.plot(fig, filename="tutorial_7 (Bar Charts)[Part-3].html",
         image_width=1600,
         image_height=900,)
```

Out[18]:
```
'tutorial_7 (Bar Charts)[Part-3].html'
```

In [19]:
```
trace_1 = go.Bar(x = winter_olympics_2018_data["NOC"],
                 y = winter_olympics_2018_data["Gold"],
                 marker=dict(color="#FFD700"),
                 marker_pattern_shape="/",
                 name="Gold")
```

In [20]:
```
trace_2 = go.Bar(x = winter_olympics_2018_data["NOC"],
                 y = winter_olympics_2018_data["Silver"],
                 marker=dict(color="#C0C0C0"),
                 marker_pattern_shape="x",
                 name="Silver")
```

In [21]:
```
trace_3 = go.Bar(x = winter_olympics_2018_data["NOC"],
                 y = winter_olympics_2018_data["Bronze"],
                 marker=dict(color="#B87333"),
                 marker_pattern_shape="+",
                 name="Bronze")
```

In [22]:
```
data = [trace_1, trace_2, trace_3]
```

In [23]:
```
fig = go.Figure(data, layout)
```

In [24]:
```
pyo.iplot(fig)
```

In [25]:
```
pyo.plot(fig, filename="tutorial_7 (Bar Charts)[Part-4].html",
         image_width=1600,
         image_height=900,)
```

Out[25]:
```
'tutorial_7 (Bar Charts)[Part-4].html'
```

Date 05-07-2021

Bar Charts Exercise

In [1]:
```
import pandas as pd
import numpy as np
import plotly.offline as pyo
import plotly.graph_objs as go
```

In [2]:
```
mock_survey_data = pd.read_csv("mocksurvey.csv")
mock_survey_data.set_index("Unnamed: 0", inplace=True)
mock_survey_data
```

Out[2]:

Unnamed: 0	Strongly Agree	Somewhat Agree	Neutral	Somewhat Disagree	Strongly Disagree
Question 1	0.45	0.25	0.10	0.12	0.08
Question 2	0.12	0.07	0.48	0.18	0.15
Question 3	0.05	0.22	0.19	0.23	0.31

In [3]:
```
mock_survey_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3 entries, Question 1 to Question 3
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Strongly Agree      3 non-null      float64
 1   Somewhat Agree      3 non-null      float64
 2   Neutral             3 non-null      float64
 3   Somewhat Disagree   3 non-null      float64
 4   Strongly Disagree   3 non-null      float64
dtypes: float64(5)
memory usage: 144.0+ bytes
```

In [4]:
```
mock_survey_data.describe()
```

Out[4]:

	Strongly Agree	Somewhat Agree	Neutral	Somewhat Disagree	Strongly Disagree
count	3.000000	3.000000	3.000000	3.000000	3.000000
mean	0.206667	0.180000	0.256667	0.176667	0.180000
std	0.213620	0.096437	0.198578	0.055076	0.117898
min	0.050000	0.070000	0.100000	0.120000	0.080000
25%	0.085000	0.145000	0.145000	0.150000	0.115000
50%	0.120000	0.220000	0.190000	0.180000	0.150000
75%	0.285000	0.235000	0.335000	0.205000	0.230000
max	0.450000	0.250000	0.480000	0.230000	0.310000

My First Solution Starts Here

In [5]:
```
cols = ["Strongly Agree", "Somewhat Agree", "Neutral", "Somewhat Disagree", "Strongly Disagree"]
cols
```

Out[5]:
```
['Strongly Agree',
 'Somewhat Agree',
 'Neutral',
 'Somewhat Disagree',
 'Strongly Disagree']
```

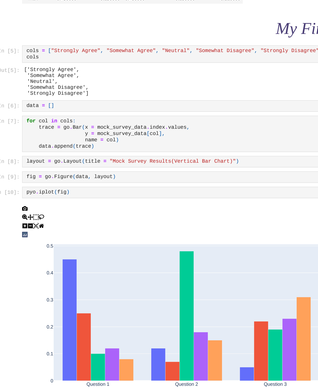
In [6]:
```
data = []
```

In [7]:
```
for col in cols:
    trace = go.Bar(x = mock_survey_data.index.values,
                   y = mock_survey_data[col],
                   name = col)
    data.append(trace)
```

In [8]:
```
layout = go.Layout(title = "Mock Survey Results(Vertical Bar Chart)")
```

In [9]:
```
fig = go.Figure(data, layout)
```

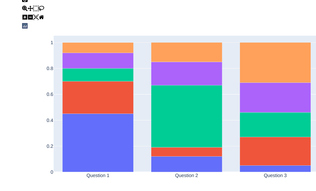
In [10]:
```
pyo.iplot(fig)
```



In [11]:
```
layout = go.Layout(title = "Mock Survey Results(Vertical Bar Chart)", barmode = "stack")
```

In [12]:
```
fig = go.Figure(data, layout)
```

In [13]:
```
pyo.iplot(fig)
```



In [14]:
```
pyo.plot(fig, filename="tutorial_8 (Bar Charts Exercise)[Part-1].html",
         image_width=1600,
         image_height=900,)
```

Out[14]:
```
'tutorial_8 (Bar Charts Exercise)[Part-1].html'
```

My Second Solution Starts Here

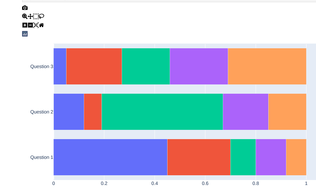
In [15]:
```
data = []
```

In [16]:
```
for col in cols:
    trace = go.Bar(y = mock_survey_data.index.values,
                   x = mock_survey_data[col],
                   name = col,
                   orientation='h')
    data.append(trace)
```

In [17]:
```
layout = go.Layout(title = "Mock Survey Results(Horizontal Bar Chart)", barmode = "stack")
```

In [18]:
```
fig = go.Figure(data, layout)
```

In [19]:
```
pyo.iplot(fig)
```

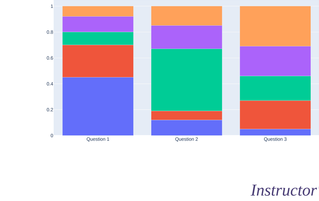


In [20]:
```
pyo.plot(fig, filename="tutorial_8 (Bar Charts Exercise)[Part-2].html",
         image_width=1600,
         image_height=900,)
```

Out[20]:
```
'tutorial_8 (Bar Charts Exercise)[Part-2].html'
```

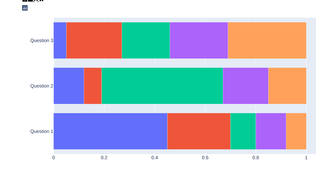
Instructor's First Solution is Down

In [21]:
```
######
# Objective: Create a stacked bar chart from
# the file ../data/mocksurvey.csv. Note that questions appear in
# the index (and should be used for the x-axis), while responses
# appear as column labels.  Extra Credit: make a horizontal bar chart!
######

# Perform imports here:
import plotly.offline as pyo
import plotly.graph_objs as go
import pandas as pd

# create a DataFrame from the .csv file:
df = pd.read_csv('mocksurvey.csv',index_col=0)

# create traces using a list comprehension:
data = [go.Bar(
    x = df.index,
    y = df[response],
    name=response
) for response in df.columns]

# create a layout, remember to set the barmode here
layout = go.Layout(
    title='Mock Survey Results',
    barmode='stack'
)

# create a fig from data & layout, and plot the fig
fig = go.Figure(data=data, layout=layout)

pyo.iplot(fig)
```



Instructor's Second Solution is Down

In [22]:
```
######
# Objective: Create a stacked bar chart from
# the file ../data/mocksurvey.csv. Note that questions appear in
# the index (and should be used for the x-axis), while responses
# appear as column labels.  Extra Credit: make a horizontal bar chart!
######

# Perform imports here:
import plotly.offline as pyo
import plotly.graph_objs as go
import pandas as pd

# create a DataFrame from the .csv file:
df = pd.read_csv('mocksurvey.csv',index_col=0)

# create traces using a list comprehension:
data = [go.Bar(
    x = df.index,          # reverse your x- and y-axis assignments
    y = df[response],
    orientation="h",  # this line makes it horizontal!
    name=response
) for response in df.columns]

# create a layout, remember to set the barmode here
layout = go.Layout(
    title='Mock Survey Results',
    barmode='stack'
)

# create a fig from data & layout, and plot the fig.
fig = go.Figure(data=data, layout=layout)

pyo.iplot(fig)
```



Date 05-07-2021

Bubble Plots

In [1]:
```
import pandas as pd
import plotly.offline as pyo
import plotly.graph_objs as go
```

In [2]:
```
mpg_data_csv = pd.read_csv("mpg.csv")
mpg_data_csv
```

Out[2]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

398 rows × 9 columns

In [3]:
```
mpg_data_csv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object 
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model_year    398 non-null    int64  
 7   origin        398 non-null    int64  
 8   name          398 non-null    object 
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

In [4]:
```
mpg_data_csv.describe()
```

Out[4]:

	mpg	cylinders	displacement	weight	acceleration	model_year	origin
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	2970.424623	15.568090	76.010050	1.572864
std	7.815984	1.701004	104.269838	846.841774	2.757689	3.697627	0.802055
min	9.000000	3.000000	68.000000	1613.000000	8.000000	70.000000	1.000000
25%	17.500000	4.000000	104.250000	2223.750000	13.825000	73.000000	1.000000
50%	23.000000	4.000000	148.500000	2803.500000	15.500000	76.000000	1.000000
75%	29.000000	8.000000	262.000000	3608.000000	17.175000	79.000000	2.000000
max	46.600000	8.000000	455.000000	5140.000000	24.800000	82.000000	3.000000

In [5]:
```
mpg_data_csv.columns
```

Out[5]:
```
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'model_year', 'origin', 'name'],
      dtype='object')
```

In [6]:
```
data = [go.Scatter(x = mpg_data_csv["horsepower"],
                   y = mpg_data_csv["mpg"],
                   text = mpg_data_csv["name"],
                   mode = "markers",
                   marker = dict(size = mpg_data_csv["weight"]/100,
                                 color = mpg_data_csv["cylinders"],
                                 showscale = True))]
```

In [7]:
```
layout = go.Layout(title = "Bubble Chart of mpg DataSet",
                   xaxis = dict(title = 'Horsepower'),
                   yaxis = dict(title = 'Miles Per gallon'),
                   hovermode='closest')
```

In [8]:
```
fig = go.Figure(data, layout)
```

In [9]:
```
pyo.iplot(fig)
```

In [10]:
```
pyo.plot(fig,
         filename = "tutorial_10 (Bubble Plots).html",
         image_width=1600,
         image_height=900,)
```

Out[10]:
```
'tutorial_10 (Bubble Plots).html'
```

Date 06-07-2021

Box Plots

In [1]:
```
import pandas as pd
import numpy as np
import plotly.offline as pyo
import plotly.graph_objs as go
```

In [2]:
```
y = [9,14,14,15,16,18,18,19,19,20,20,23,24,26,27,27,28,29,33,54]
```

In [3]:
```
data = [go.Box(y = y,
        name = "Box Plot Example")]
```

In [4]:
```
pyo.iplot(data)
```

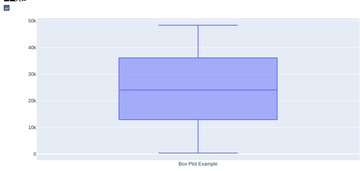
In [5]:
```
pyo.plot(data, filename = "tutorial_13 (Box Plots)[Part-1].html")
```

Out[5]:
```
'tutorial_13 (Box Plots)[Part-1].html'
```

In [6]:
```
np.random.seed(61)
y = np.random.randint(78, 48512, 1000)
```

In [7]:
```
data = [go.Box(y = y,
        name = "Box Plot Example")]
```

In [8]:
```
pyo.iplot(data)
```



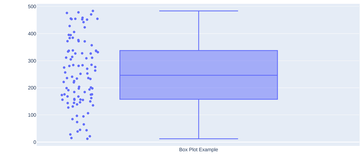
In [9]:
```
pyo.plot(data, filename = "tutorial_13 (Box Plots)[Part-2].html")
```

Out[9]:
```
'tutorial_13 (Box Plots)[Part-2].html'
```

In [10]:
```
np.random.seed(5465)
y = np.random.uniform(7.8, 484.512, 100)
```

In [11]:
```
data = [go.Box(y = y,
               boxpoints = "all",
               name = "Box Plot Example")]
```

In [12]:
```
pyo.iplot(data)
```



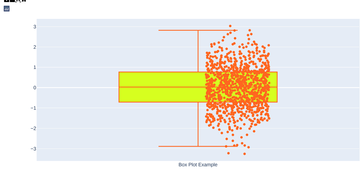
In [13]:
```
pyo.plot(data, filename = "tutorial_13 (Box Plots)[Part-3].html")
```

Out[13]:
```
'tutorial_13 (Box Plots)[Part-3].html'
```

In [14]:
```
np.random.seed(4852)
y = np.random.randn(1000)
```

In [15]:
```
data = [go.Box(y = y,
               boxpoints = "all",
               name = "Box Plot Example",
               jitter = .4,
               pointpos = .5,
               fillcolor = "#ddfffd",
               marker_color = "#ff661f")]
```

In [16]:
```
pyo.iplot(data)
```



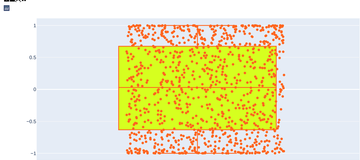
In [17]:
```
pyo.plot(data, filename = "tutorial_13 (Box Plots)[Part-4].html")
```

Out[17]:
```
'tutorial_13 (Box Plots)[Part-4].html'
```

In [18]:
```
y = np.sin(y)
```

In [19]:
```
data = [go.Box(y = y,
               boxpoints = "all",
               name = "Box Plot Example",
               jitter = 1,
               pointpos = .1,
               fillcolor = "#ddfffd",
               marker_color = "#ff661f")]
```

In [20]:
```
pyo.iplot(data)
```



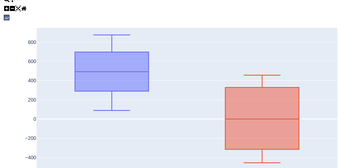
In [21]:
```
pyo.plot(data, filename = "tutorial_13 (Box Plots)[Part-5].html")
```

Out[21]:
```
'tutorial_13 (Box Plots)[Part-5].html'
```

In [22]:
```
np.random.seed(9546)
y = np.random.uniform(87.151, 875.547786, 1000)
y2 = np.sin(y)*456
np.random.shuffle(y2)
```

In [23]:
```
data = [go.Box(y = y,
               name = "Box Plot 1"),
        go.Box(y = y2,
               name = "Box Plot 2")]
```

In [24]:
```
pyo.iplot(data)
```

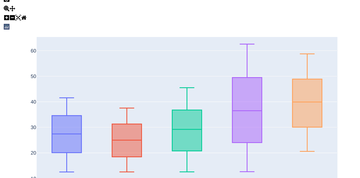


In [25]:
```
pyo.plot(data, filename = "tutorial_13 (Box Plots)[Part-6].html")
```

Out[25]:
```
'tutorial_13 (Box Plots)[Part-6].html'
```

In [26]:
```
data = [go.Box(y = np.random.uniform(12.548, 41.547786, 1000),
               name = "Box Plot 1"),
        go.Box(y = np.random.uniform(12.548, 37.595786, 1000),
               name = "Box Plot 2"),
        go.Box(y = np.random.uniform(12.548, 43.547786, 1000),
               name = "Box Plot 3"),
        go.Box(y = np.random.uniform(12.548, 62.5547786, 1000),
               name = "Box Plot 4"),
        go.Box(y = np.random.uniform(20.548, 58.786, 1000),
               name = "Box Plot 5")]
```

In [27]:
```
pyo.iplot(data)
```



In [28]:
```
pyo.plot(data, filename = "tutorial_13 (Box Plots)[Part-7].html")
```

Out[28]:
```
'tutorial_13 (Box Plots)[Part-7].html'
```

Date 06-07-2021

Box Plots Exercise

In [1]:
```
import pandas as pd
import numpy as np
import plotly.offline as pyo
import plotly.graph_objs as go
```

In [2]:
```
abalone_data_csv = pd.read_csv("abalone.csv")
abalone_data_csv
```

Out[2]:

	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

4177 rows × 9 columns

In [3]:
```
np.random.seed(78)
y1 = np.random.choice(abalone_data_csv["rings"], 10, replace = False)
y2 = np.random.choice(abalone_data_csv["rings"], 10, replace = False)
y1, y2
```

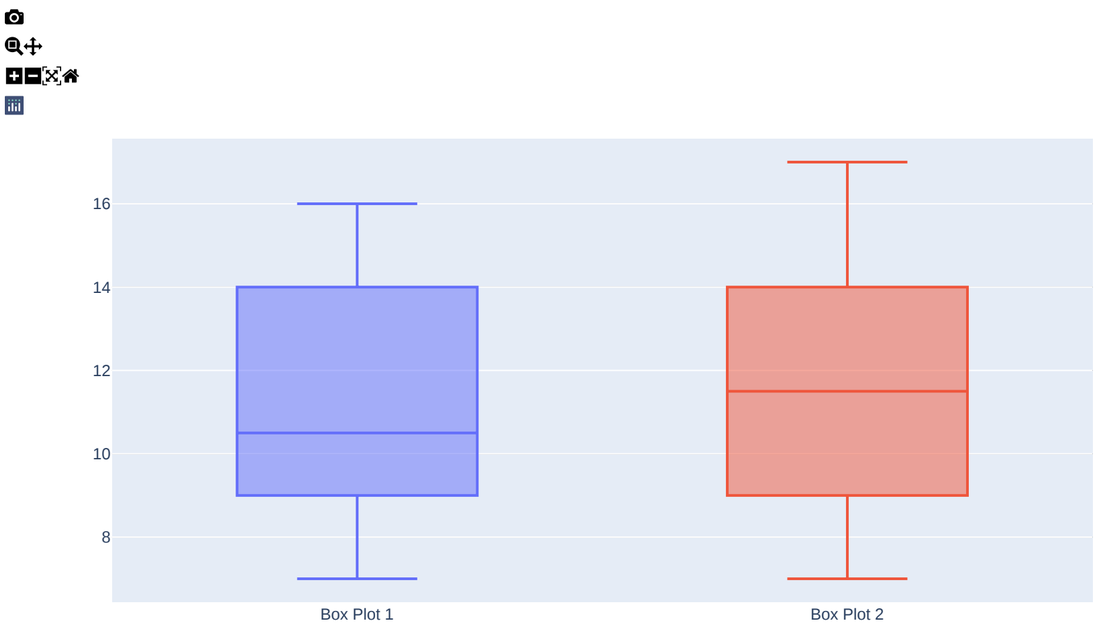
Out[3]:
```
(array([16, 11,  9, 14, 16,  9, 10,  7, 11, 10], dtype=int64),
 array([ 8, 17, 12,  7, 11, 14, 15,  9,  9, 13], dtype=int64))
```

In [4]:
```
data = [go.Box(y = y1,
               name = "Box Plot 1"),
        go.Box(y = y2,
               name = "Box Plot 2")]
```

In [5]:
```
layout = go.Layout(title = "Two Random Samples")
```

In [6]:
```
fig = go.Figure(data, layout)
```

In [7]:
```
pyo.iplot(fig)
```



In [8]:
```
pyo.plot(fig, filename = "tutorial_14 (Box Plots Exercise).html")
```

Out[8]:
```
'tutorial_14 (Box Plots Exercise).html'
```

Date 06-07-2021

Histograms

In [1]:
```
import pandas as pd
import numpy as np
import plotly.offline as pyo
import plotly.graph_objs as go
```

In [2]:
```
mpg_data_csv = pd.read_csv("mpg.csv", usecols = ["mpg"])
mpg_data_csv
```

Out[2]:

	mpg
0	18.0
1	15.0
2	18.0
3	16.0
4	17.0
...	...
393	27.0
394	44.0
395	32.0
396	28.0
397	31.0

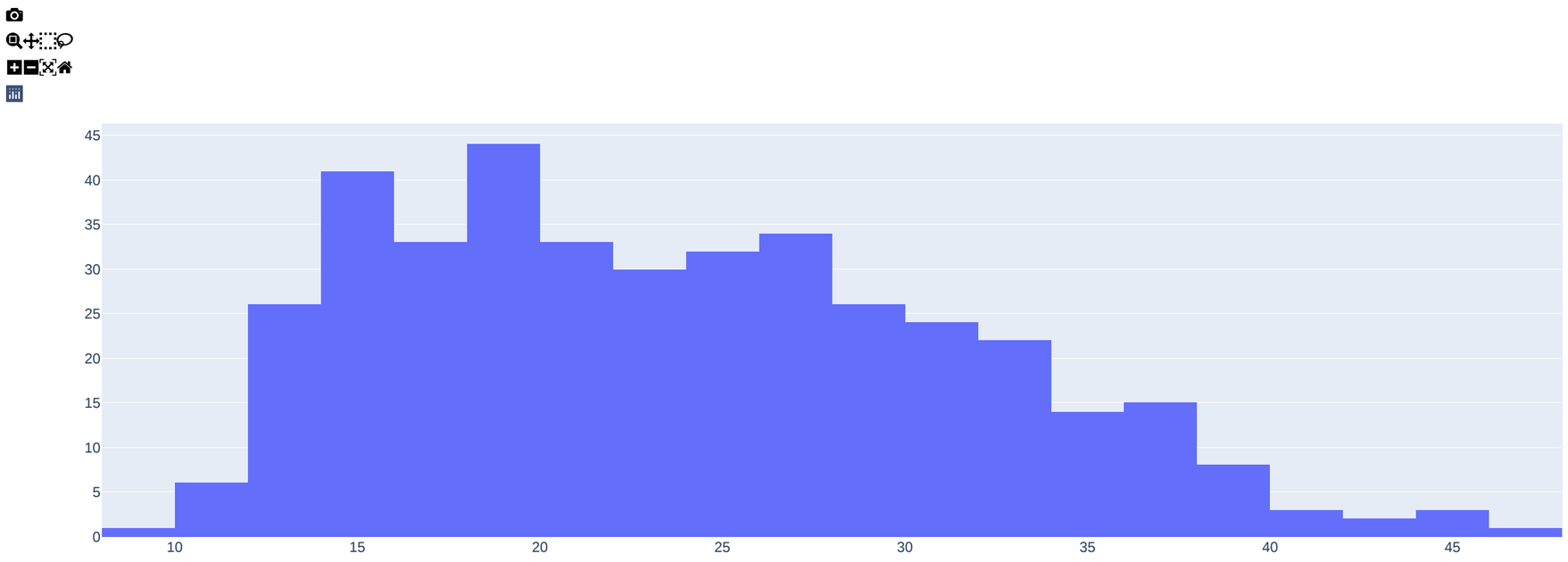
398 rows × 1 columns

In [3]:
```
data = [go.Histogram(x = mpg_data_csv["mpg"])]
```

In [4]:
```
layout = go.Layout(title = "This is a Histogram with size = default bins")
```

In [5]:
```
fig = go.Figure(data, layout)
```

In [6]:
```
pyo.iplot(fig)
```



In [7]:
```
pyo.plot(fig, filename = "tutorial_16 (Histograms)[Part-1]{Graph}.html")
```

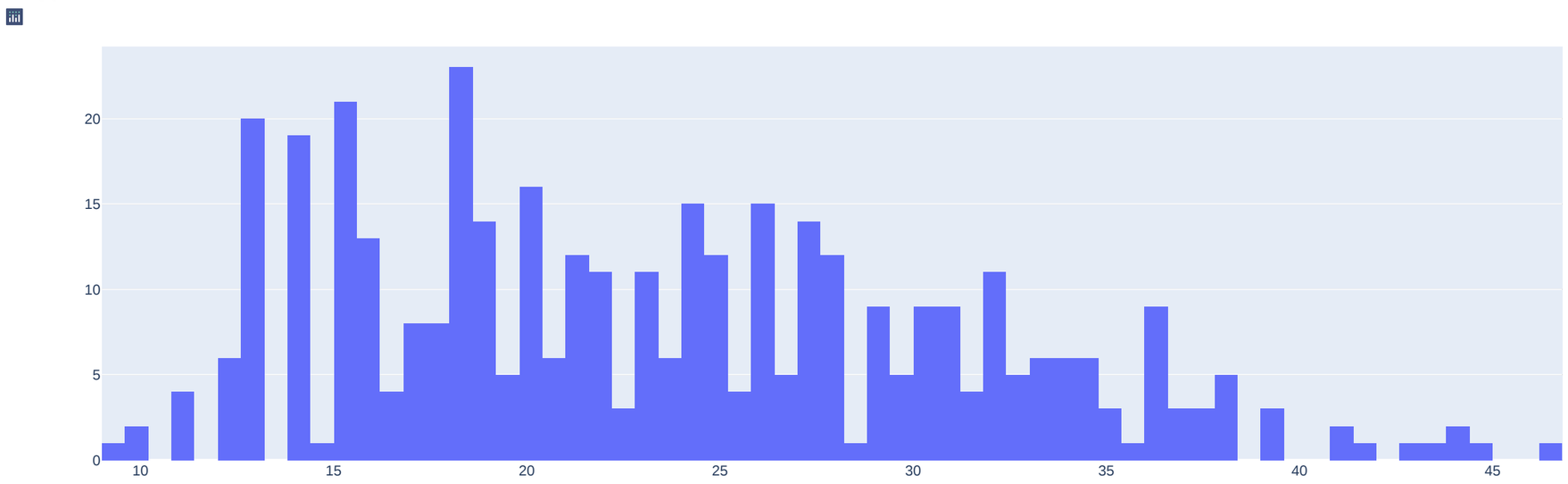
Out[7]:
```
'tutorial_16 (Histograms)[Part-1]{Graph}.html'
```

In [8]:
```
data = [go.Histogram(x = mpg_data_csv["mpg"],
                     xbins = dict(start = 0,
                                  end = 500,
                                  size = .6))]
```

In [9]:
```
layout = go.Layout(title = "This is a Histogram with size = .6 bins")
```

In [10]:
```
fig = go.Figure(data, layout)
```

In [11]:
```
pyo.iplot(fig)
```



In [12]:
```
pyo.plot(fig, filename = "tutorial_16 (Histograms)[Part-2]{Graph}.html")
```

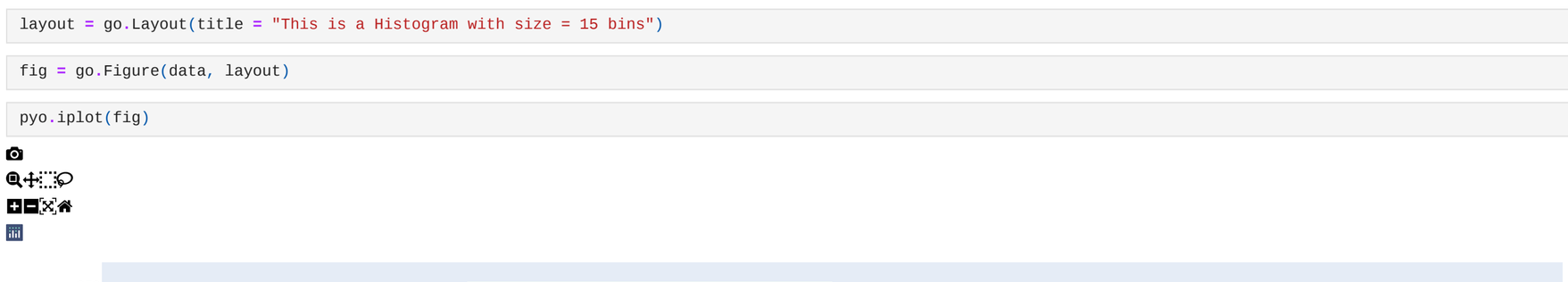
Out[12]:
```
'tutorial_16 (Histograms)[Part-2]{Graph}.html'
```

In [13]:
```
data = [go.Histogram(x = mpg_data_csv["mpg"],
                     xbins = dict(start = 0,
                                  end = 500,
                                  size = 15))]
```

In [14]:
```
layout = go.Layout(title = "This is a Histogram with size = 15 bins")
```

In [15]:
```
fig = go.Figure(data, layout)
```

In [16]:
```
pyo.iplot(fig)
```



In [17]:
```
pyo.plot(fig, filename = "tutorial_16 (Histograms)[Part-3]{Graph}.html")
```

Out[17]:
```
'tutorial_16 (Histograms)[Part-3]{Graph}.html'
```

Date 06-07-2021

Histograms Exercise

In [1]:
```
import pandas as pd
import plotly.offline as pyo
import plotly.graph_objs as go
```

In [2]:
```
abalone_data = pd.read_csv("abalone.csv")
length_data = abalone_data["length"].to_numpy()
length_data
```

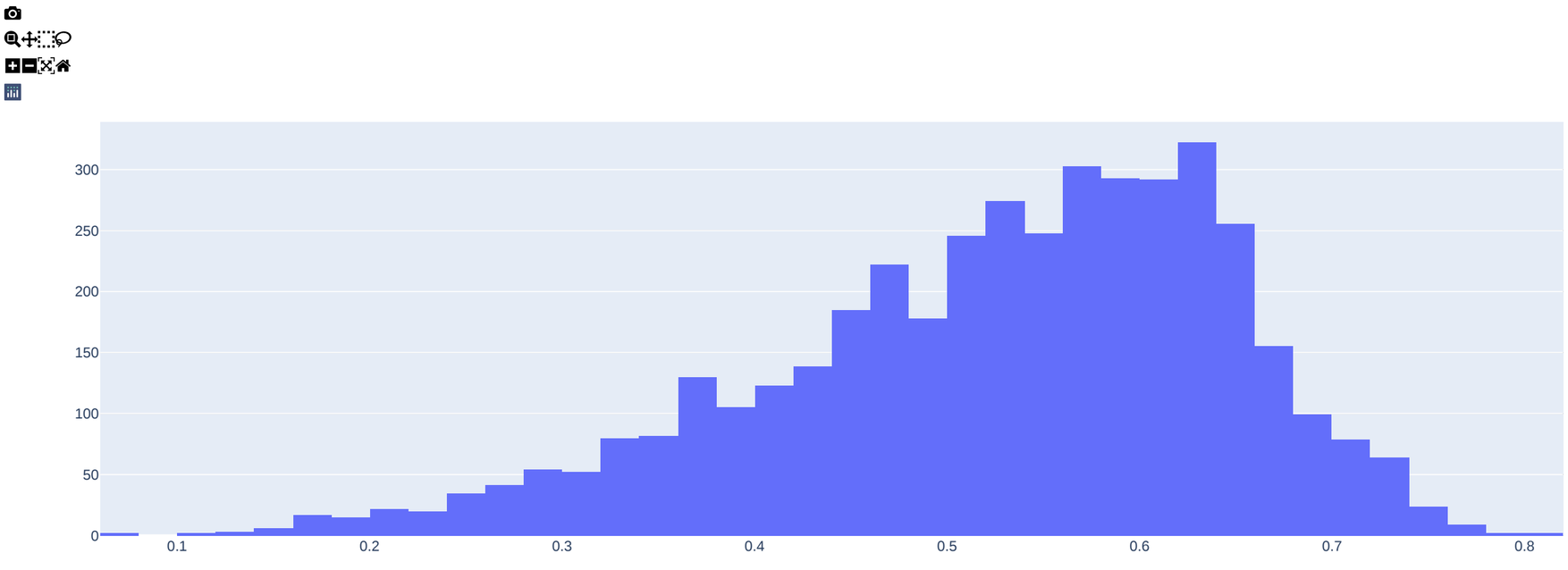
Out[2]:
```
array([0.455, 0.35 , 0.53 , ..., 0.6  , 0.625, 0.71 ])
```

In [3]:
```
data = [go.Histogram(x = length_data,
        name = "Data of Length",
        xbins = dict(start = 0,
                     end = 1,
                     size = .02))]
```

In [4]:
```
layout = go.Layout(title = "This is a Length Data's histogram taken from \"abalone.csv\"",
                   title_x = .5)
```

In [5]:
```
fig = go.Figure(data, layout)
```

In [6]:
```
pyo.iplot(fig)
```



In [7]:
```
pyo.plot(fig, filename = "tutorial_17 (Histograms Exercise){Graph}.html")
```

Out[7]:
```
'tutorial_17 (Histograms Exercise){Graph}.html'
```

Date 06-07-2021

Distplots

In [1]:
```
import pandas as pd
import numpy as np
import plotly.offline as pyo
import plotly.figure_factory as ff
```

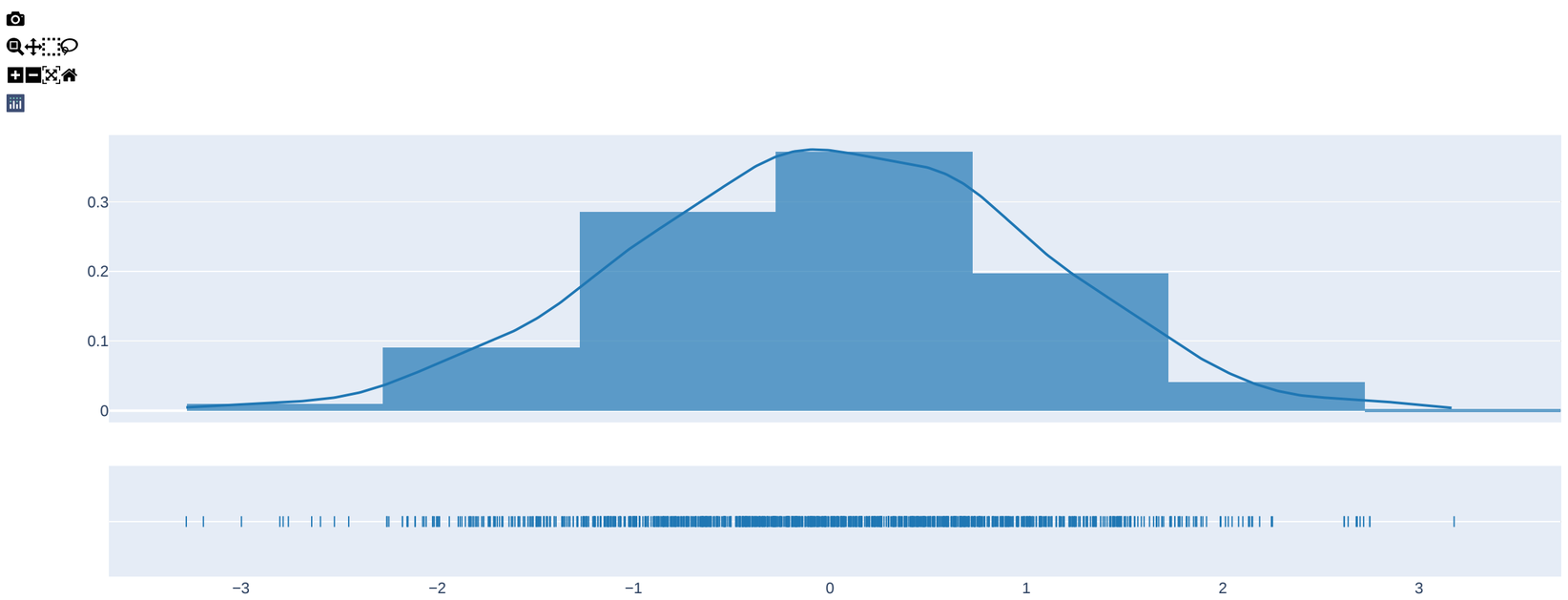
In [2]:
```
np.random.seed(5465)
x = np.random.randn(1000)
```

In [3]:
```
hist_data = [x]
```

In [4]:
```
group_labels = ["Distance Plots"]
```

In [5]:
```
fig = ff.create_distplot(hist_data,
                         group_labels)
```

In [6]:
```
pyo.iplot(fig)
```



In [7]:
```
pyo.plot(fig, filename = "tutorial_19 (Distplots)[Part-1]{Graph}.html")
```

Out[7]:
```
'tutorial_19 (Distplots)[Part-1]{Graph}.html'
```

In [8]:
```
x1 = np.random.randn(1000) - 2
x2 = np.random.randn(1000) + 2
x3 = np.random.randn(1000) - 4
x4 = np.random.randn(1000) + 4
x5 = np.random.randn(1000) + 6
x6 = np.random.randn(1000) - 6
x7 = np.random.randn(1000)
```

In [9]:
```
hist_data = [x1, x2, x3, x4, x5, x6, x7]
```

In [10]:
```
group_labels = ["X1", "X2", "X3", "X4", "X5", "X6", "X7"]
```

In [11]:
```
fig = ff.create_distplot(hist_data,
                         group_labels,
                         bin_size = np.random.uniform(.1, .9, 7).tolist())
```

In [12]:
```
pyo.iplot(fig)
```


In [13]:
```
pyo.plot(fig, filename = "tutorial_19 (Distplots)[Part-2]{Graph}.html")
```

Out[13]:
```
'tutorial_19 (Distplots)[Part-2]{Graph}.html'
```

Date 06-07-2021

Distplots Exercise

In [1]:
```
import pandas as pd
import numpy as np
import plotly.offline as pyo
import plotly.figure_factory as ff
```

In [2]:
```
iris_data_csv = pd.read_csv("iris.csv", usecols = ["petal_length", "class"])
iris_data_csv
```

Out[2]:

	petal_length	class
0	1.4	Iris-setosa
1	1.4	Iris-setosa
2	1.3	Iris-setosa
3	1.5	Iris-setosa
4	1.4	Iris-setosa
...	...	...
145	5.2	Iris-virginica
146	5.0	Iris-virginica
147	5.2	Iris-virginica
148	5.4	Iris-virginica
149	5.1	Iris-virginica

150 rows × 2 columns

In [3]:
```
iris_data_csv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   petal_length  150 non-null    float64
 1   class         150 non-null    object 
dtypes: float64(1), object(1)
memory usage: 2.5+ KB
```

In [4]:
```
iris_data_csv.describe()
```

Out[4]:

	petal_length
count	150.000000
mean	3.758667
std	1.764420
min	1.000000
25%	1.600000
50%	4.350000
75%	5.100000
max	6.900000

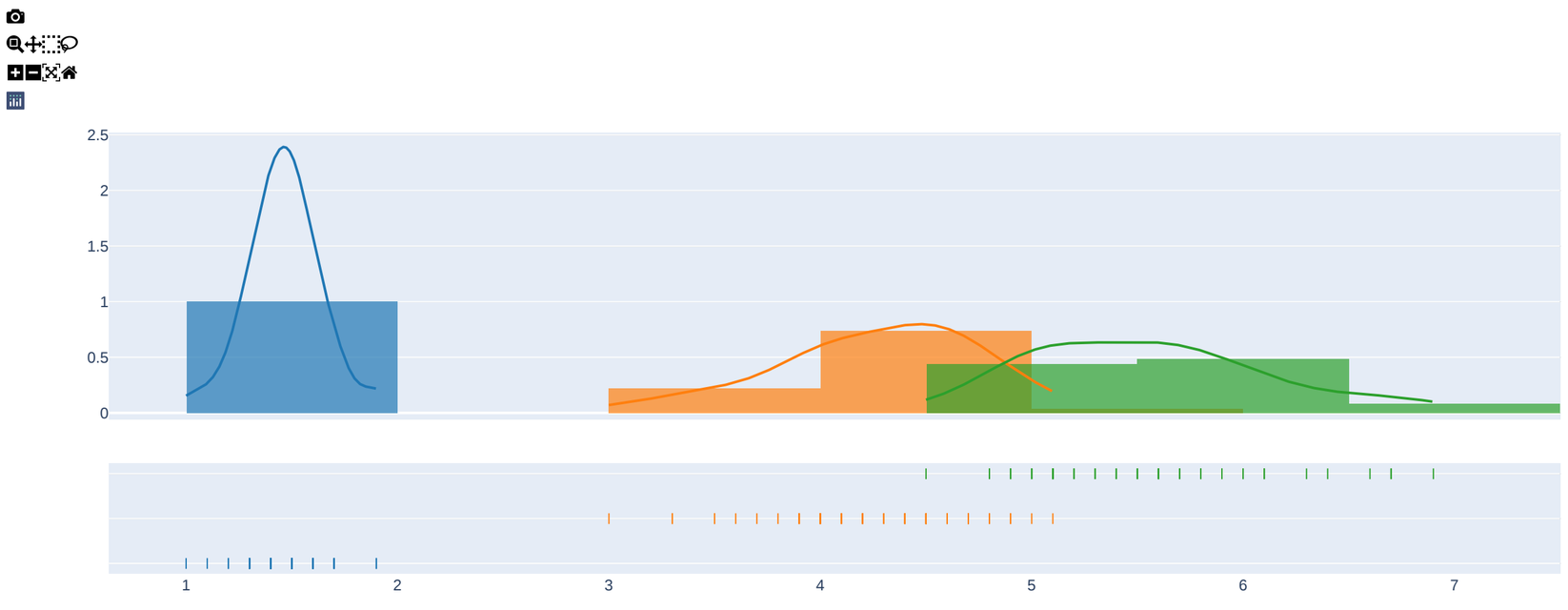
In [5]:
```
data_to_plot_1_Iris_setosa = iris_data_csv[iris_data_csv['class']=='Iris-setosa']['petal_length']
data_to_plot_2_Iris_versicolor = iris_data_csv[iris_data_csv['class']=='Iris-versicolor']['petal_length']
data_to_plot_3_Iris_virginica = iris_data_csv[iris_data_csv['class']=='Iris-virginica']['petal_length']
```

In [6]:
```
hist_data = [data_to_plot_1_Iris_setosa, data_to_plot_2_Iris_versicolor, data_to_plot_3_Iris_virginica]
```

In [7]:
```
group_labels = ["Iris Setosa", "Iris Versicolor", "Iris Virginica"]
```

In [8]:
```
fig = ff.create_distplot(hist_data,
                         group_labels)
```

In [9]:
```
pyo.iplot(fig)
```

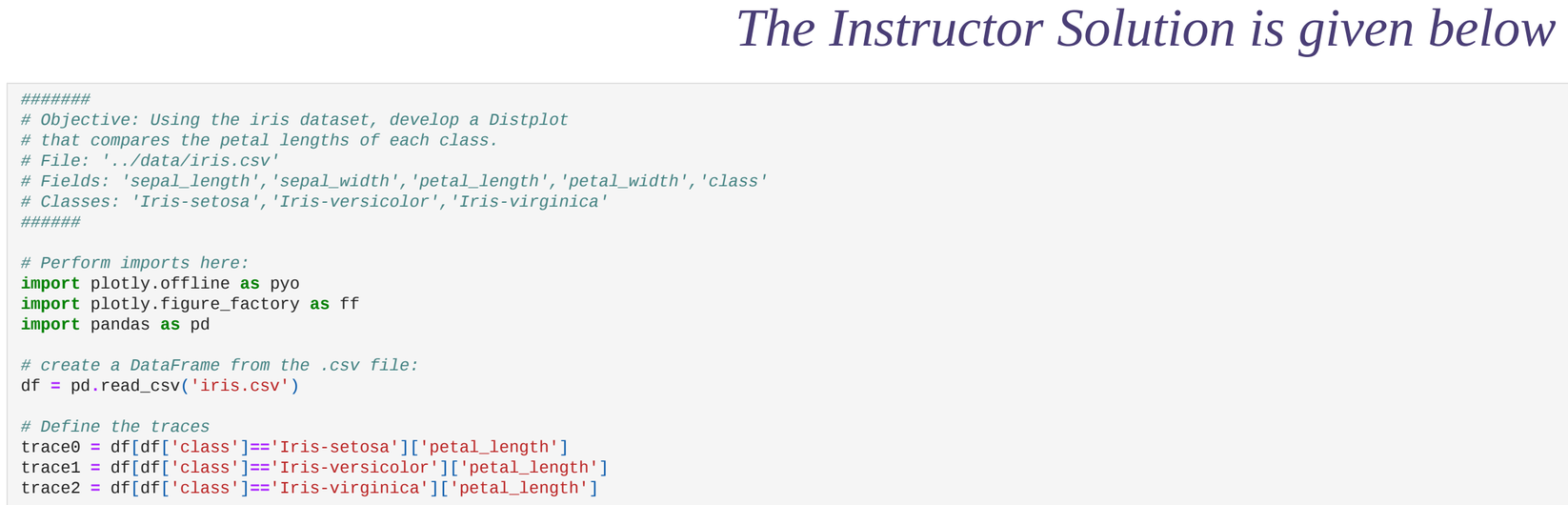


In [10]:
```
pyo.plot(fig, filename = "tutorial_20 (Distplots Exercise){Graph}.html")
```

Out[10]:
```
'tutorial_20 (Distplots Exercise){Graph}.html'
```

The Instructor Solution is given below

In [11]:
```
#######
# Objective: Using the iris dataset, develop a Distplot
# that compares the petal lengths of each class.
# File: '../data/iris.csv'
# Fields: 'sepal_length','sepal_width','petal_length','petal_width','class'
# Classes: 'Iris-setosa','Iris-versicolor','Iris-virginica'
######

# Perform imports here:
import plotly.offline as pyo
import plotly.figure_factory as ff
import pandas as pd

# create a DataFrame from the .csv file:
df = pd.read_csv('iris.csv')

# Define the traces
trace0 = df[df['class']=='Iris-setosa']['petal_length']
trace1 = df[df['class']=='Iris-versicolor']['petal_length']
trace2 = df[df['class']=='Iris-virginica']['petal_length']

# Define a data variable
hist_data = [trace0, trace1, trace2]
group_labels = ['Iris Setosa','Iris Versicolor','Iris Virginica']

# Create a fig from data and layout, and plot the fig
fig = ff.create_distplot(hist_data, group_labels)
pyo.iplot(fig)

########
# Great! This shows that if given a flower with a petal length
# between 1-2cm, it is almost certainly an Iris Setosa!
######
```



Date 06-07-2021

Heatmaps

In [1]:
```
import pandas as pd
import plotly.offline as pyo
import plotly.graph_objs as go
from plotly import subplots
```

In [2]:
```
temperature_data_csv = pd.read_csv("2010SantaBarbaraCA.csv")
temperature_data_csv
```

Out[2]:

	LST_DATE	DAY	LST_TIME	T_HR_AVG
0	20100601	TUESDAY	0:00	12.7
1	20100601	TUESDAY	1:00	12.7
2	20100601	TUESDAY	2:00	12.3
3	20100601	TUESDAY	3:00	12.5
4	20100601	TUESDAY	4:00	12.7
...	...	...	...	...
163	20100607	MONDAY	19:00	15.6
164	20100607	MONDAY	20:00	14.8
165	20100607	MONDAY	21:00	14.3
166	20100607	MONDAY	22:00	14.4
167	20100607	MONDAY	23:00	14.6

168 rows × 4 columns

In [3]:
```
temperature_data_csv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 168 entries, 0 to 167
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   LST_DATE    168 non-null    int64 
 1   DAY         168 non-null    object
 2   LST_TIME    168 non-null    object
 3   T_HR_AVG    168 non-null    float64
dtypes: float64(1), int64(1), object(2)
memory usage: 5.4+ KB
```

In [4]:
```
temperature_data_csv.describe()
```

Out[4]:

	LST_DATE	T_HR_AVG
count	1.680000e+02	168.000000
mean	2.010060e+07	14.645833
std	2.005979e+00	1.631569
min	2.010060e+07	10.300000
25%	2.010060e+07	13.700000
50%	2.010060e+07	14.900000
75%	2.010061e+07	15.625000
max	2.010061e+07	17.700000

In [5]:
```
temperature_data_csv.columns
```

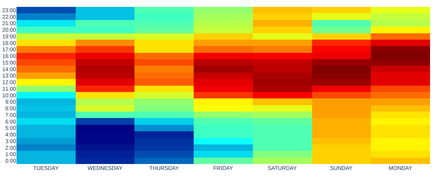
Out[5]:
```
Index(['LST_DATE', 'DAY', 'LST_TIME', 'T_HR_AVG'], dtype='object')
```

In [6]:
```
data = [go.Heatmap(x = temperature_data_csv["DAY"],
                   y = temperature_data_csv["LST_TIME"],
                   z = temperature_data_csv["T_HR_AVG"].values.tolist(),
                   colorbar=dict(title="Temperature"),
                   colorscale = "jet")]
```

In [7]:
```
layout = go.Layout(title = "HeatMap of Temperature in Santa Barbara, California", title_x = 0.5)
```

In [8]:
```
fig = go.Figure(data, layout)
```

In [9]:
```
fig.show()
```



In [10]:
```
pyo.plot(fig, filename = "tutorial_22 (Heatmaps)[Part-1](Graph).html")
```

Out[10]:
```
'tutorial_22 (Heatmaps)[Part-1](Graph).html'
```

In [11]:
```
temperature_data_csv = pd.read_csv("2010YumaAZ.csv")
temperature_data_csv
```

Out[11]:

	LST_DATE	DAY	LST_TIME	T_HR_AVG
0	20100601	TUESDAY	0:00	25.2
1	20100601	TUESDAY	1:00	24.1
2	20100601	TUESDAY	2:00	24.4
3	20100601	TUESDAY	3:00	24.9
4	20100601	TUESDAY	4:00	22.8
...	...	...	...	...
163	20100607	MONDAY	19:00	39.4
164	20100607	MONDAY	20:00	38.5
165	20100607	MONDAY	21:00	37.0
166	20100607	MONDAY	22:00	34.7
167	20100607	MONDAY	23:00	32.6

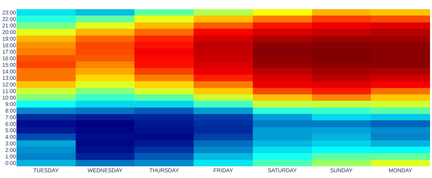
168 rows × 4 columns

In [12]:
```
data = [go.Heatmap(x = temperature_data_csv["DAY"],
                   y = temperature_data_csv["LST_TIME"],
                   z = temperature_data_csv["T_HR_AVG"].values.tolist(),
                   colorbar=dict(title="Temperature"),
                   colorscale = "jet")]
```

In [13]:
```
layout = go.Layout(title = "HeatMap of Temperature in Yuma, Arizona", title_x = 0.5)
```

In [14]:
```
fig = go.Figure(data, layout)
```

In [15]:
```
fig.show()
```



In [16]:
```
pyo.plot(fig, filename = "tutorial_22 (Heatmaps)[Part-2](Graph).html")
```

Out[16]:
```
'tutorial_22 (Heatmaps)[Part-2](Graph).html'
```

Mutiple Heatmaps Using SubPlots

In [17]:
```
temperature_data_1_of_santa_barbara_in_california_csv = pd.read_csv("2010SantaBarbaraCA.csv")
temperature_data_2_of_yuma_in_arizona_csv = pd.read_csv("2010YumaAZ.csv")
temperature_data_3_of_sitka_in_alaska_csv = pd.read_csv("2010SitkaAK.csv")
```

In [18]:
```
trace_1 = go.Heatmap(go.Heatmap(x = temperature_data_1_of_santa_barbara_in_california_csv["DAY"],
                                y = temperature_data_1_of_santa_barbara_in_california_csv["LST_TIME"],
                                z = temperature_data_1_of_santa_barbara_in_california_csv["T_HR_AVG"].values.tolist(),
                                zmin = 5,
                                zmax = 40,
                                colorbar=dict(title="Temperature"),
                                colorscale = "jet"))
```

In [19]:
```
trace_2 = go.Heatmap(go.Heatmap(x = temperature_data_2_of_yuma_in_arizona_csv["DAY"],
                                y = temperature_data_2_of_yuma_in_arizona_csv["LST_TIME"],
                                z = temperature_data_2_of_yuma_in_arizona_csv["T_HR_AVG"].values.tolist(),
                                zmin = 5,
                                zmax = 40,
                                colorbar=dict(title="Temperature"),
                                colorscale = "jet"))
```

In [20]:
```
trace_3 = go.Heatmap(go.Heatmap(x = temperature_data_3_of_sitka_in_alaska_csv["DAY"],
                                y = temperature_data_3_of_sitka_in_alaska_csv["LST_TIME"],
                                z = temperature_data_3_of_sitka_in_alaska_csv["T_HR_AVG"].values.tolist(),
                                zmin = 5,
                                zmax = 40,
                                colorbar=dict(title="Temperature"),
                                colorscale = "jet"))
```

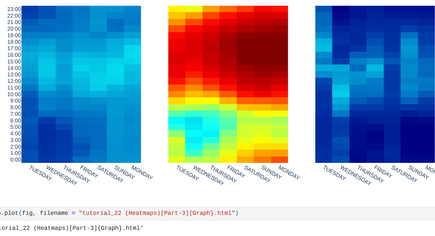
In [21]:
```
fig = subplots.make_subplots(1, 3,
                             subplot_titles = ["Santa Barbara in California", "Yuma in Arizona", "Sitka in Alaska"],
                             shared_yaxes = True)
```

In [22]:
```
fig.append_trace(trace_1, 1, 1)
fig.append_trace(trace_2, 1, 2)
fig.append_trace(trace_3, 1, 3)
```

In [23]:
```
fig['layout'].update(title = "Temperature of Diffrent Cities", title_x = 0.5)
```

Out[23]:
```
Layout({
    'annotations': [{'font': {'size': 16},
                     'showarrow': False,
                     'text': 'Santa Barbara in California',
                     'x': 0.14444444444444446,
                     'xanchor': 'center',
                     'xref': 'paper',
                     'y': 1.0,
                     'yanchor': 'bottom',
                     'yref': 'paper'},
                    {'font': {'size': 16},
                     'showarrow': False,
                     'text': 'Yuma in Arizona',
                     'x': 0.5,
                     'xanchor': 'center',
                     'xref': 'paper',
                     'y': 1.0,
                     'yanchor': 'bottom',
                     'yref': 'paper'},
                    {'font': {'size': 16},
                     'showarrow': False,
                     'text': 'Sitka in Alaska',
                     'x': 0.8555555555555556,
                     'xanchor': 'center',
                     'xref': 'paper',
                     'y': 1.0,
                     'yanchor': 'bottom',
                     'yref': 'paper'}],
    'template': '...',
    'title': {'text': 'Temperature of Diffrent Cities', 'x': 0.5},
    'xaxis': {'anchor': 'y', 'domain': [0.0, 0.2888888888888889]},
    'xaxis2': {'anchor': 'y2', 'domain': [0.35555555555555557, 0.6444444444444445]},
    'xaxis3': {'anchor': 'y3', 'domain': [0.7111111111111111, 1.0]},
    'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0]},
    'yaxis2': {'anchor': 'x2', 'domain': [0.0, 1.0], 'matches': 'y', 'showticklabels': False},
    'yaxis3': {'anchor': 'x3', 'domain': [0.0, 1.0], 'matches': 'y', 'showticklabels': False}
})
```

In [24]:
```
fig.show()
```



In [25]:
```
pyo.plot(fig, filename = "tutorial_22 (Heatmaps)[Part-3](Graph).html")
```

Out[25]:
```
'tutorial_22 (Heatmaps)[Part-3](Graph).html'
```

Date 07-07-2021

Heatmaps Exercise

In [1]:
```
import pandas as pd
import plotly.offline as pyo
import plotly.graph_objs as go
```

In [2]:
```
flights_data_csv = pd.read_csv("flights.csv")
flights_data_csv
```

Out[2]:

	year	month	passengers
0	1949	January	112
1	1949	February	118
2	1949	March	132
3	1949	April	129
4	1949	May	121
...	...	...	...
139	1960	August	606
140	1960	September	508
141	1960	October	461
142	1960	November	390
143	1960	December	432

144 rows × 3 columns

In [3]:
```
flights_data_csv.info()
```

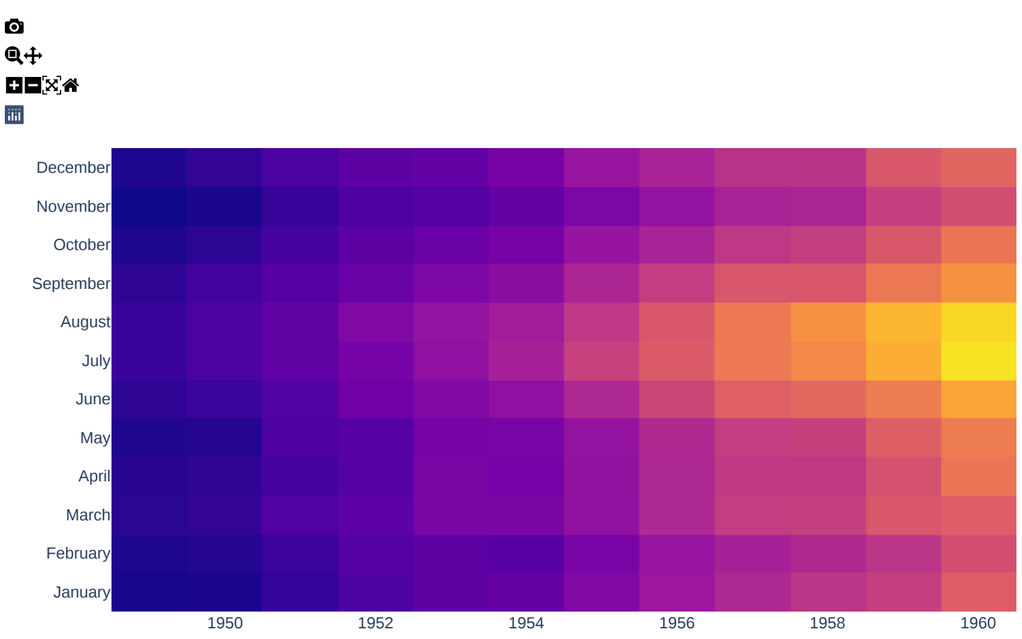
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144 entries, 0 to 143
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   year        144 non-null    int64 
 1   month       144 non-null    object
 2   passengers  144 non-null    int64 
dtypes: int64(2), object(1)
memory usage: 3.5+ KB
```

In [4]:
```
data = [go.Heatmap(x = flights_data_csv["year"],
                   y = flights_data_csv["month"],
                   z = flights_data_csv["passengers"].values.tolist(),
                   colorbar = dict(title="Passengers traveled <br>by Flights"),
                   zmin = 100,
                   zmax = 650,
                   hovertemplate='Year: %{x}<br>Month: %{y}<br>Passengers: %{z}',
                   name='Passengers<br>Traveled<br>by Flights<br>Data')]
```

In [5]:
```
layout = go.Layout(title = "Exercise of HeatMaps <br>For Showing Flights details", title_x = 0.5)
```

In [6]:
```
fig = go.Figure(data, layout)
```

In [7]:
```
fig.show()
```



In [8]:
```
pyo.plot(fig, filename = "tutorial_23 (Heatmaps Exercise){Graph}.html")
```

Out[8]:
```
'tutorial_23 (Heatmaps Exercise){Graph}.html'
```

Instructor Solution is Down Given

In [9]:
```
#######
# Objective: Using the "flights" dataset available
# from the data folder as flights.csv
# create a heatmap with the following parameters:
# x-axis="year"
# y-axis="month"
# z-axis(color)="passengers"
######

# Perform imports here:
import plotly.offline as pyo
import plotly.graph_objs as go
import pandas as pd
# Create a DataFrame from  "flights" data
df = pd.read_csv('flights.csv')

# Define a data variable
data = [go.Heatmap(
    x=df['year'],
    y=df['month'],
    z=df['passengers']
)]

# Define the layout
layout = go.Layout(
    title='Flights'
)
# Create a fig from data and layout, and plot the fig
fig = go.Figure(data=data, layout=layout)
pyo.iplot(fig)

#######
# Excellent! This shows two distinct trends - an increase in
# passengers flying over the years, and a greater number of
# passengers flying in the summer months.
######
```