boggarapu-venkata-srujana

October 30, 2025

Infosys Springboard Virtual Internship 6.0

Project Name: BudgetWise AI-based Expense Forecasting Tool

Introduction to NumPy(Numerical Python)

[1]:
```
import numpy as np
n1=np.array([10,20,30,40])
n1
```

[1]:
```
array([10, 20, 30, 40])
```

[2]:
```
import numpy as mnp
n2=np.array([[10,20,30],[50,60,70]])
n2
```

[2]:
```
array([[10, 20, 30],
       [50, 60, 70]])
```

[3]:
```
#Initialising numpy array with zeroes
n3=np.zeros((1,2))
n3
```

[3]:
```
array([[0., 0.]])
```

[4]:
```
n4=np.zeros((4,4))
n4
```

[4]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

[5]:
```
#Inialising numpy array with same number
n5=np.full((3,3),11)
n5
```

[5]:
```
array([[11, 11, 11],
       [11, 11, 11],
```

```
       [11, 11, 11]])
```

[6]:
```
#Iniatialising numpy array to be within a range
n6=np.arange(1,10)
n6
```

[6]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[7]:
```
n7=np.arange(1,50,10) #Increments by 10
n7
```

[7]:
```
array([ 1, 11, 21, 31, 41])
```

[8]:
```
#Initialising numpy array with random numbers
n8=np.random.randint(1,100,10)
n8
```

[8]:
```
array([40, 78, 41, 79, 95, 65, 28, 15, 94, 43])
```

[9]:
```
#Checking the shape of Numpy arrays
n9=np.array([[1,2,3],[4,5,6],[7,8,9],[10,1,19]])
n9.shape
```

[9]:
```
(4, 3)
```

[10]:
```
#Joining Numpy array using Vertical stack(Both arrays will be placed one after␣
 ↪the other)
n10=np.array([1,2,3])
n11=np.array([4,5,6])
np.vstack((n10,n11))
```

[10]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

[11]:
```
#Joining Numpy array using Horizontal stack(Both arrays will be placed side by␣
 ↪side)
n12=np.array([1,2,3])
n13=np.array([4,5,6])
np.hstack((n12,n13))
```

[11]:
```
array([1, 2, 3, 4, 5, 6])
```

[12]:
```
#Joining Numpy array using Column stack(Here each row will be converted into a␣
 ↪column)
n14=np.array([1,2,3])
n15=np.array([4,5,6])
np.column_stack((n14,n15))
```

[12]:
```
array([[1, 4],
       [2, 5],
       [3, 6]])
```

[13]:
```
#Intersection and Differences of Numpy arrays
n16=np.array([1,2,3,4,5])
n17=np.array([4,5,6,7])
np.intersect1d(n16,n17)
```

[13]:
```
array([4, 5])
```

[14]:
```
#Gives elements in array A that are not in array B
np.setdiff1d(n16,n17)
```

[14]:
```
array([1, 2, 3])
```

[15]:
```
np.setdiff1d(n17,n16)
```

[15]:
```
array([6, 7])
```

[16]:
```
#Addition of Numpy arrays
n18=np.array([2,3])
n19=np.array([4,5])
np.sum([n18,n19])
```

[16]:
```
14
```

[17]:
```
#Sum of elements along the axis
np.sum([n18,n19],axis=0)
```

[17]:
```
array([6, 8])
```

[18]:
```
np.sum([n18,n19],axis=1)
```

[18]:
```
array([5, 9])
```

[19]:
```
n20=np.array([10,20,20])
#Basic Addition
n1=n1+1
n1
```

[19]:
```
array([11, 21, 31, 41])
```

[20]:
```
#Basic Subtraction
n1=n1-1
n1
```

[20]:
```
array([10, 20, 30, 40])
```

[21]:
```
#Multiplication
n1=n1*10
n1
```

[21]:
```
array([100, 200, 300, 400])
```

[22]:
```
#Division
n1=n1/2
n1
```

[22]:
```
array([ 50., 100., 150., 200.])
```

[23]:
```
#Finding Mean
np.mean(n1)
```

[23]:
```
125.0
```

[24]:
```
#Finding Standard Deviation
np.std(n1)
```

[24]:
```
55.90169943749474
```

[25]:
```
#Finding Median
np.median(n1)
```

[25]:
```
125.0
```

[26]:
```
#Forming a matrix using NumPy array
n21=np.array([[1,2,3],[4,5,6],[7,8,9]])
n21
```

[26]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

[27]:
```
#For printing Rows
n21[0]
```

[27]:
```
array([1, 2, 3])
```

[28]:
```
#For printing Columns
n21[:,1]
```

[28]:
```
array([2, 5, 8])
```

[29]:
```
n21[:,2]
```

[29]:
```
array([3, 6, 9])
```

[30]:
```
#Extracting single element
import pandas as pd
s1=pd.Series([1,2,3,4,5,6,7,8,9])
s1[3]
```

[30]:
```
4
```

[31]:
```
#Extracting elements from back
s1[-3:]
```

[31]:
```
6    7
7    8
8    9
dtype: int64
```

[32]:
```
#Extraacting a sequence of elements
s1[:4]
```

[32]:
```
0    1
1    2
2    3
3    4
dtype: int64
```

Pandas

[33]:
```
import pandas as pd
ds=pd.read_csv('Iris.csv')
```

[34]:
```
#Gives first 5 rows
ds.head()
```

[34]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

[35]:
```
#Gives first 10 rows
ds.head(10)
```

[35]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

[36]:
```
#Gives last 5 rows
ds.tail()
```

[36]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

[37]:
```
#Gives last 10 rows
ds.tail(10)
```

[37]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
140	141	6.7	3.1	5.6	2.4	Iris-virginica
141	142	6.9	3.1	5.1	2.3	Iris-virginica
142	143	5.8	2.7	5.1	1.9	Iris-virginica
143	144	6.8	3.2	5.9	2.3	Iris-virginica
144	145	6.7	3.3	5.7	2.5	Iris-virginica
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

```
142  Iris-virginica
143  Iris-virginica
144  Iris-virginica
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica
```

[38]:
```
#Gives (no.of rows,no.of columns)
ds.shape
```

[38]:
```
(150, 6)
```

[39]:
```
ds.describe()
```

[39]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

[40]:
```
ds.iloc[0:3,0:2]
```

[40]:

	Id	SepalLengthCm
0	1	5.1
1	2	4.9
2	3	4.7

[41]:
```
ds.iloc[0:15,0:5]
```

[41]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	1	5.1	3.5	1.4	0.2
1	2	4.9	3.0	1.4	0.2
2	3	4.7	3.2	1.3	0.2
3	4	4.6	3.1	1.5	0.2
4	5	5.0	3.6	1.4	0.2
5	6	5.4	3.9	1.7	0.4
6	7	4.6	3.4	1.4	0.3
7	8	5.0	3.4	1.5	0.2
8	9	4.4	2.9	1.4	0.2
9	10	4.9	3.1	1.5	0.1
10	11	5.4	3.7	1.5	0.2

```
11  12            4.8           3.4            1.6           0.2
12  13            4.8           3.0            1.4           0.1
13  14            4.3           3.0            1.1           0.1
14  15            5.8           4.0            1.2           0.2
```

[42]:
```
ds.loc[0:3,("SepalLengthCm","PetalLengthCm")]
```

[42]:
```
   SepalLengthCm  PetalLengthCm
0            5.1            1.4
1            4.9            1.4
2            4.7            1.3
3            4.6            1.5
```

[43]:
```
#to drop a column
ds.drop('SepalLengthCm',axis=1)
```

[43]:
```
      Id  SepalWidthCm  PetalLengthCm  PetalWidthCm         Species
0      1           3.5            1.4           0.2     Iris-setosa
1      2           3.0            1.4           0.2     Iris-setosa
2      3           3.2            1.3           0.2     Iris-setosa
3      4           3.1            1.5           0.2     Iris-setosa
4      5           3.6            1.4           0.2     Iris-setosa
..   ...           ...            ...           ...             ...
145  146           3.0            5.2           2.3  Iris-virginica
146  147           2.5            5.0           1.9  Iris-virginica
147  148           3.0            5.2           2.0  Iris-virginica
148  149           3.4            5.4           2.3  Iris-virginica
149  150           3.0            5.1           1.8  Iris-virginica

[150 rows x 5 columns]
```

[44]:
```
#to drop specific columns
ds.drop([1,2,3],axis=0)
```

[44]:
```
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
4      5            5.0           3.6            1.4           0.2
5      6            5.4           3.9            1.7           0.4
6      7            4.6           3.4            1.4           0.3
7      8            5.0           3.4            1.5           0.2
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8
```

```
            Species
0       Iris-setosa
4       Iris-setosa
5       Iris-setosa
6       Iris-setosa
7       Iris-setosa
..              …
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[147 rows x 6 columns]
```

[45]:
```
#Gives mean value
ds.mean
```

[45]:
```
<bound method DataFrame.mean of       Id  SepalLengthCm  SepalWidthCm
PetalLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   …            …             …              …             …
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              …
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]>
```

[46]:
```
#Gives minimum value in each column
ds.min()
```

[46]:
```
Id                        1
SepalLengthCm           4.3
SepalWidthCm            2.0
PetalLengthCm           1.0
PetalWidthCm            0.1
Species         Iris-setosa
dtype: object
```

[47]:
```
#Gives maximum value in each column
ds.max()
```

[47]:
```
Id                          150
SepalLengthCm               7.9
SepalWidthCm                4.4
PetalLengthCm               6.9
PetalWidthCm                2.5
Species          Iris-virginica
dtype: object
```

[48]:
```
#Gives median
ds.median
```

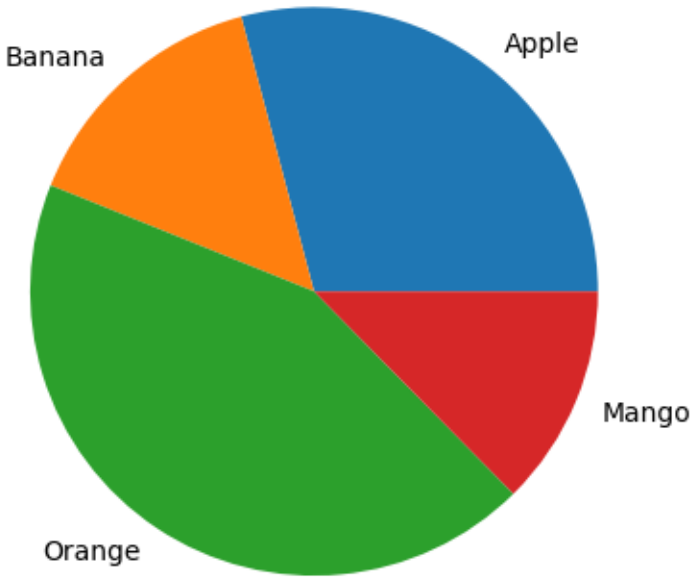
[48]:
```
<bound method DataFrame.median of       Id  SepalLengthCm  SepalWidthCm
PetalLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   …            …             …              …             …
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

              Species
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
..                …
145  Iris-virginica
```

```
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]>
```

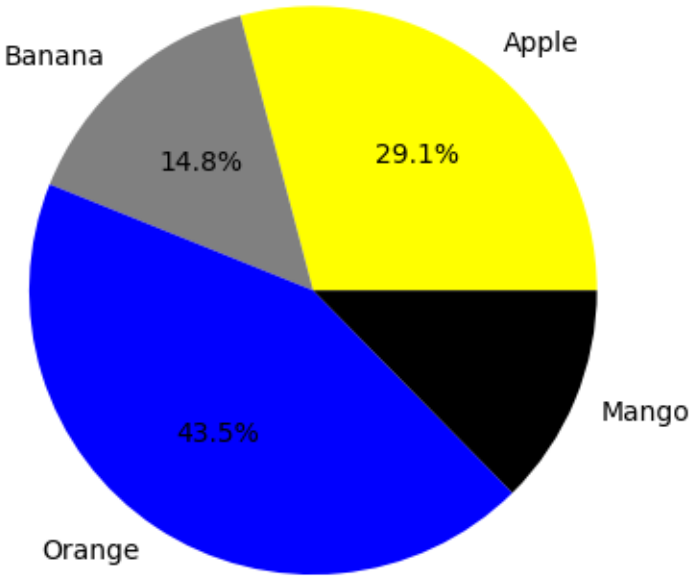
Matplotlib

[49]:
```
from matplotlib import pyplot as plt
```

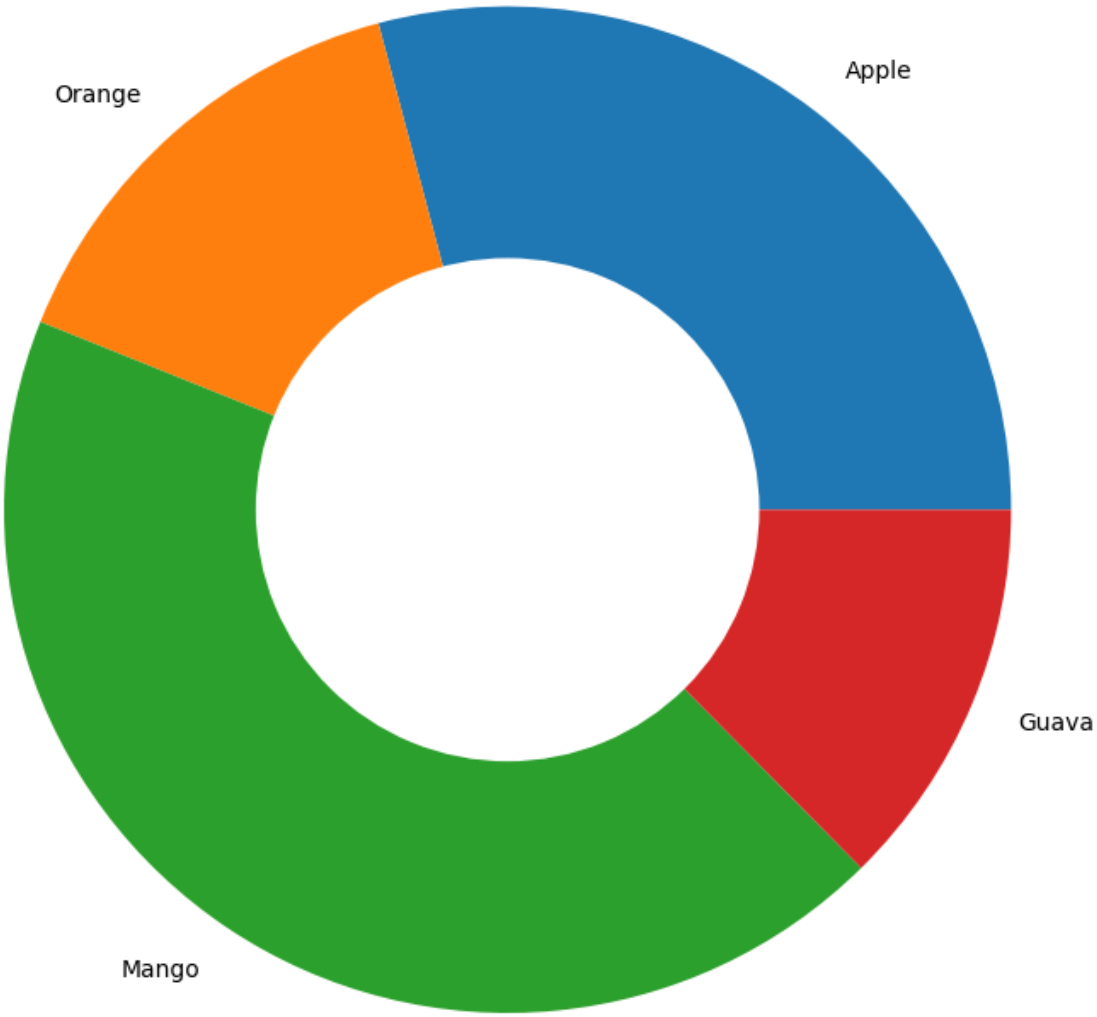
[50]:
```
#Creating a pie chart
fruit=['Apple','Banana','Orange','Mango']
quantity=[67,34,100,29]
plt.pie(quantity,labels=fruit)
plt.show()
```



[51]:
```
#Creating a pie chart with wanted colours
plt.pie(quantity,labels=fruit,autopct='%0.
  ↪1f%%',colors=['yellow','grey','blue','black'])
plt.show()
```

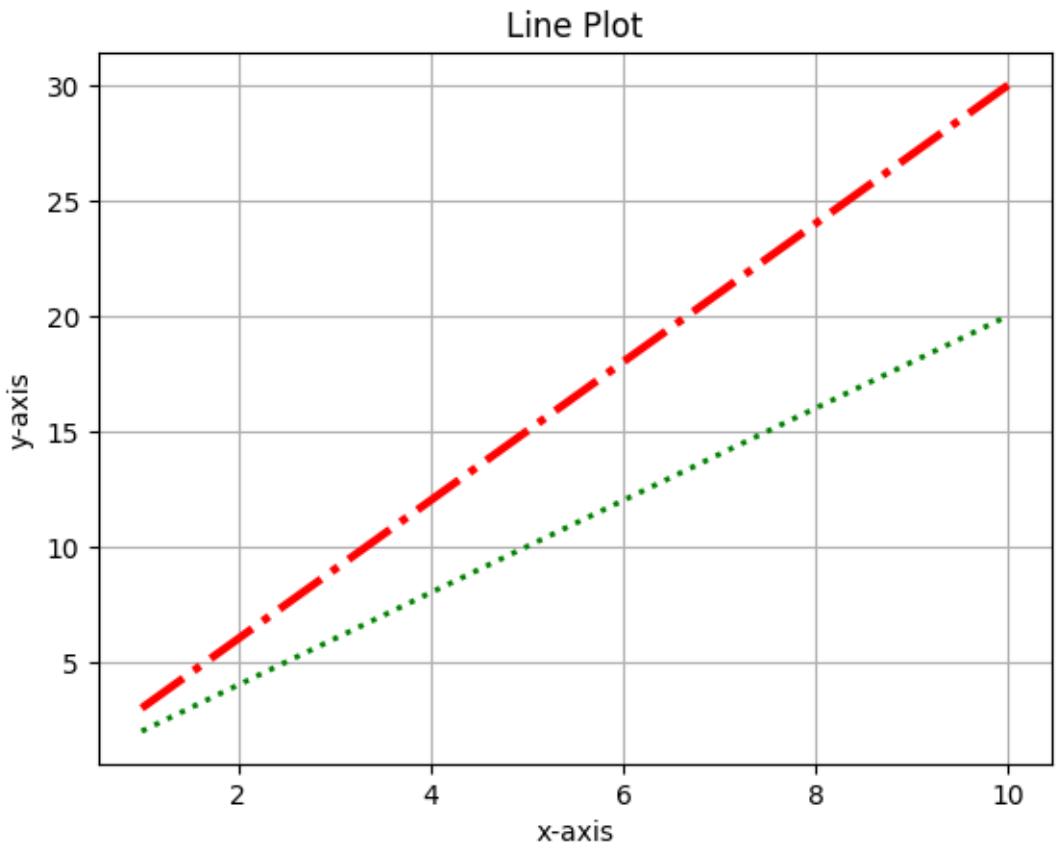


[52]:
```
#Doughnut chart
fruit=['Apple','Orange','Mango','Guava']
quantity=[67,34,100,29]
plt.pie(quantity,labels=fruit,radius=2)
plt.pie([1],colors=['w'],radius=1)
plt.show()
```

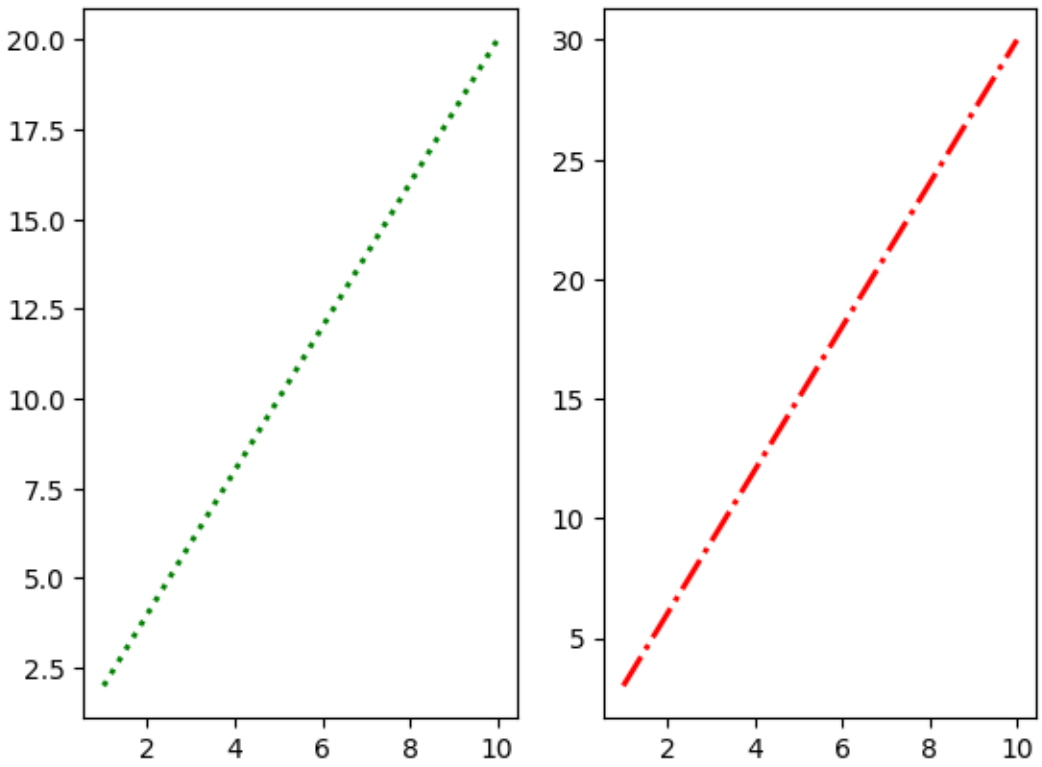


[53]:
```
#Line plot
x=np.arange(1,11)
y1=2*x
y2=3*x
```

[54]:
```
#Line Plot example(Adding two lines in the same plot)
plt.plot(x,y1,color='g',linestyle=':',linewidth='2')
plt.plot(x,y2,color='r',linestyle='-.',linewidth='3')
plt.title("Line Plot")
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.grid(True)
plt.show()
```

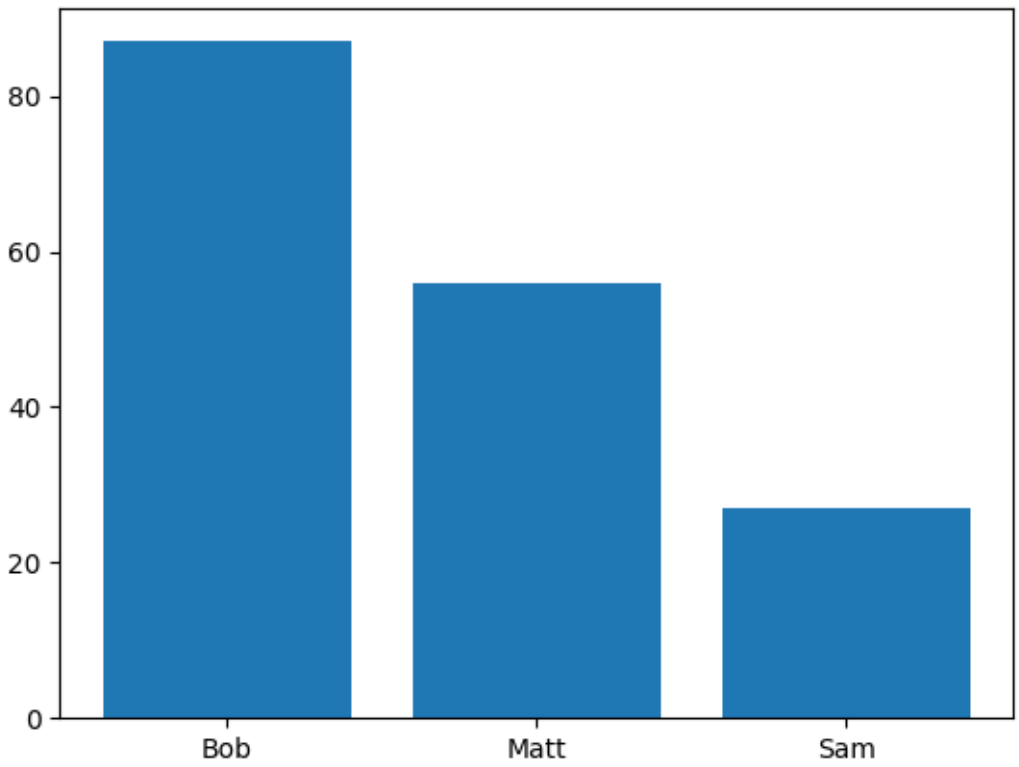


[55]:
```
#Adding sub-plots
y1=2*x
y2-3*x

plt.subplot(1,2,1)
plt.plot(x,y1,color='g',linestyle=':',linewidth=2)

plt.subplot(1,2,2)
plt.plot(x,y2,color='r',linestyle='-.',linewidth=2)

plt.show()
```

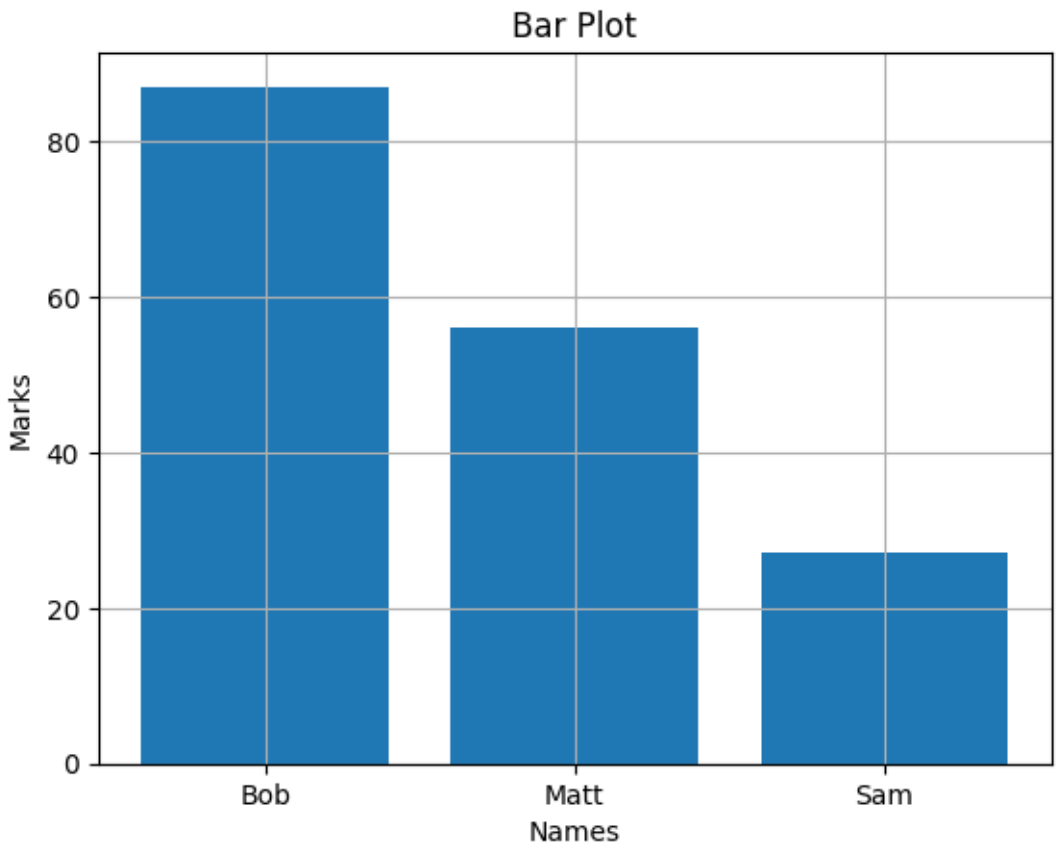


[56]:
```
#Bar plot
student={"Bob":87,"Matt":56,"Sam":27}
names=list(student.keys())
values=list(student.values())
```

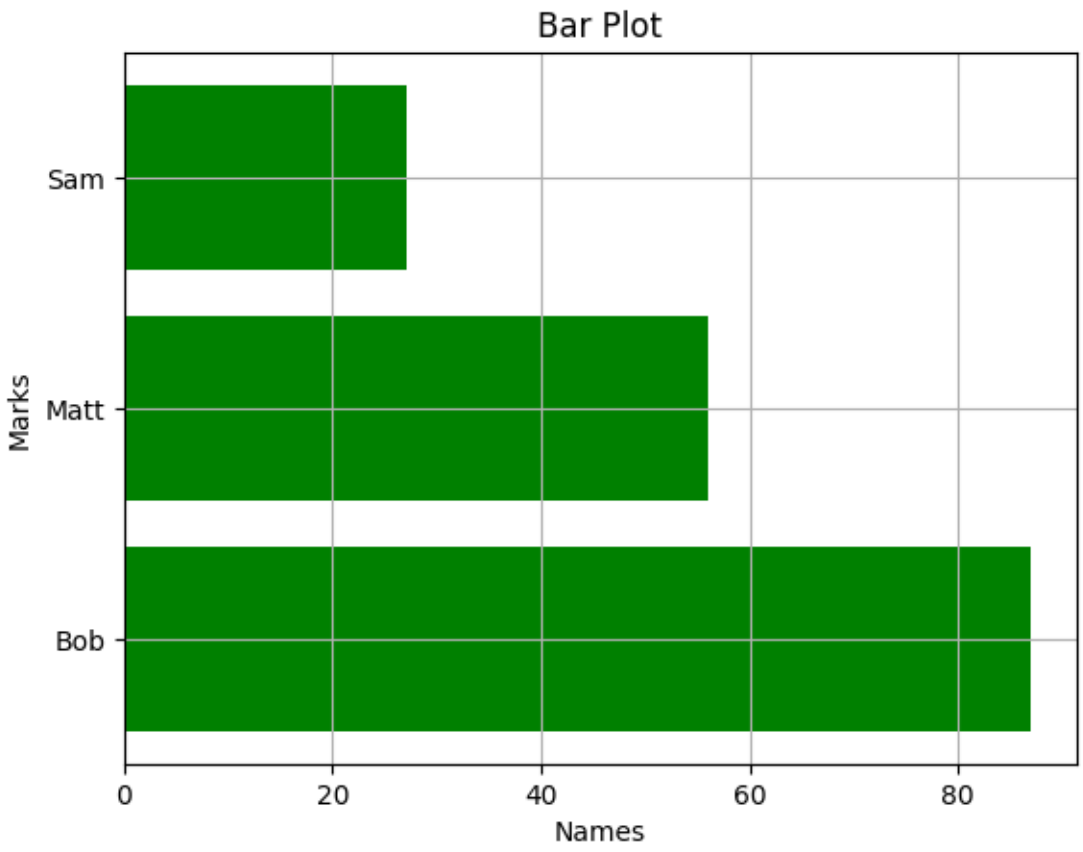
[57]:
```
plt.bar(names,values)
plt.show()
```



[58]:
```
#Adding titles and labels
plt.bar(names,values)
plt.title("Bar Plot")
plt.xlabel("Names")
plt.ylabel("Marks")
plt.grid(True)
plt.show()
```



[59]:
```
#Horizontal Bar plot
plt.barh(names,values,color='g')
plt.title("Bar Plot")
plt.xlabel("Names")
plt.ylabel("Marks")
plt.grid(True)
plt.show()
```



Seaborn Library

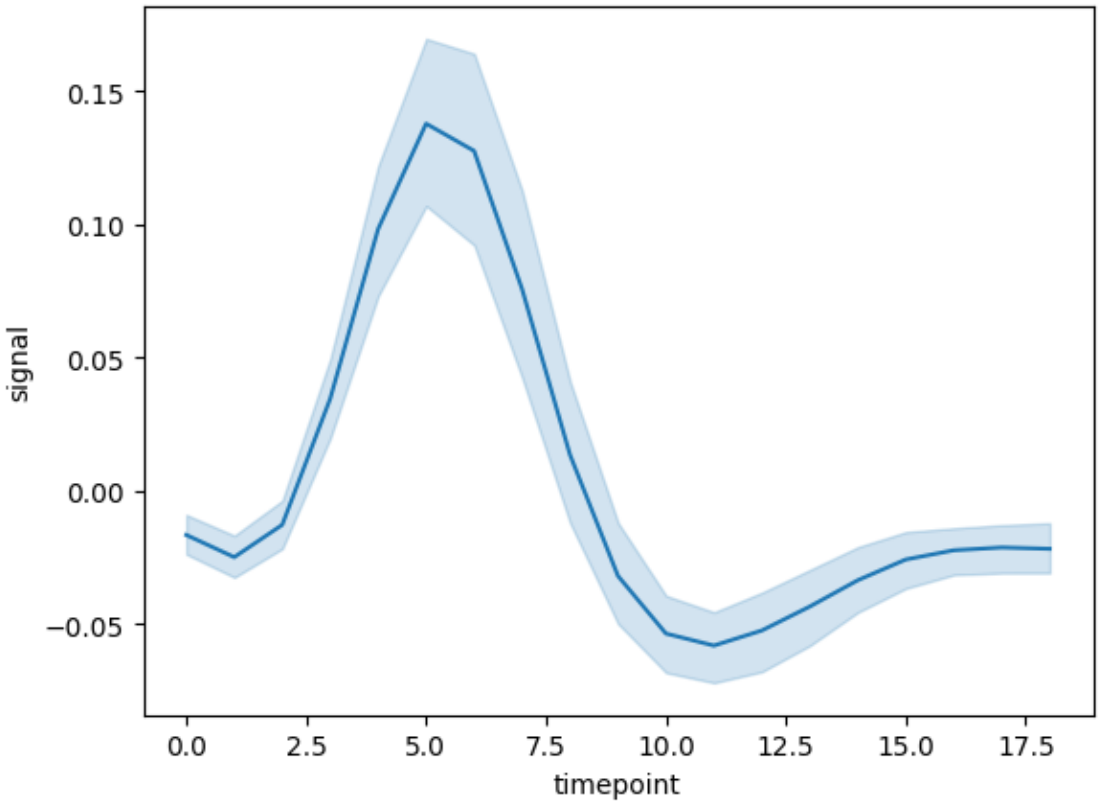
[60]:
```
import seaborn as sns
```

[61]:
```
#Loading a dataset and printing the first 5 rows using head()
fmri=sns.load_dataset('fmri')
fmri.head()
```

[61]:

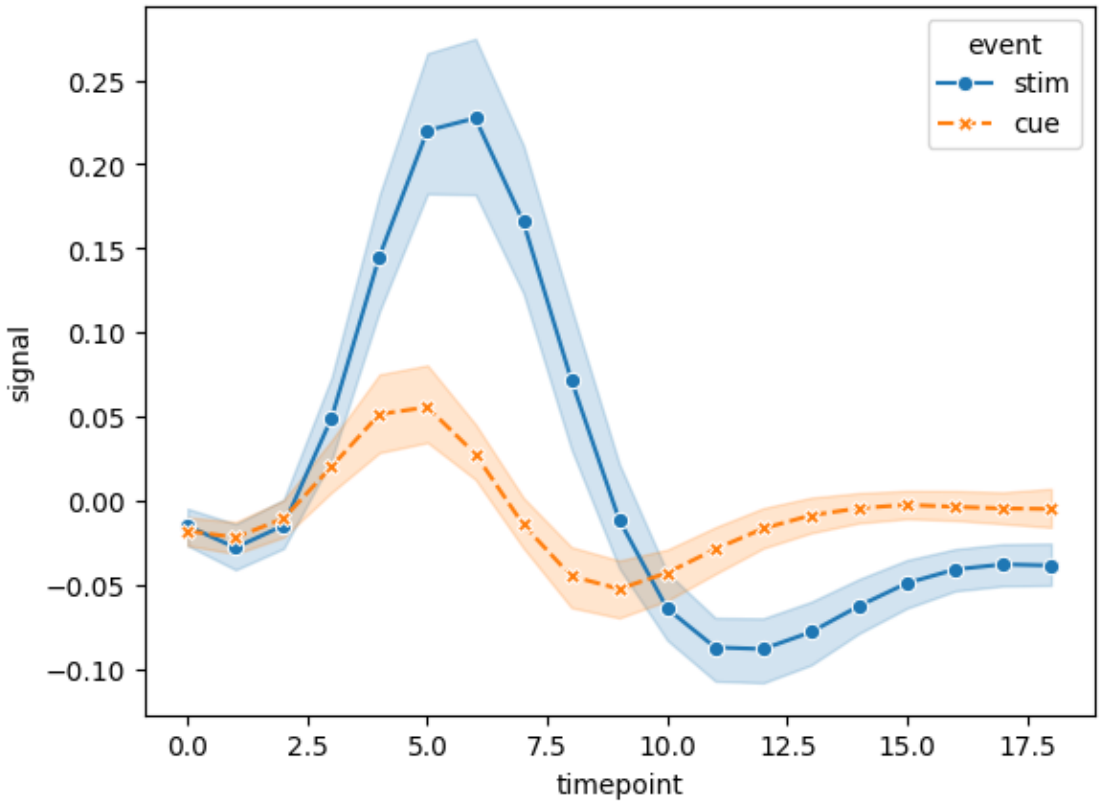
	subject	timepoint	event	region	signal
0	s13	18	stim	parietal	-0.017552
1	s5	14	stim	parietal	-0.080883
2	s12	18	stim	parietal	-0.081033
3	s11	18	stim	parietal	-0.046134
4	s10	18	stim	parietal	-0.037970

[62]:
```
#making a line plot with Timepoint as X-axis and Signal as Y-axis for the␣
↪loaded dataset
sns.lineplot(x="timepoint",y="signal",data=fmri)
plt.show()
```



[63]:
```
#hue means conditions
#Working with hue
sns.
 ↪lineplot(x="timepoint",y="signal",hue="event",style="event",markers=True,data=fmri)
```

[63]: <Axes: xlabel='timepoint', ylabel='signal'>



[64]:
```
#This is a code cell where we can import all libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[65]:
```
#Loading a dataset "tips" from seaborn
ds=sns.load_dataset('tips')
```

[66]:
```
#Prints the first 5 rows
ds.head()
```

[66]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

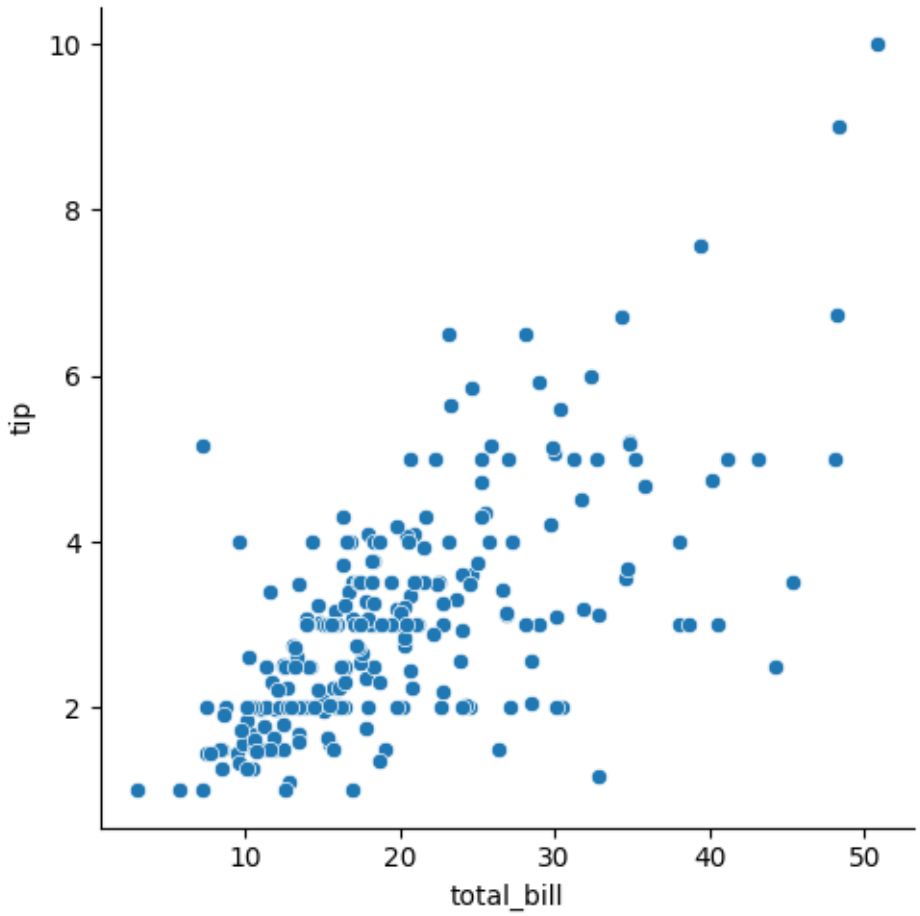
[67]:
```
#Gives a tuple that describes no.of rows and no.of columns
ds.shape
```

[67]: (244, 7)

Relative Plot

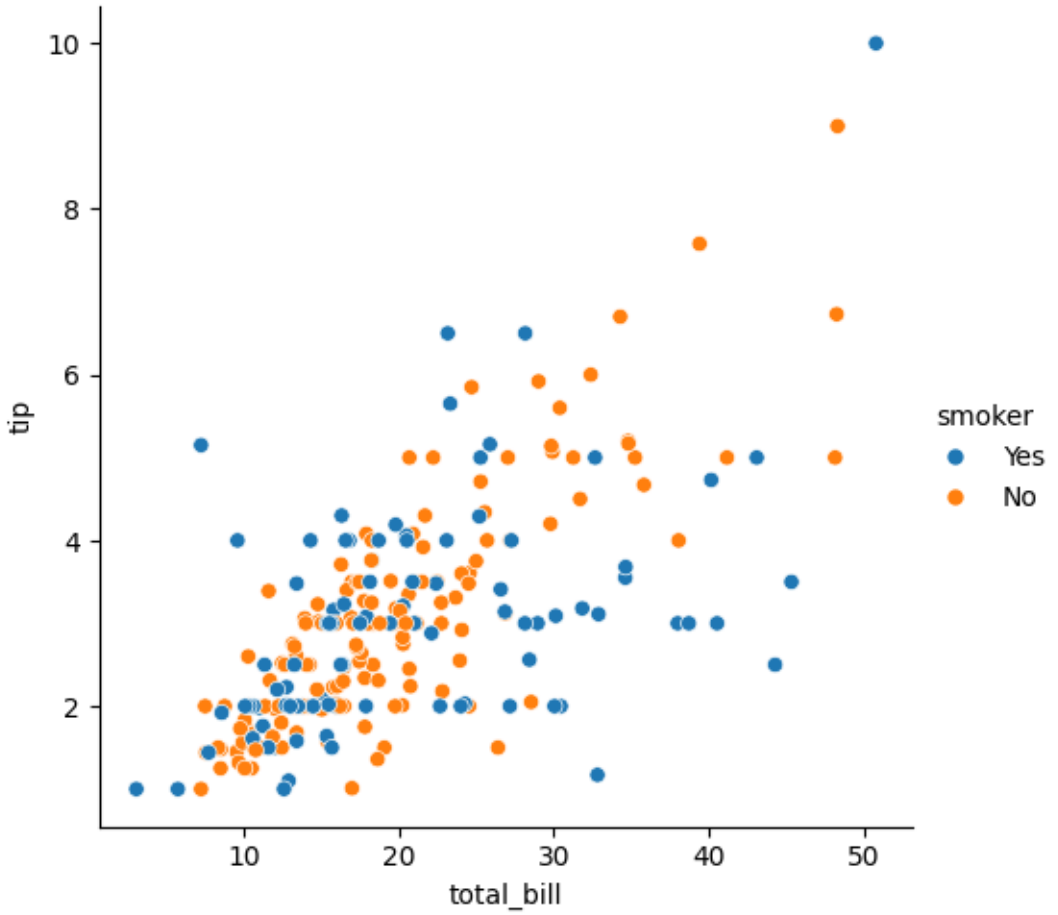
[68]:
```
#Relative plot
sns.relplot(data=ds,x='total_bill',y='tip')
```

[68]: <seaborn.axisgrid.FacetGrid at 0x193f35dae70>



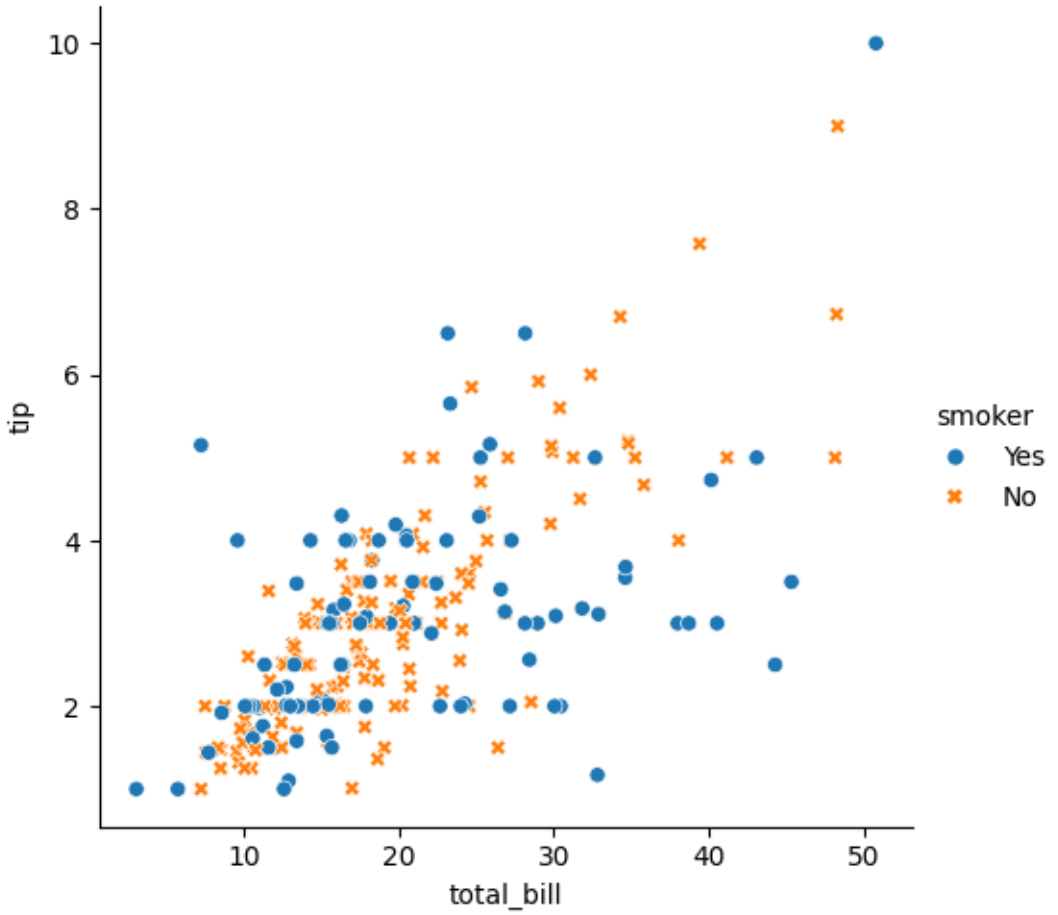
[69]:
```
#Relative plot using hue(condition)
sns.relplot(data=ds,x='total_bill',y='tip',hue='smoker')
```

[69]: <seaborn.axisgrid.FacetGrid at 0x193eea2cec0>



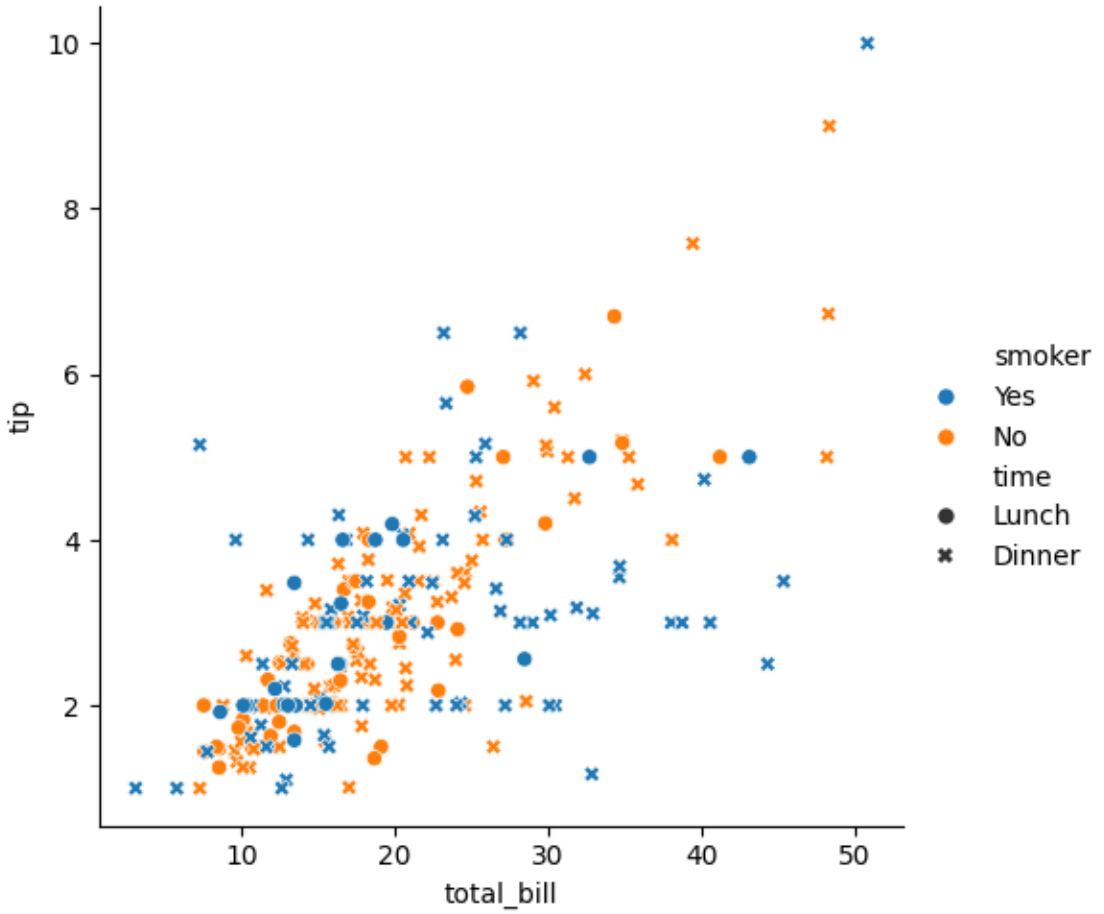
[70]:
```
#Relative plot using hue(condition) and marker
sns.relplot(data=ds,x='total_bill',y='tip',hue='smoker',style='smoker')
```

[70]: <seaborn.axisgrid.FacetGrid at 0x193eea4fc20>



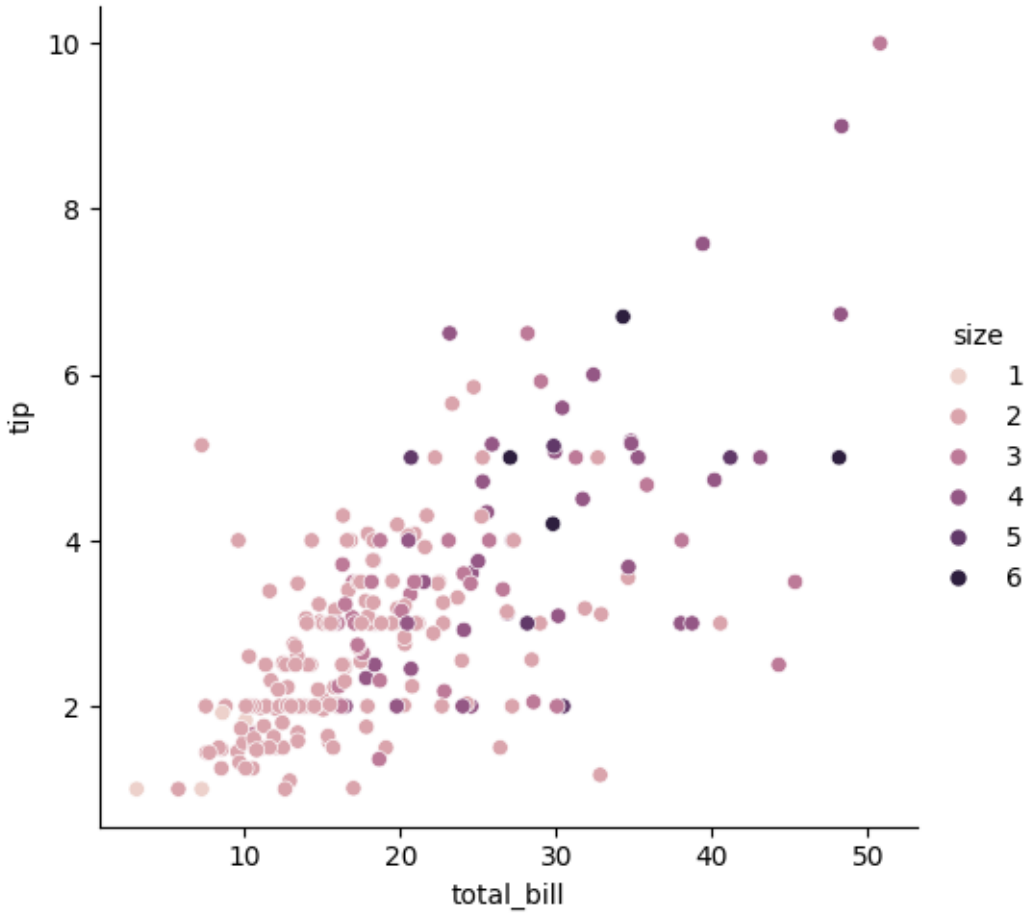
[71]:
```
#Using style
sns.relplot(data=ds,x='total_bill',y='tip',hue='smoker',style='time')
```

[71]: <seaborn.axisgrid.FacetGrid at 0x193f35ed9d0>



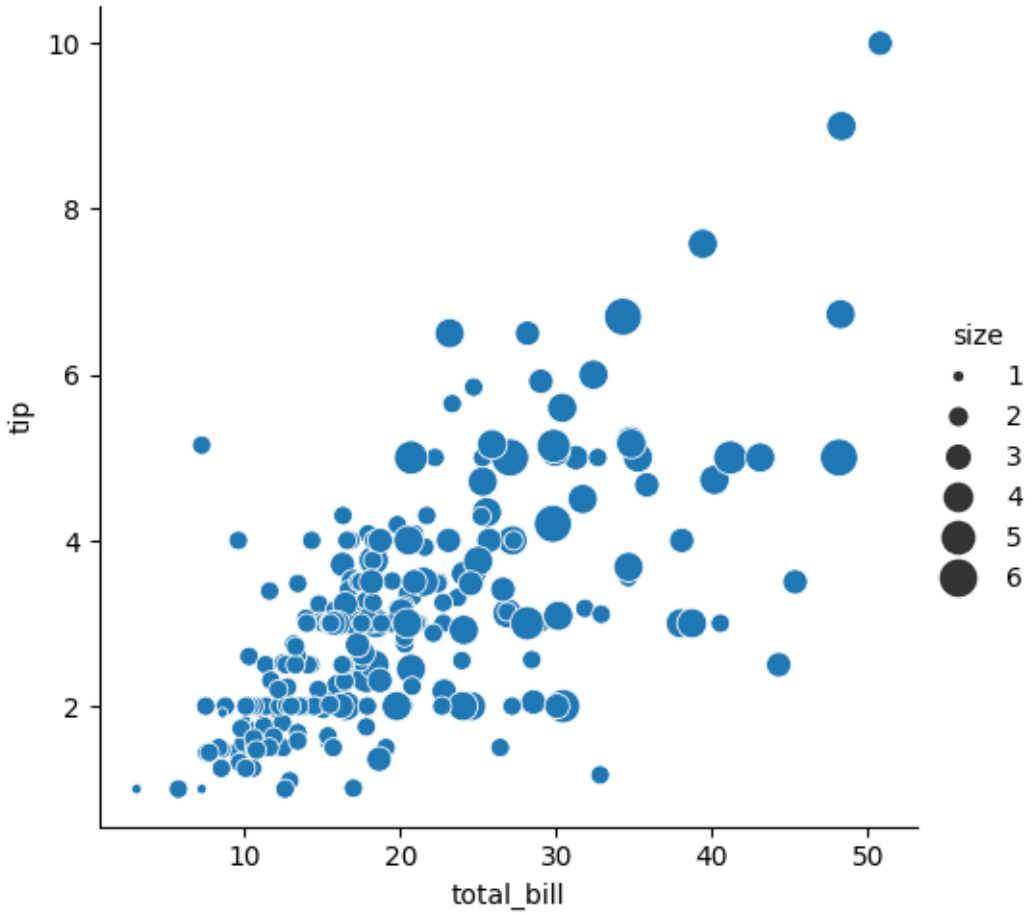
[72]:
```
sns.relplot(data=ds,x='total_bill',y='tip',hue='size')
```

[72]: <seaborn.axisgrid.FacetGrid at 0x193ff7a7c20>



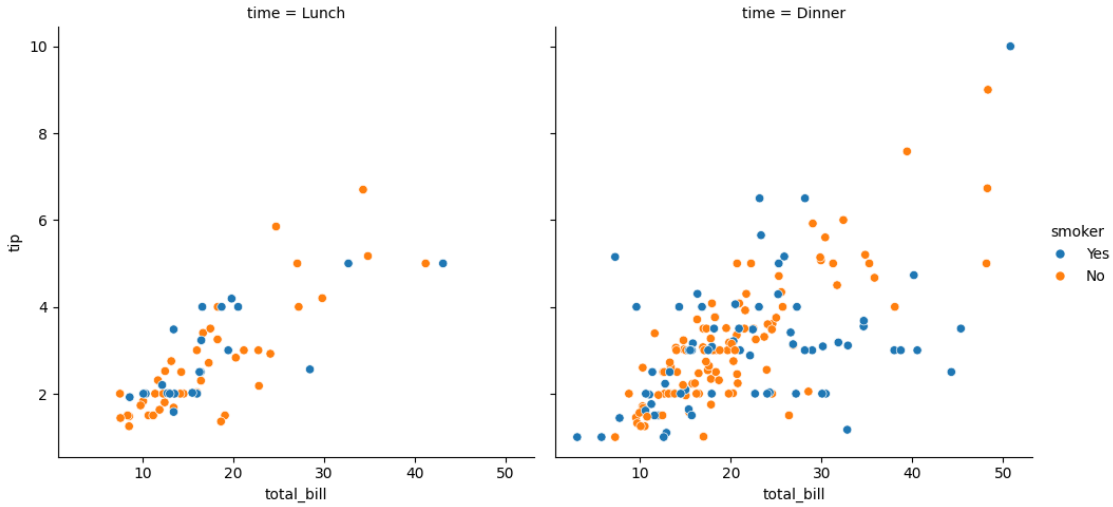
[73]:
```
#Using sizes
sns.relplot(data=ds,x='total_bill',y='tip',size='size',sizes=(15,200))
```

[73]: <seaborn.axisgrid.FacetGrid at 0x193ff77b770>



[74]:
```
#Making subplots
sns.relplot(data=ds,x='total_bill',y='tip',hue='smoker',col='time')
```

[74]: <seaborn.axisgrid.FacetGrid at 0x193ffd22e70>



Performing all the above repeated codes for another dataset in seaborn

[75]:
```
#Loading fmri dataset from seaborn
fmri=sns.load_dataset("fmri")
```

[76]:
```
#Prints first 5 rows
fmri.head()
```

[76]:

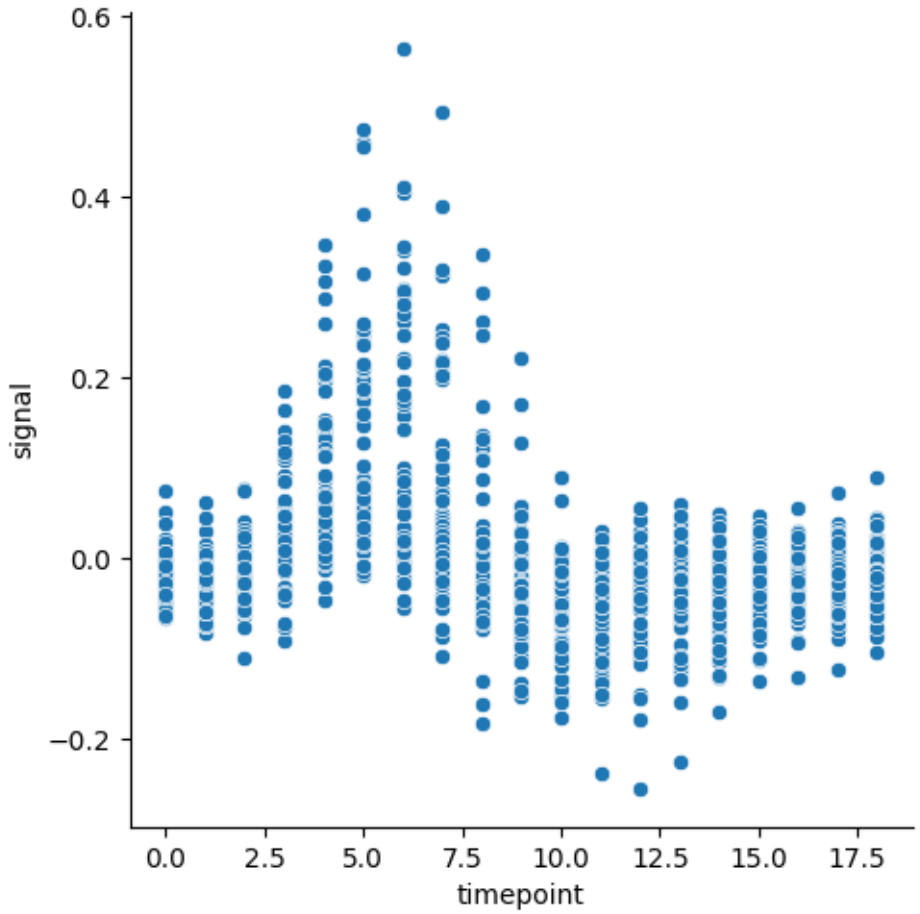
	subject	timepoint	event	region	signal
0	s13	18	stim	parietal	-0.017552
1	s5	14	stim	parietal	-0.080883
2	s12	18	stim	parietal	-0.081033
3	s11	18	stim	parietal	-0.046134
4	s10	18	stim	parietal	-0.037970

[77]:
```
#Gives a tuple that describes no.of rows and no.of columns
fmri.shape
```

[77]:
```
(1064, 5)
```

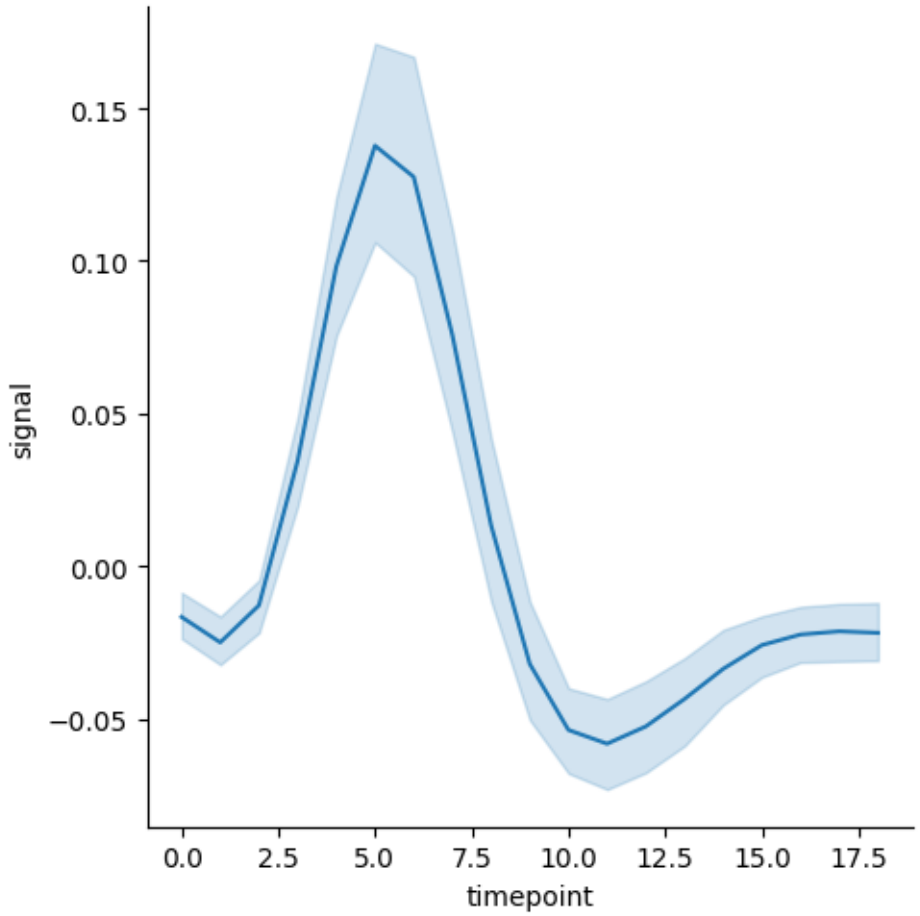
[78]:
```
sns.relplot(data=fmri,x='timepoint',y='signal')
```

[78]:
```
<seaborn.axisgrid.FacetGrid at 0x193ffe5afc0>
```



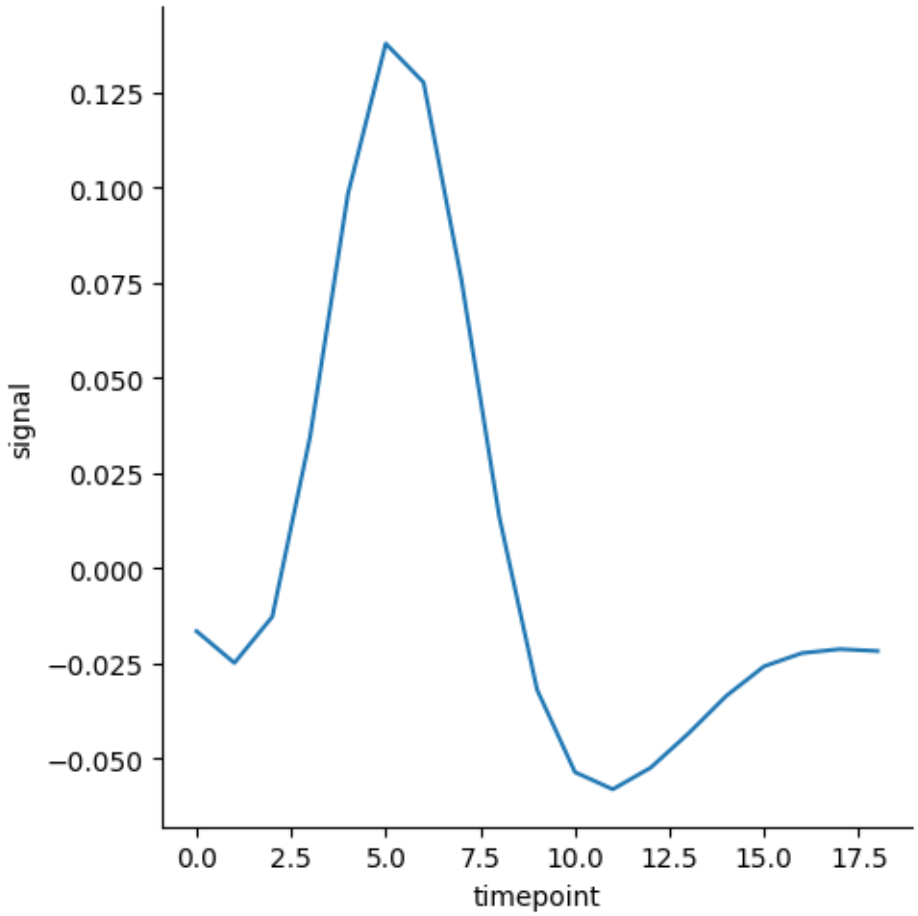
[79]:
```
# Plots a line graph showing how the signal changes over timepoint using the␣
↪fmri dataset
sns.relplot(data=fmri,x='timepoint',y='signal',kind='line')
```

[79]:
```
<seaborn.axisgrid.FacetGrid at 0x1938d132e70>
```



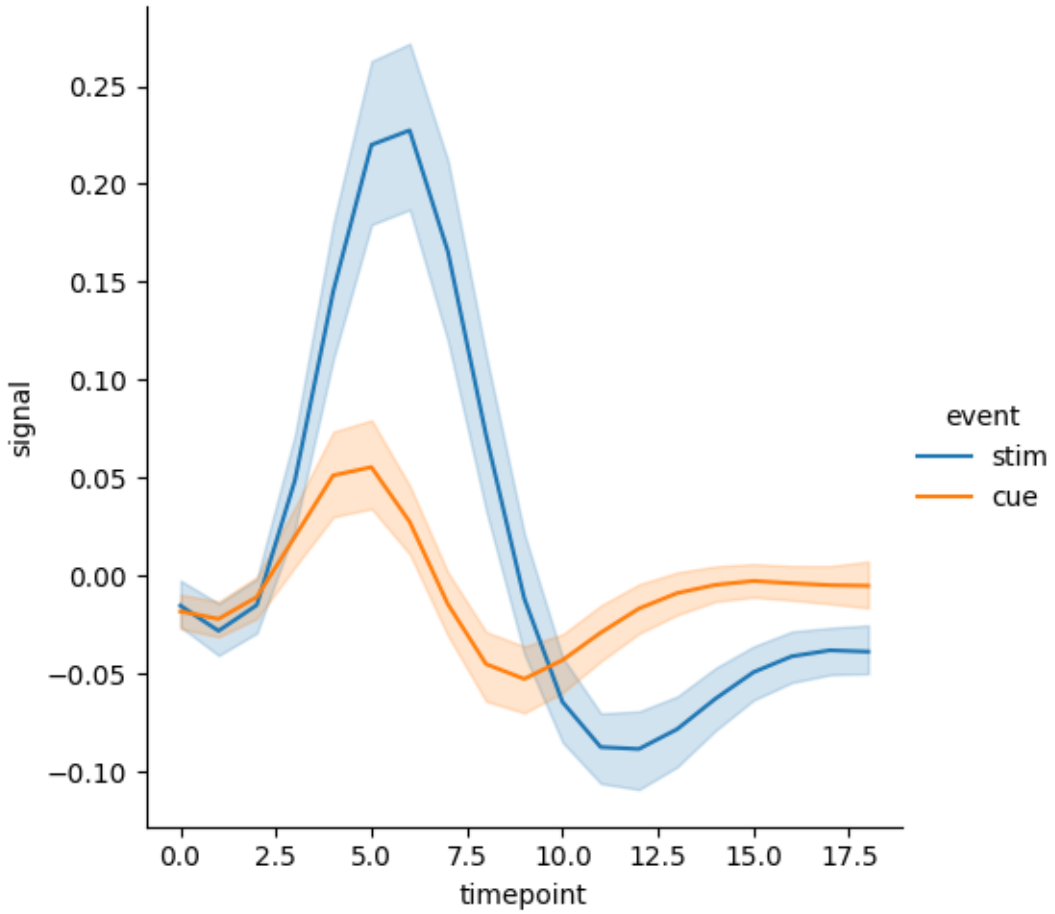
[80]:
```
# Plots a line graph of signal vs timepoint without displaying error bars
sns.relplot(data=fmri,x='timepoint',y='signal',kind='line',errorbar=None)
```

[80]: <seaborn.axisgrid.FacetGrid at 0x193ffd62e70>



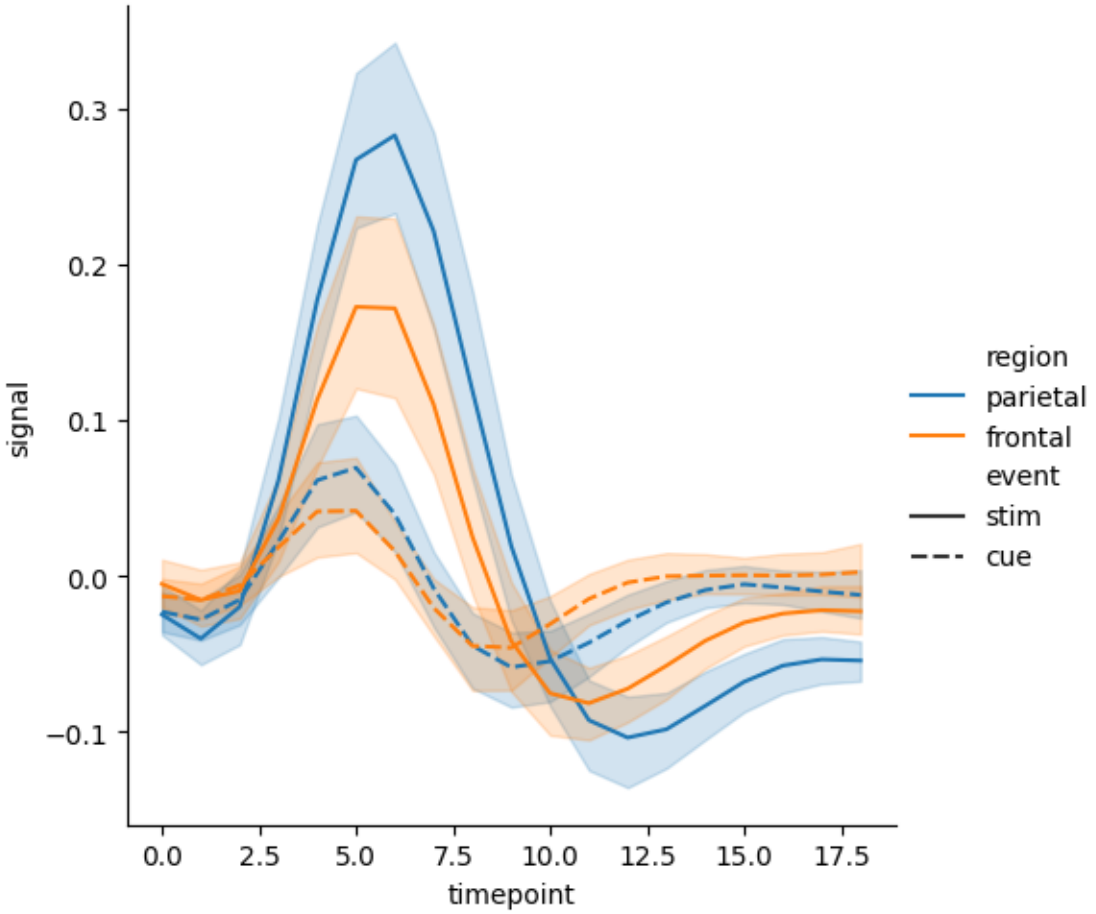
[81]:
```
#Adding a hue semantic with two level splits
#They plot into two lines and error bands
sns.relplot(data=fmri,kind='line',x='timepoint',y='signal',hue='event')
```

[81]: <seaborn.axisgrid.FacetGrid at 0x1938d699340>



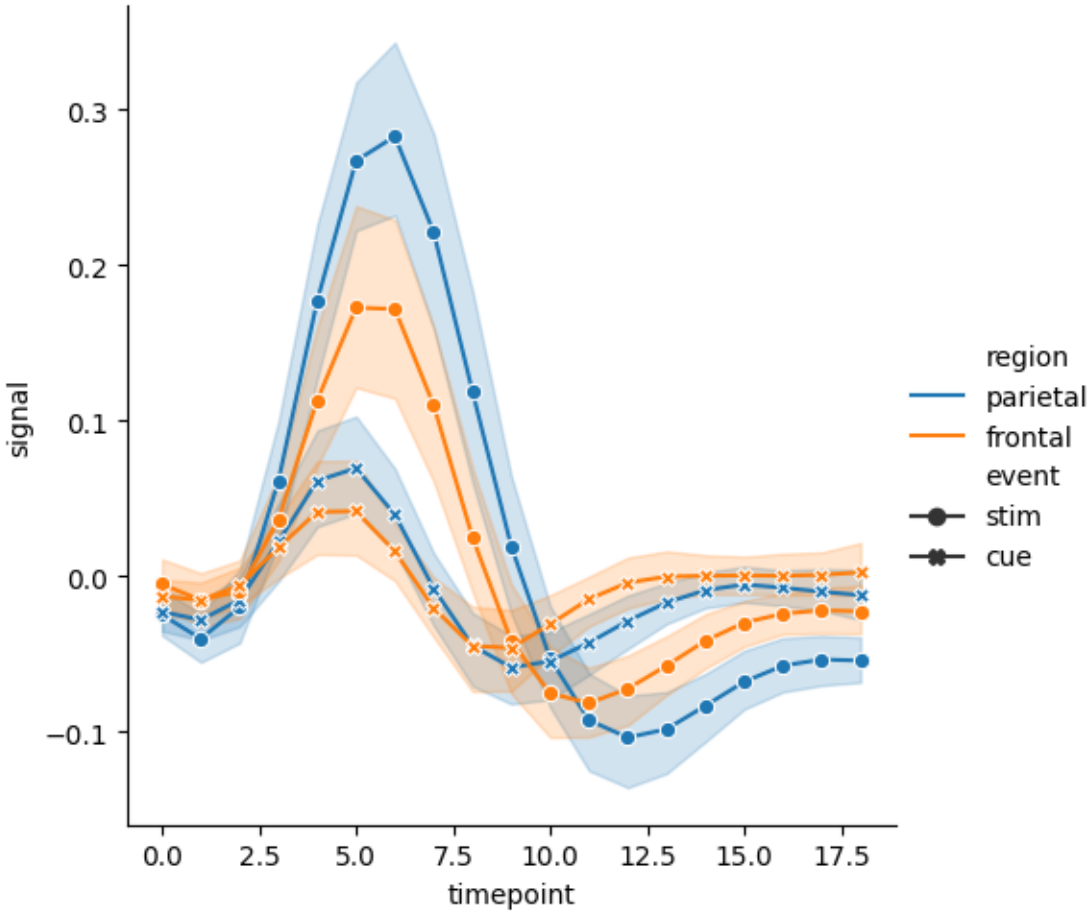
[82]:
```
#Plots a line graph of signal vs timepoint, differentiating lines by color and style
sns.relplot(data=fmri,kind='line',x='timepoint',y='signal',hue='region',style='event')
```

[82]: <seaborn.axisgrid.FacetGrid at 0x1938d148c20>



[83]:
```
#We can identify subsets by the markers used at each observation
sns.
 ↪relplot(data=fmri,kind='line',x='timepoint',y='signal',hue='region',style='event',dashes=Fa
```

[83]: <seaborn.axisgrid.FacetGrid at 0x1938d827bc0>



Categorical Plot

[84]:
```
#Categorical scatterplots
#The default representation of the data in catplot() uses a scatterplot
#There are actually two different categorical scatter plots in seaborn
import seaborn as sns
```

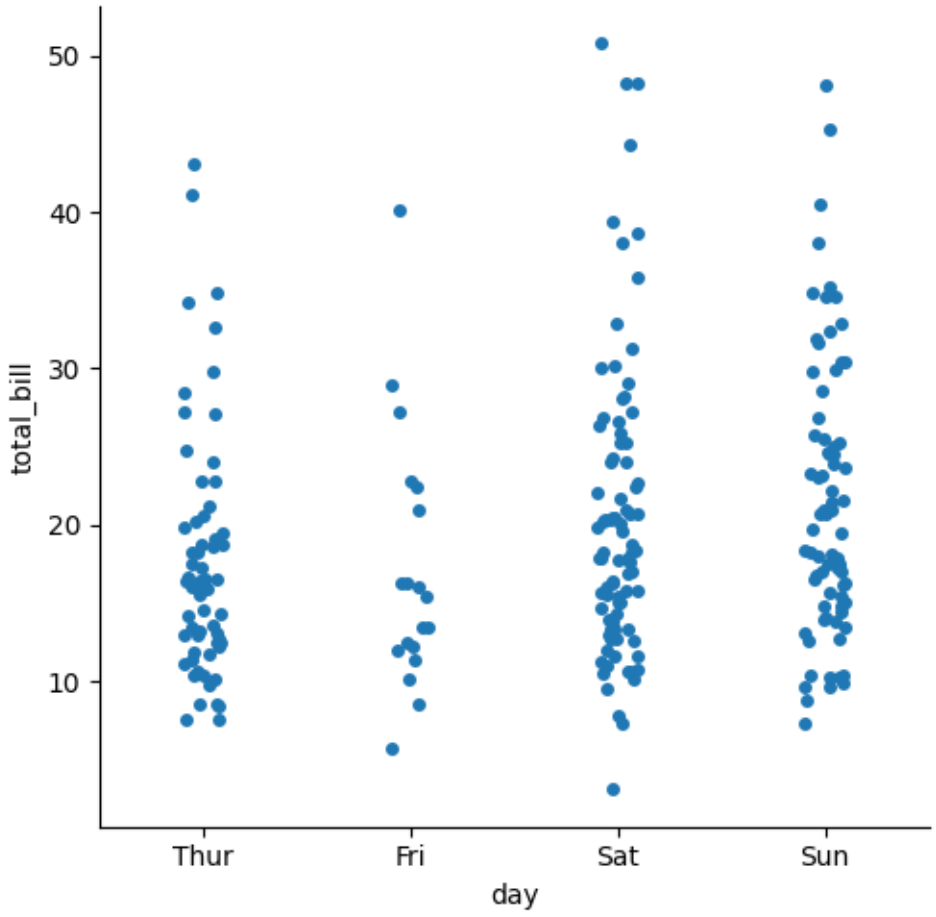
[85]:
```
#Loading tips dataset from seaborn and printing the first 5 rows
tips=sns.load_dataset('tips')
tips.head()
```

[85]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

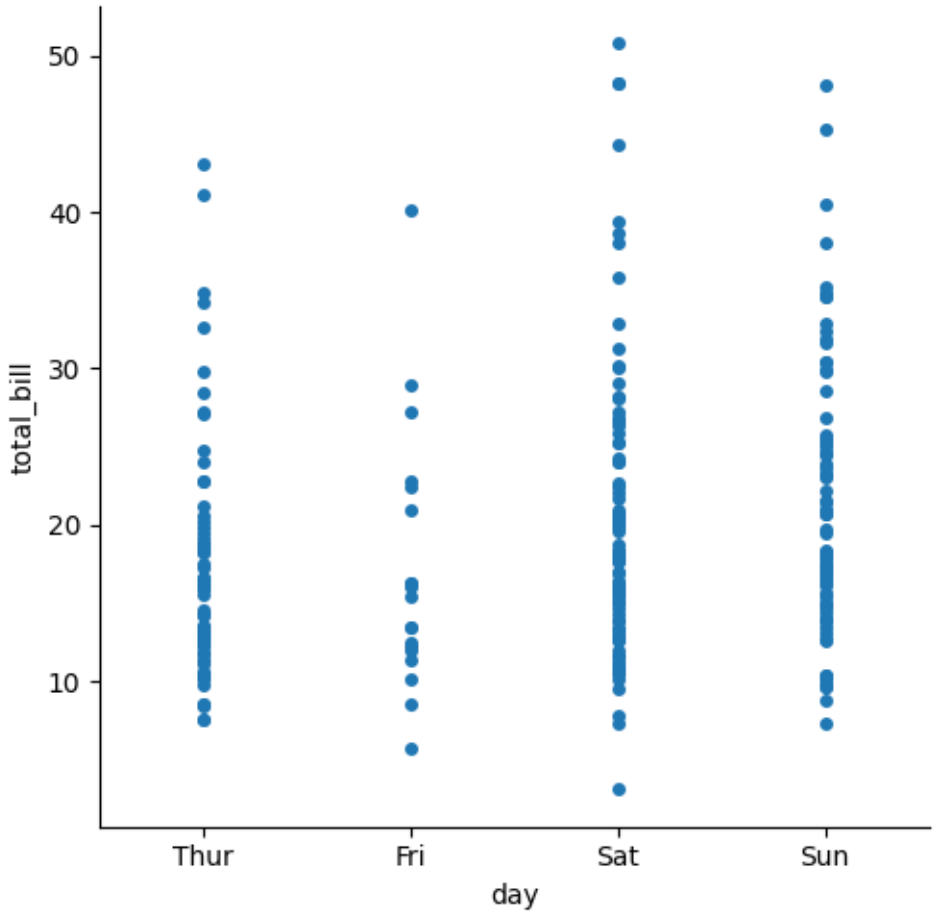
[86]:
```
#Plotting a categorical plot
sns.catplot(data=tips,x='day',y='total_bill')
```

[86]: <seaborn.axisgrid.FacetGrid at 0x1938d698fb0>



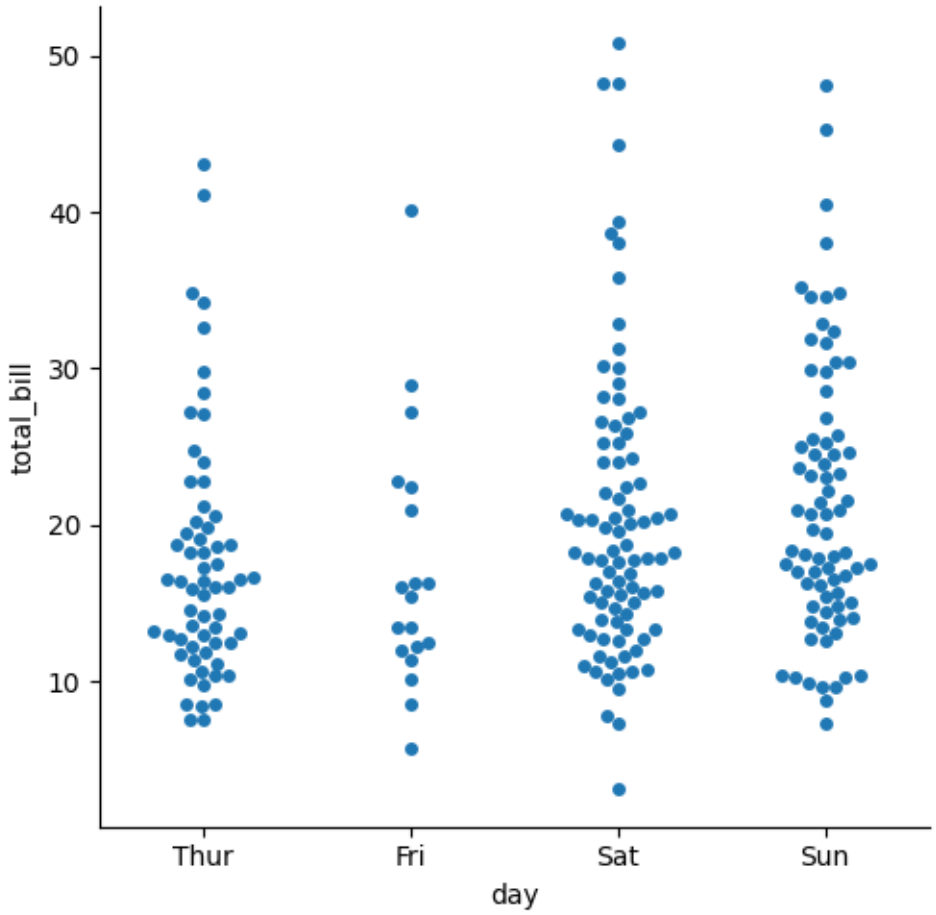
[87]:
```
#The jitter parameter controls the magnitude of jitter or disables it altogether
sns.catplot(data=tips,x='day',y='total_bill',jitter=False)
```

[87]: <seaborn.axisgrid.FacetGrid at 0x1938d743530>



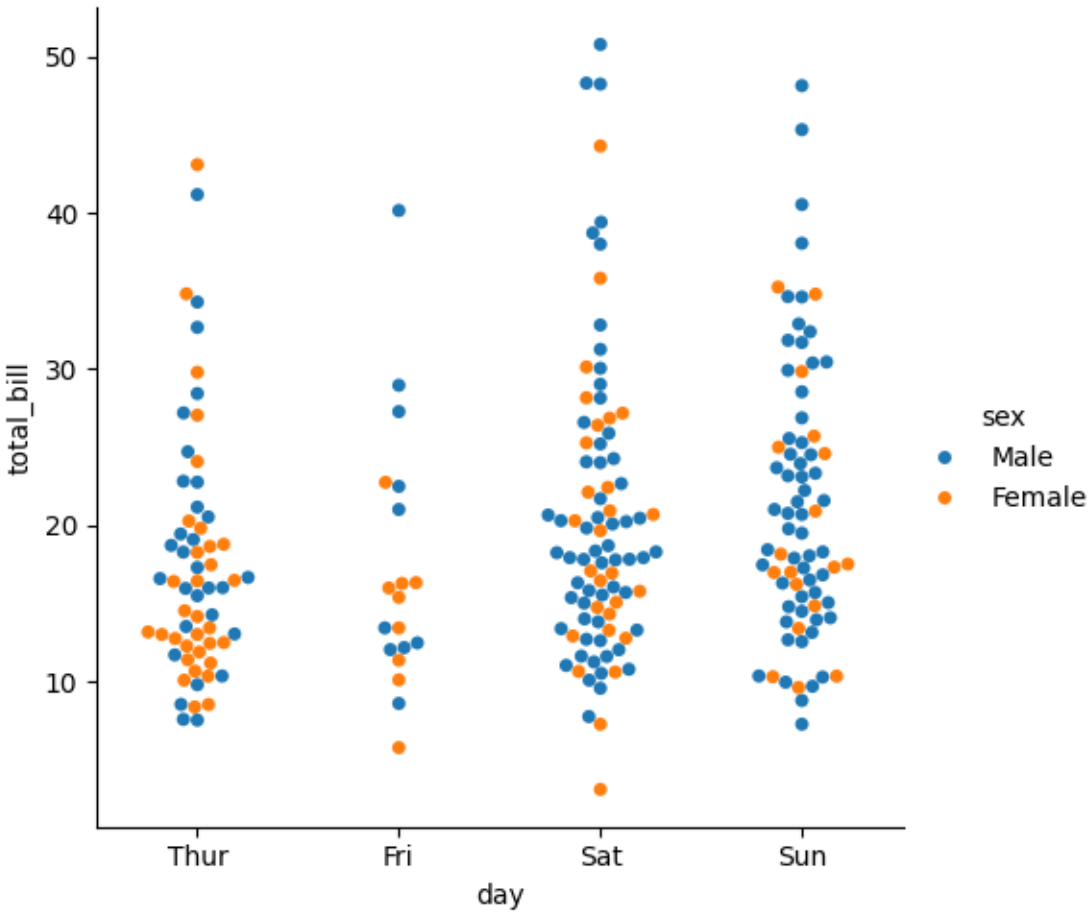
[88]:
```
#Prevent from overlapping(Swarm plot)
sns.catplot(data=tips,x='day',y='total_bill',kind='swarm')
```

[88]: <seaborn.axisgrid.FacetGrid at 0x193ff8a5400>



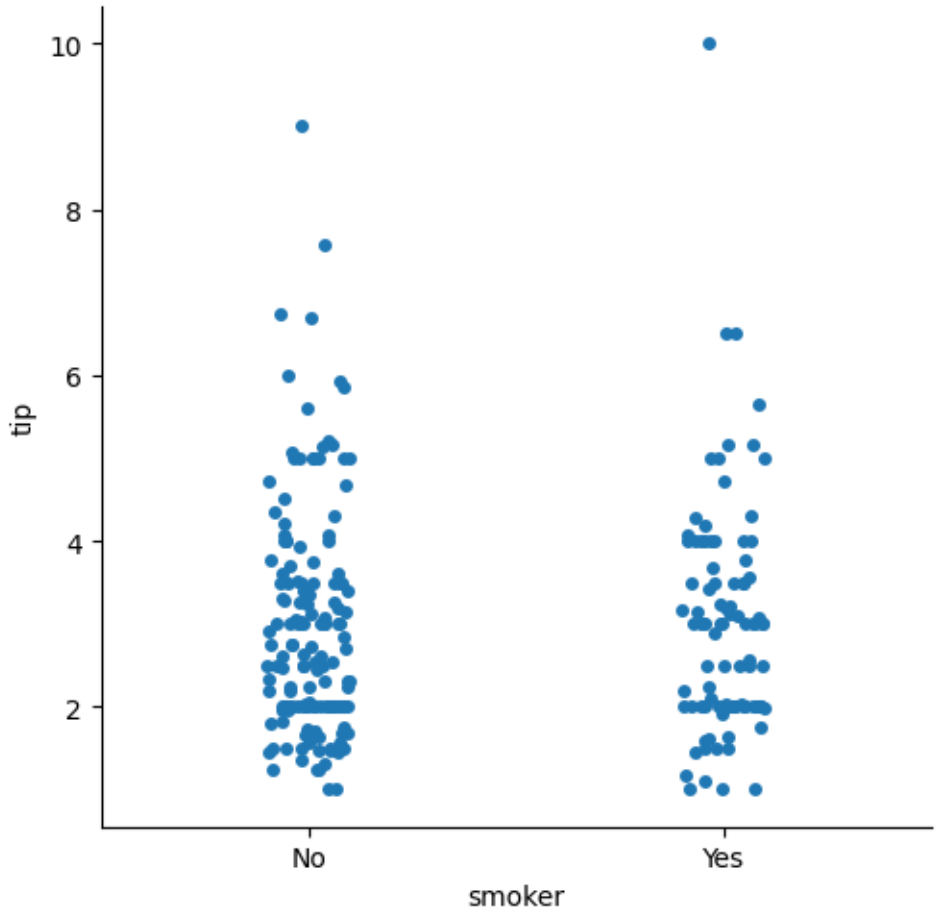
[89]:
```
#Add the hue semantic
sns.catplot(data=tips,x='day',y='total_bill',hue='sex',kind='swarm')
```

[89]: <seaborn.axisgrid.FacetGrid at 0x1938eaf7680>



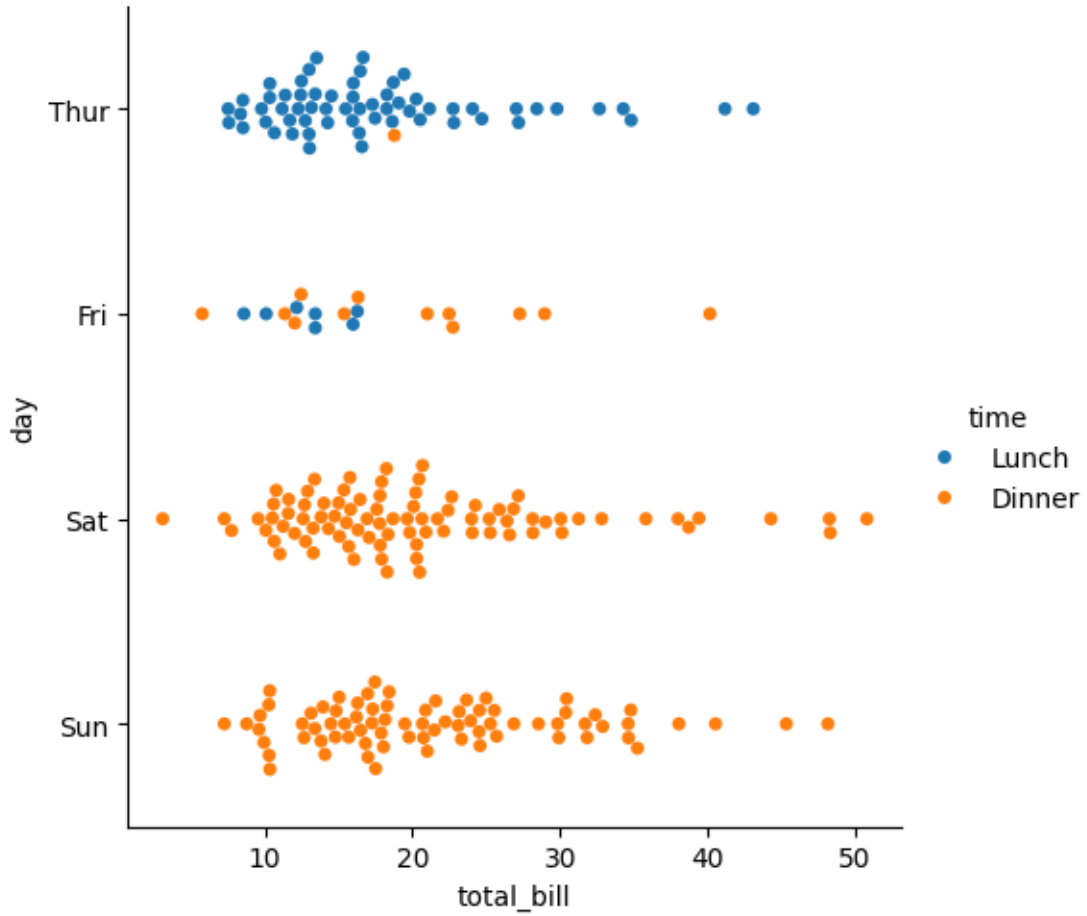
[90]:
```
#Order parameter: to display multiple categorical plot in the single figure
sns.catplot(data=tips,x='smoker',y='tip',order=['No','Yes'])
```

[90]: <seaborn.axisgrid.FacetGrid at 0x1938eef5e80>



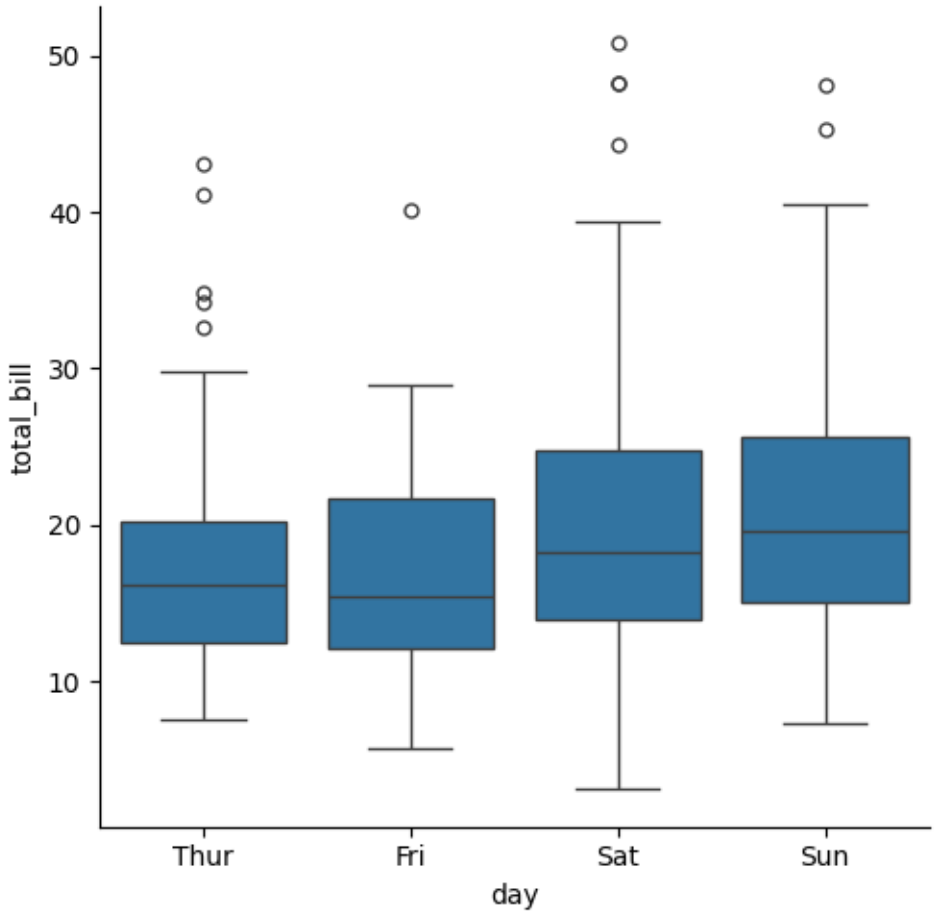
[91]:
```
#Categorical plot on vertical axis
sns.catplot(data=tips,x='total_bill',y='day',hue='time',kind='swarm')
```

[91]: <seaborn.axisgrid.FacetGrid at 0x1938d1dedb0>



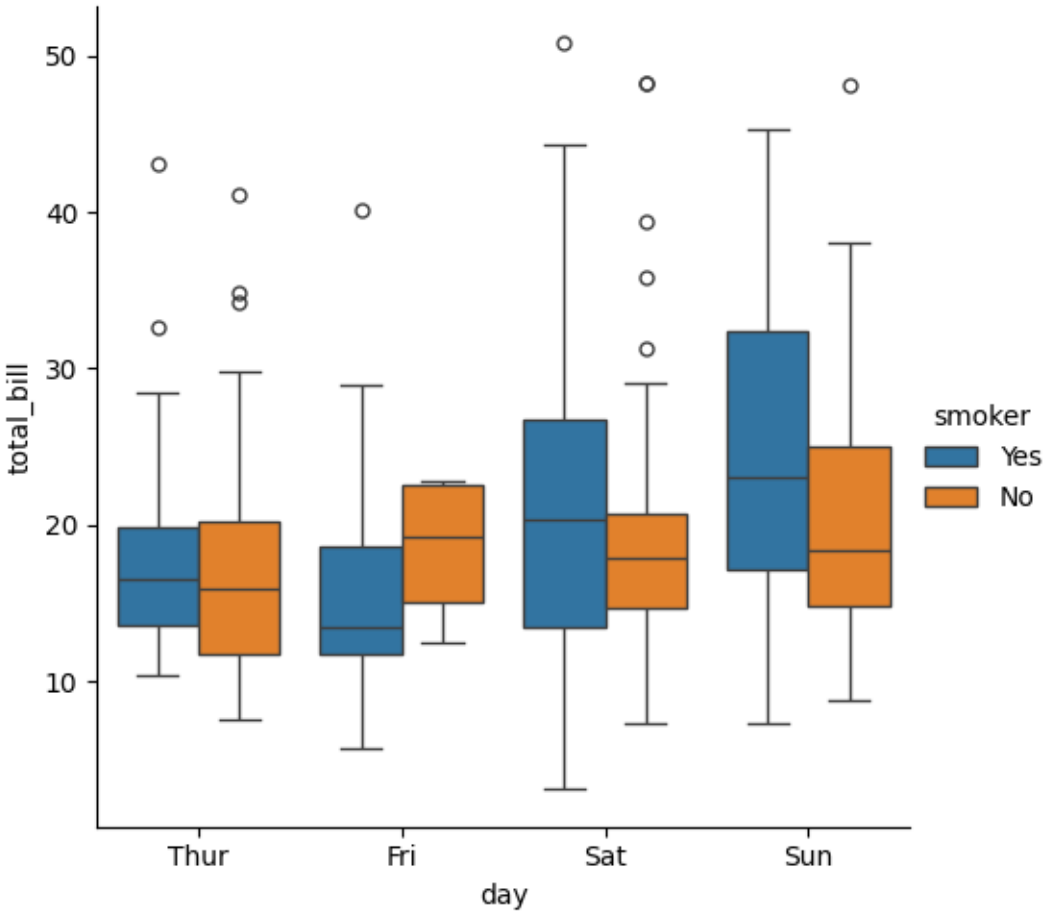
[92]:
```
#Comparing Distributions
#Boxplots
sns.catplot(data=tips,x='day',y='total_bill',kind='box')
```

[92]: <seaborn.axisgrid.FacetGrid at 0x1938ef7d190>



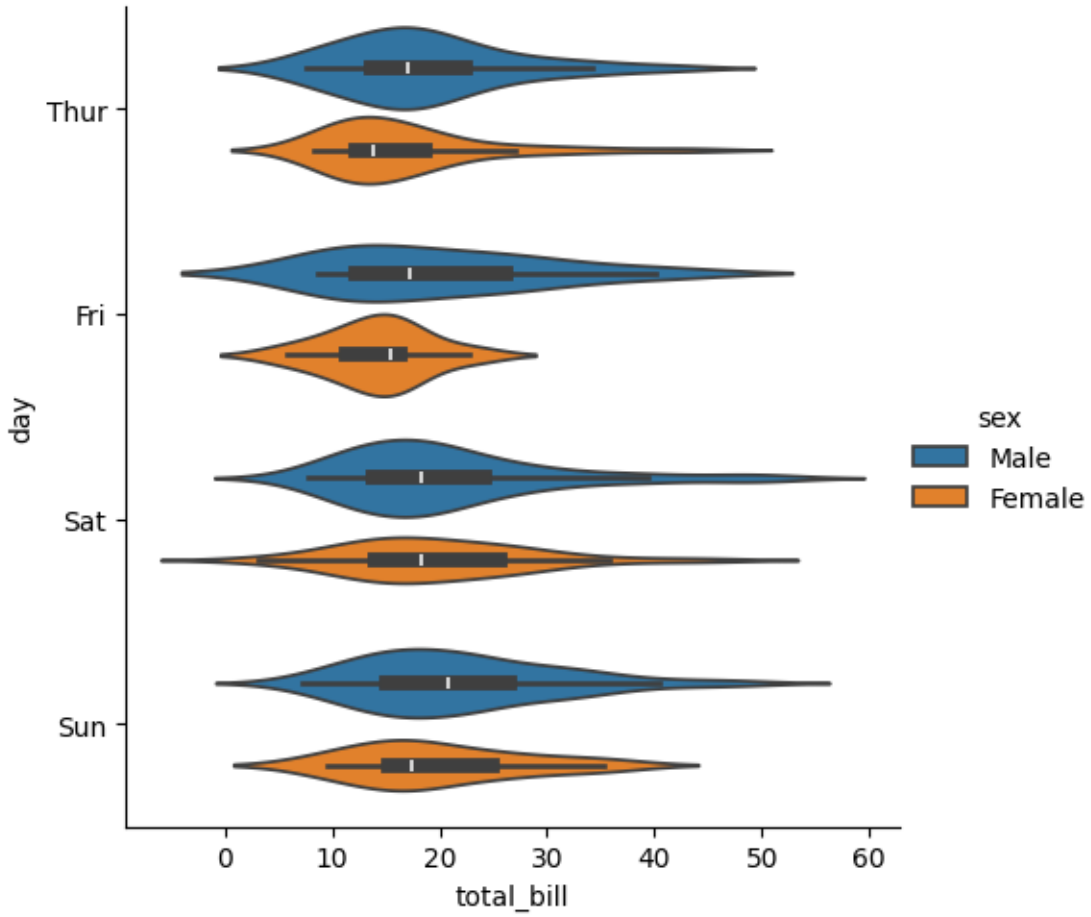
[93]:
```
#Adding hue semantic
sns.catplot(data=tips,x='day',y='total_bill',hue='smoker',kind='box')
```

[93]: <seaborn.axisgrid.FacetGrid at 0x1938efdedb0>



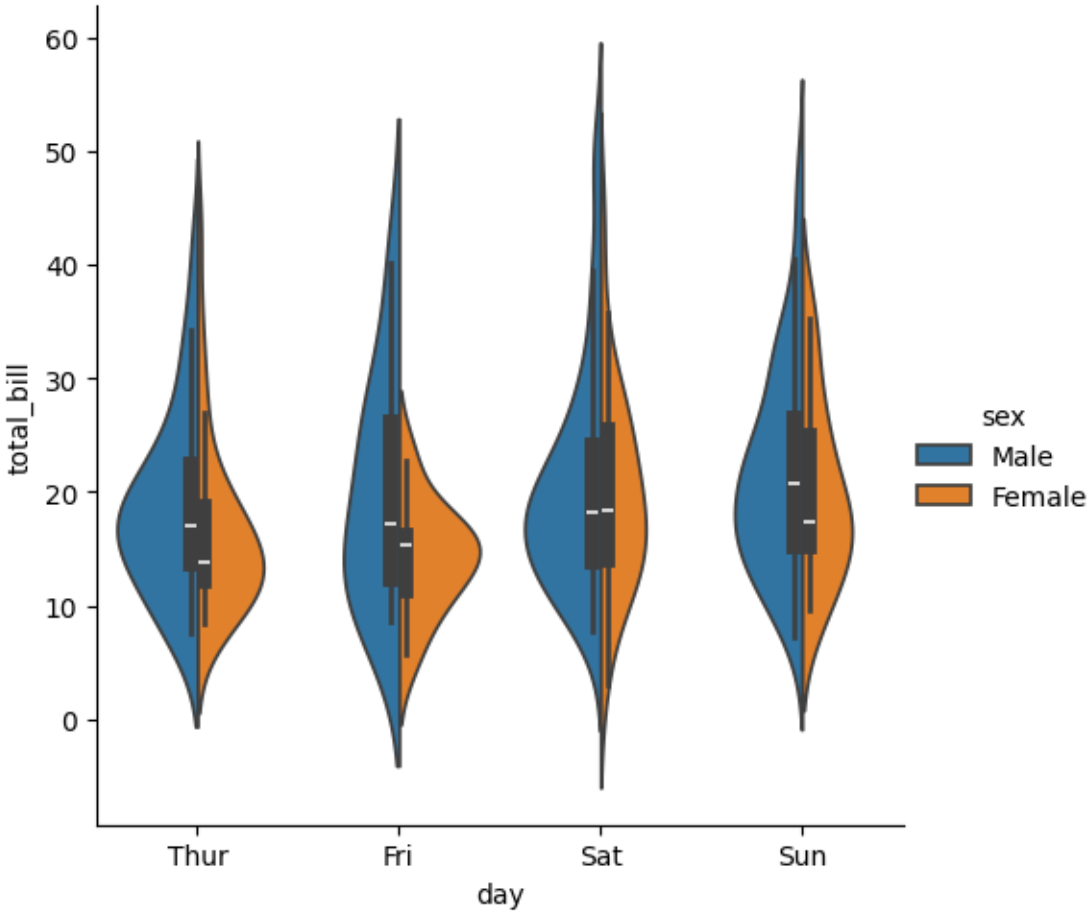
[94]:
```
#VIOLIN plor
sns.catplot(data=tips,x='total_bill',y='day',hue='sex',kind='violin')
```

[94]: <seaborn.axisgrid.FacetGrid at 0x1938efc5ee0>



[95]:
```
#Split in the violin plot
sns.catplot(data=tips,x='day',y='total_bill',hue='sex',kind='violin',split=True)
```

[95]: <seaborn.axisgrid.FacetGrid at 0x1938f053b60>


60
50
40
30
20
10
0
total_bill
Thur
Fri
Sat
Sun
day
sex
Male
Female