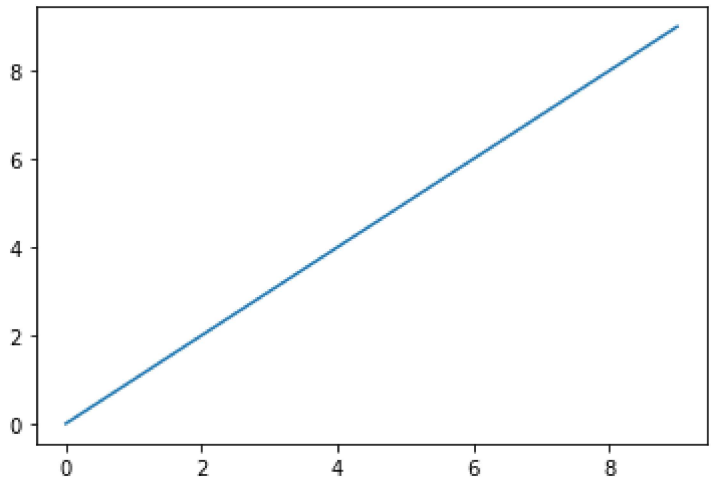
In [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
```

In [5]:
```
import matplotlib.pyplot as plt
import numpy as np
```

In [6]:
```
 data = np.arange(10)
```

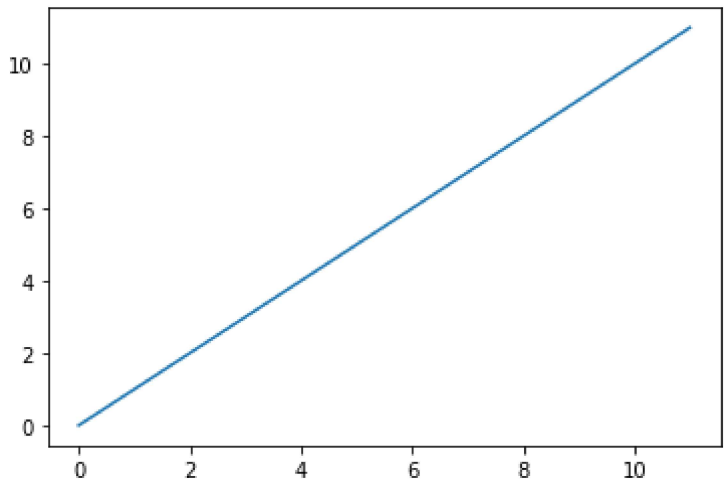
In [7]:
```
plt.plot(data)
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0x1c9ff9a41c0>]
```



In [11]:
```
d=range(12)
plt.plot(d)
```

Out[11]:
```
[<matplotlib.lines.Line2D at 0x1c9ffc3baf0>]
```



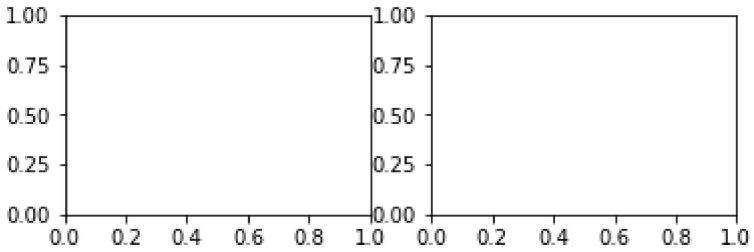
In [12]:
```
d
```

Out[12]:
```
range(0, 12)
```

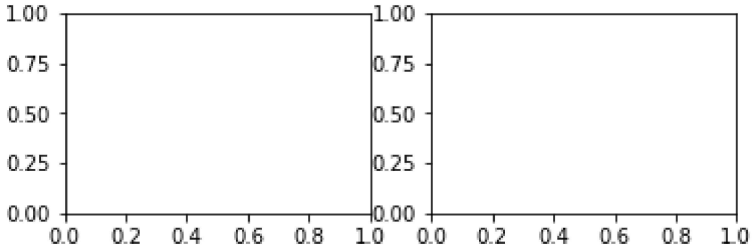
In [4]:
```
fig = plt.figure()
```

```
<Figure size 432x288 with 0 Axes>
```

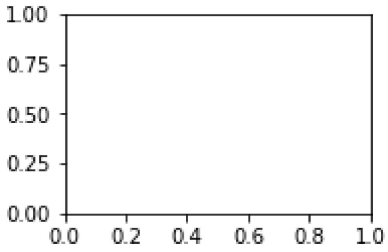
In [19]:
```
fig=plt.figure()
ax1=fig.add_subplot(2,2,1)
ax2=fig.add_subplot(2,2,2)
```



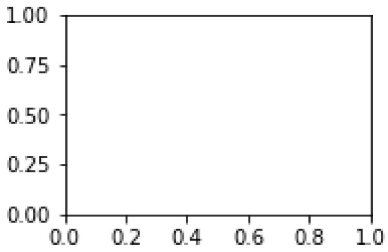
In [20]:
```
fig
```

Out[20]:


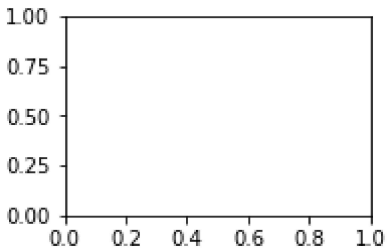
In [21]:
```
ax1 = plt.subplot(2, 2, 1)
```



In [22]:
```
ax2 = plt.subplot(2, 2, 2)
```

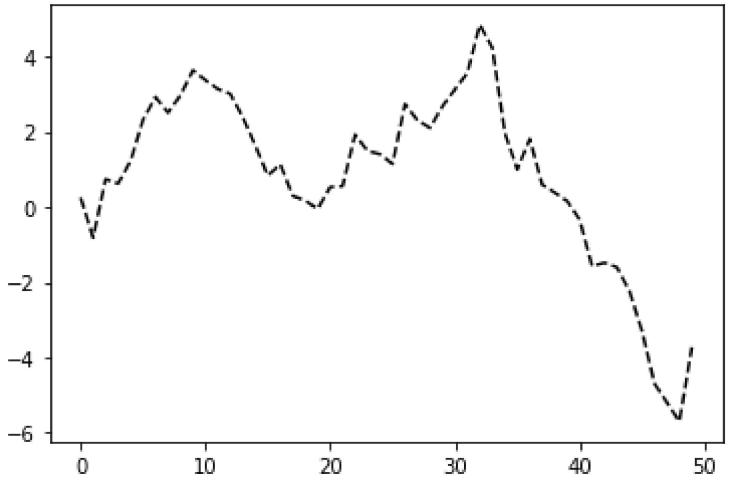


In [23]:
```
ax3 = plt.subplot(2, 2, 3)
```



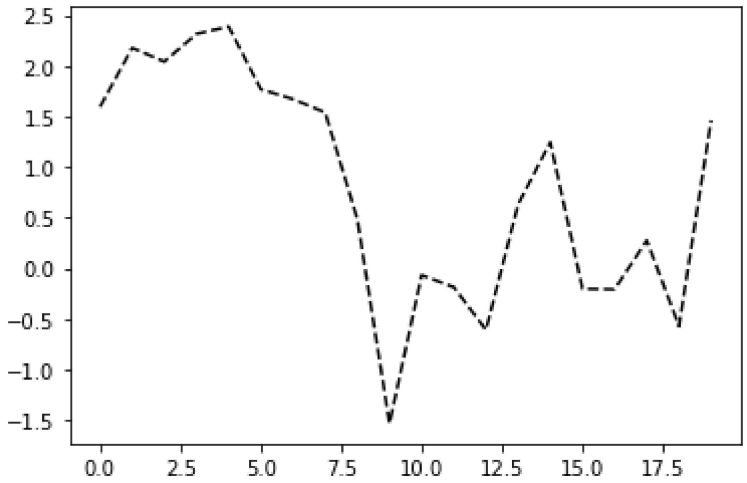
In [9]:
```
plt.plot(np.random.randn(50).cumsum(), 'k--')
```

Out[9]:
```
[<matplotlib.lines.Line2D at 0x2147eb37fa0>]
```



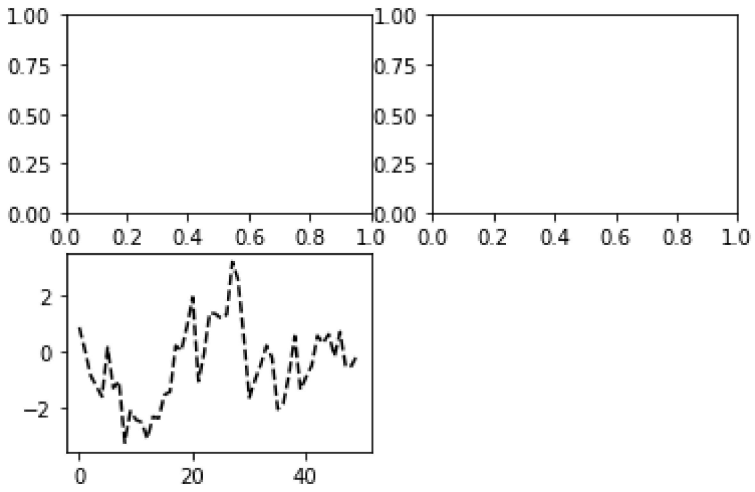
In [25]:
```
plt.plot(np.random.randn(20).cumsum(),'k--')
```

Out[25]:
```
[<matplotlib.lines.Line2D at 0x1c98b036910>]
```



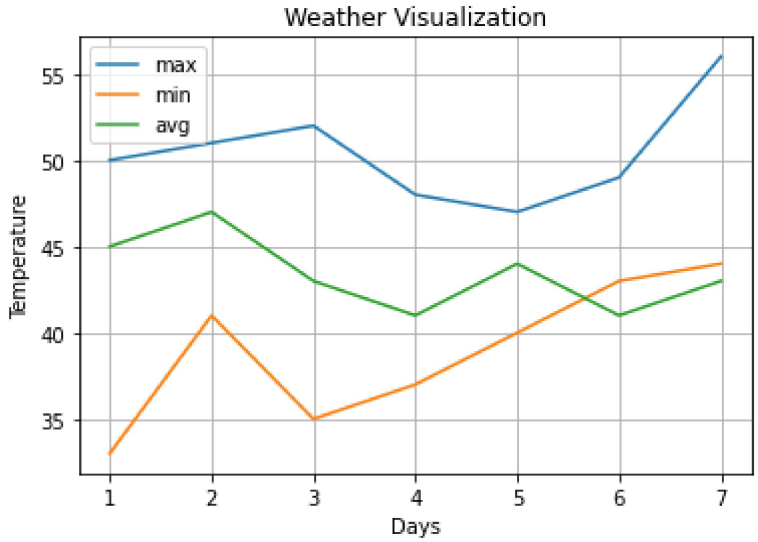
In [26]:
```
ax1 = plt.subplot(2, 2, 1)
ax2 = plt.subplot(2, 2, 2)
ax3 = plt.subplot(2, 2, 3)
#plt.show()
plt.plot(np.random.randn(50).cumsum(), 'k--')
```

Out[26]:
```
[<matplotlib.lines.Line2D at 0x1c98b0fe670>]
```



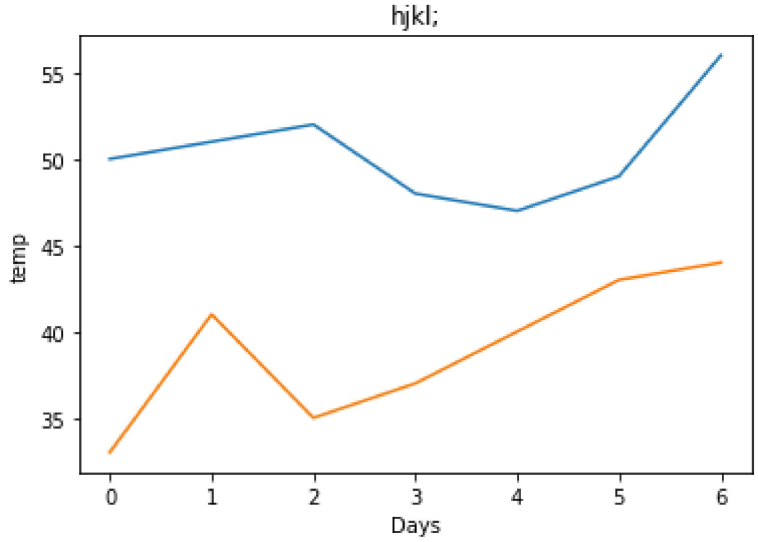
In [27]:
```
import matplotlib.pyplot as plt
import numpy as np
```

In [31]:
```
data=np.arange(1,8)
max_t=[50,51,52,48,47,49,56]
min_t=[33,41,35,37,40,43,44]
avg_t=[45,47,43,41,44,41,43]
plt.plot(data,max_t,label="max")
plt.plot(data,min_t,label="min")
plt.plot(data,avg_t,label="avg")
plt.xlabel("Days")
plt.ylabel("Temperature")
plt.title("Weather Visualization")
plt.legend()
plt.grid()
```



In [34]:
```
d=np.arange(7)
x=[50,51,52,48,47,49,56]
y=[33,41,35,37,40,43,44]
plt.plot(d,x,label='max')
plt.plot(d,y,label='min')
plt.xlabel('Days')
plt.ylabel('temp')
plt.title("hjkl;")
```

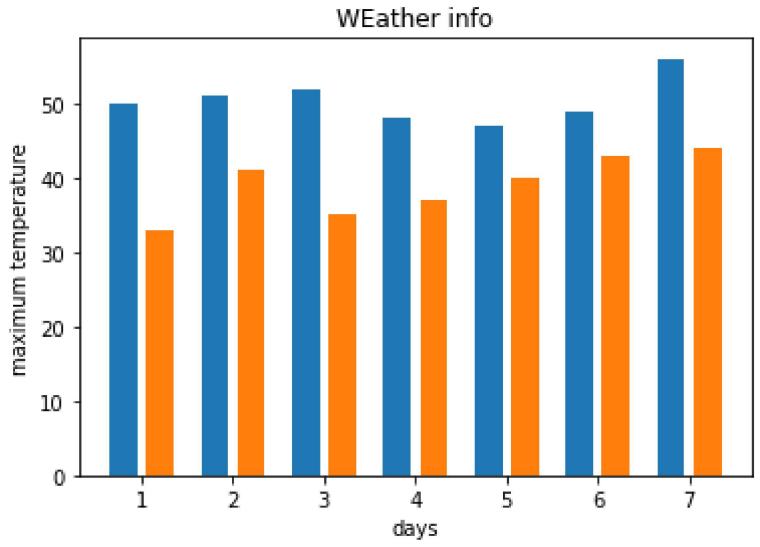
Out[34]:
```
Text(0.5, 1.0, 'hjkl;')
```



In [43]:
```
plt.bar(data-0.2,x,width=0.3,label='max')
plt.bar(data+0.2,y,width=0.3,label='min')
plt.xticks(data,[1,2,3,4,5,6,7])
plt.title("WEather info")
```

```
plt.ylabel("maximum temperature")
plt.xlabel("days")
#plt.legend()
```

Out[43]:
```
Text(0.5, 0, 'days')
```

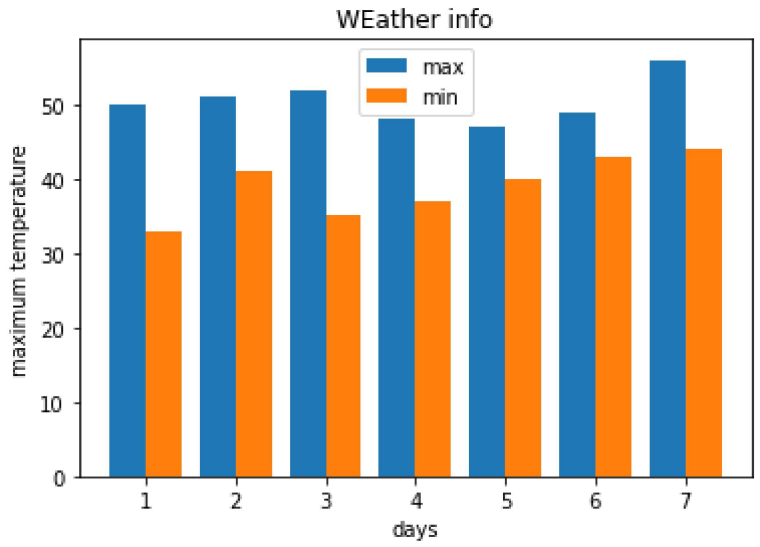


In [35]:
```
data
```

Out[35]:
```
array([1, 2, 3, 4, 5, 6, 7])
```

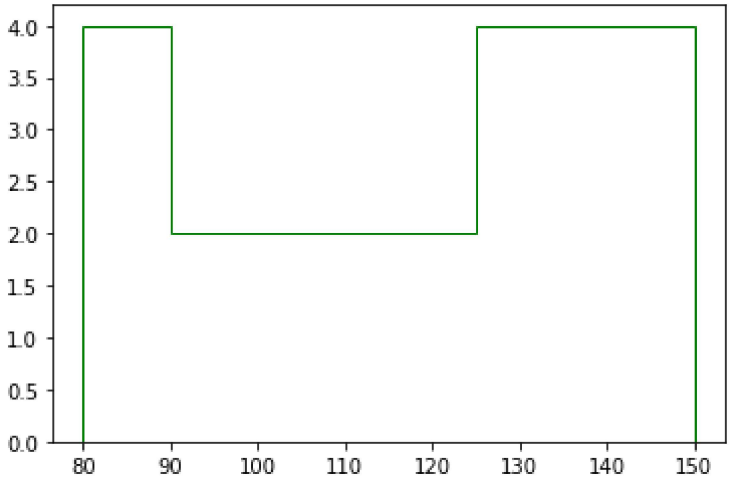
In [13]:
```
plt.ylabel("maximum temperature")
plt.xlabel("days")
plt.xticks(data,[1,2,3,4,5,6,7])
plt.title("WEather info")
plt.bar(data-0.2,max_t,width=0.4,label="max")
plt.bar(data+0.2,min_t,width=0.4,label="min")
plt.legend()
```

Out[13]:
```
<matplotlib.legend.Legend at 0x2147ed05be0>
```



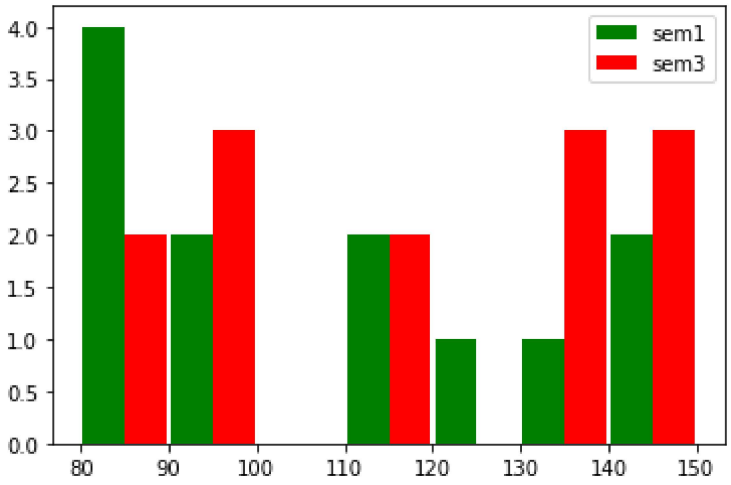
In [45]:
```
marks=[113,85,90,150,149,89,93,115,135,80,77,82,129]
plt.hist(marks,bins=[80,90,110,125,150],rwidth=0.95,color='green',histtype='step')
#plt.hist(marks,bins=[80,90,110,125],rwidth=0.95,color='red',alpha=0.5)
```

Out[45]:
```
(array([4., 2., 2., 4.]),
 array([ 80,  90, 110, 125, 150]),
 [<matplotlib.patches.Polygon at 0x1c98df85e20>])
```

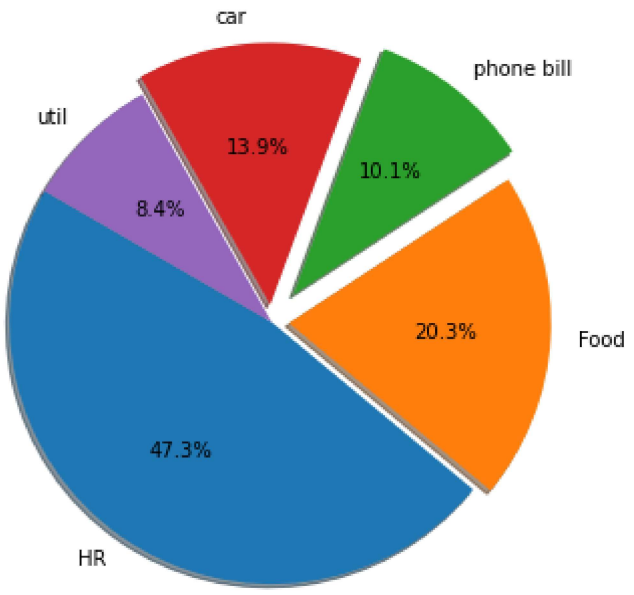


In [46]:
```
mark1=[113,85,90,150,149,89,93,115,135,80,77,82,129]
mark3=[134,145,87,93,91,85,111,117,130,145,144,130,99]
plt.hist([mark1,mark3],bins=[80,90,100,110,120,130,140,150],rwidth=0.95,color=['g'
plt.legend()
```

Out[46]:
```
<matplotlib.legend.Legend at 0x1c9fff68bb0>
```



In [16]:
```
e_value=[1400,600,300,410,250]
e_label=["HR","Food","phone bill","car","util"]
plt.axis("equal")
plt.pie(e_value,labels=e_label,radius=1.5,autopct="%0.1f%%",shadow='true',startang
plt.savefig('piechart.png',bbox_inches="tight",pad_inches=2,transparent=True)
```



In [48]:
```
plt.pie(e_value,labels=e_label,autopct='%0.1f%',shadow=True,explode[0,0.1,0.2,0.1,(
```

```
  Input In [48]
    plt.pie(e_value,labels=e_label,autopct='%0.1f%',shadow=True,explode[0,0.1,0.2,
0.1,0.1])

^
SyntaxError: positional argument follows keyword argument
```

In [17]:
```
data
```

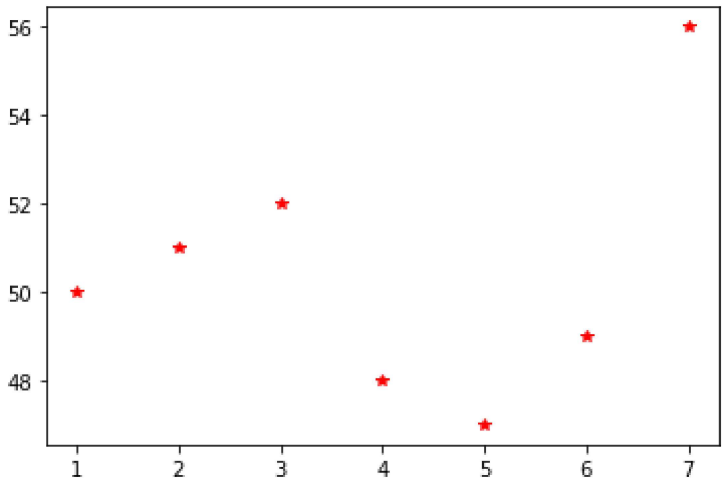
Out[17]:
```
array([1, 2, 3, 4, 5, 6, 7])
```

In [18]:
```
max_t
```

Out[18]:
```
[50, 51, 52, 48, 47, 49, 56]
```

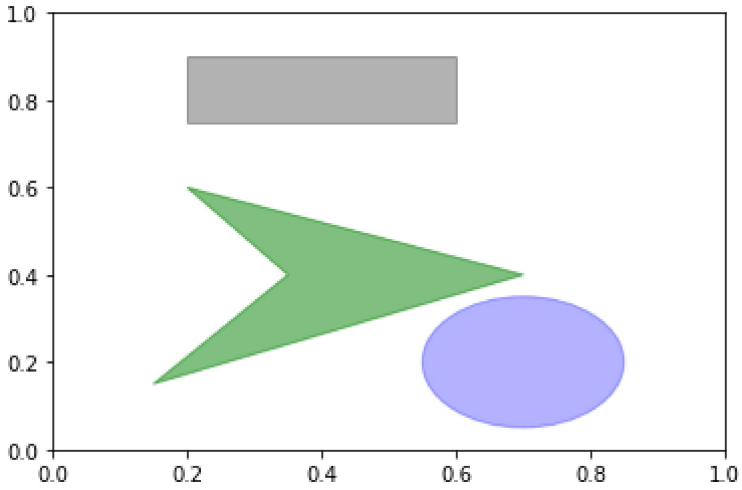
In [52]:
```
plt.plot(data,max_t,'r*',alpha=1)
```

Out[52]:
```
[<matplotlib.lines.Line2D at 0x1c98f188130>]
```



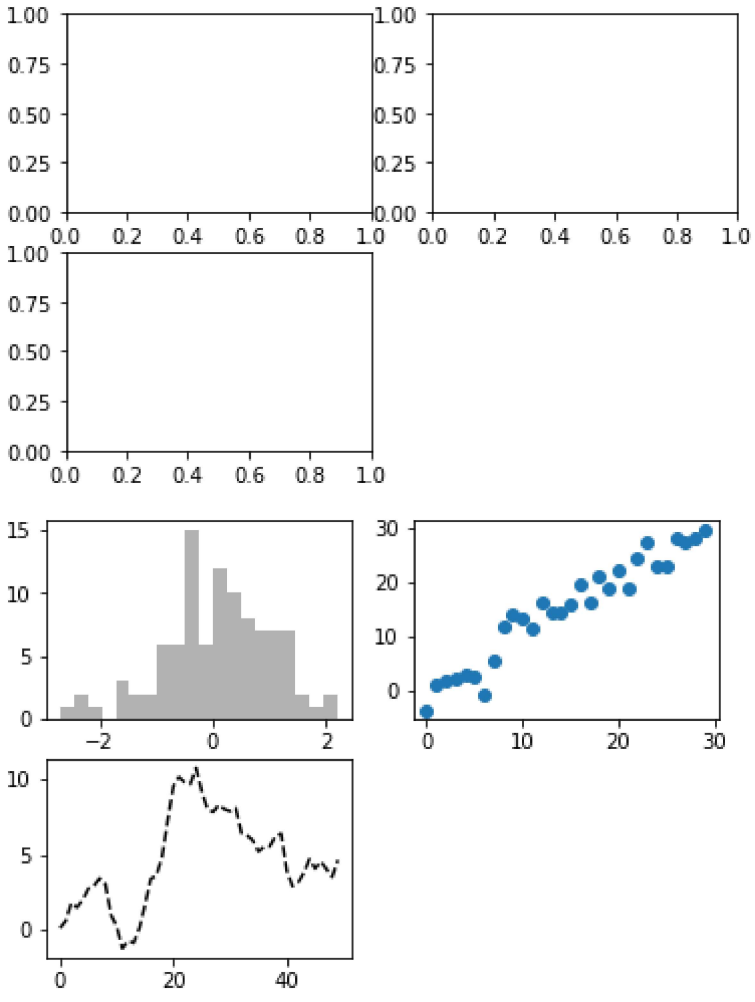
In [20]:
```
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
rect = plt.Rectangle((0.2, 0.75), 0.4, 0.15, color='k', alpha=0.3)
circ = plt.Circle((0.7, 0.2), 0.15, color='b', alpha=0.3)
pgon = plt.Polygon([[0.15, 0.15], [0.35, 0.4], [0.2, 0.6],[0.7,0.4]],
 color='g', alpha=0.5)
ax.add_patch(rect)
ax.add_patch(circ)
ax.add_patch(pgon)
```

Out[20]:
```
<matplotlib.patches.Polygon at 0x2147eae6070>
```



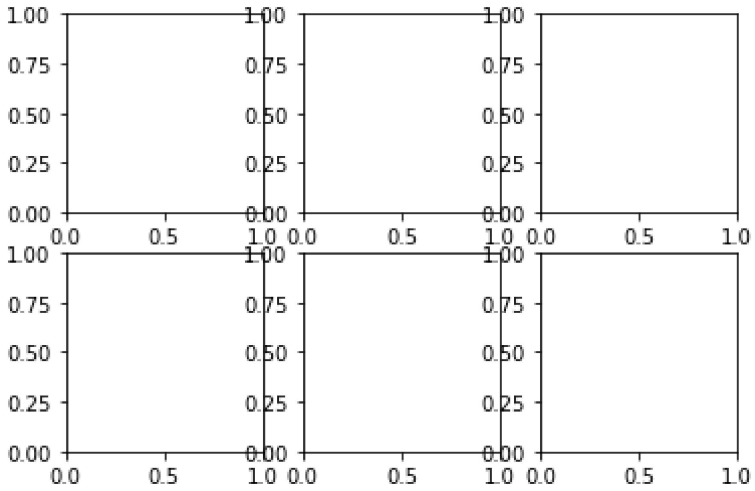
In [21]:
```
#Figures and Subplots
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
plt.plot(np.random.randn(50).cumsum(), 'k--')
_ = ax1.hist(np.random.randn(100), bins=20, color='k', alpha=0.3)
ax2.scatter(np.arange(30), np.arange(30) + 3 * np.random.randn(30))
#plt.close('all')
```

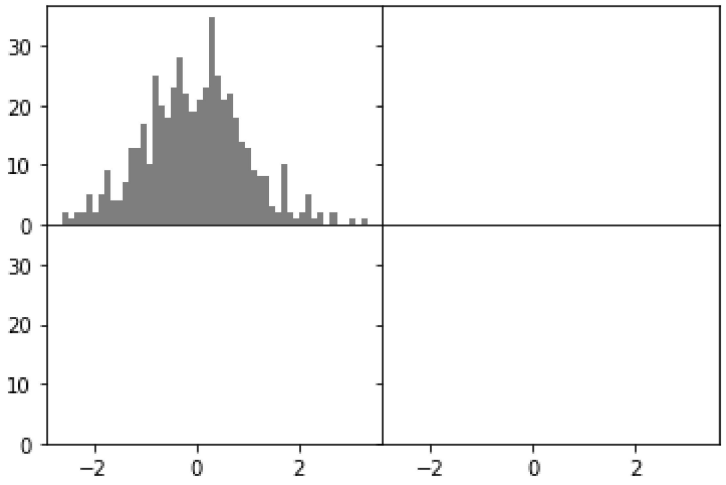
Out[21]:
```
<matplotlib.collections.PathCollection at 0x21402004670>
```

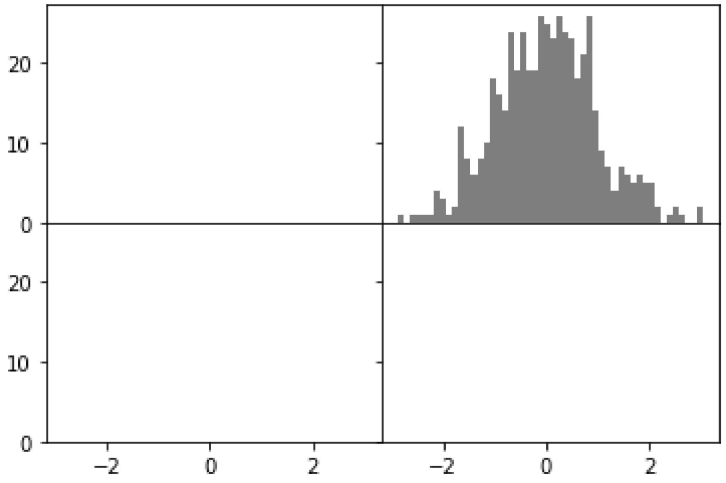


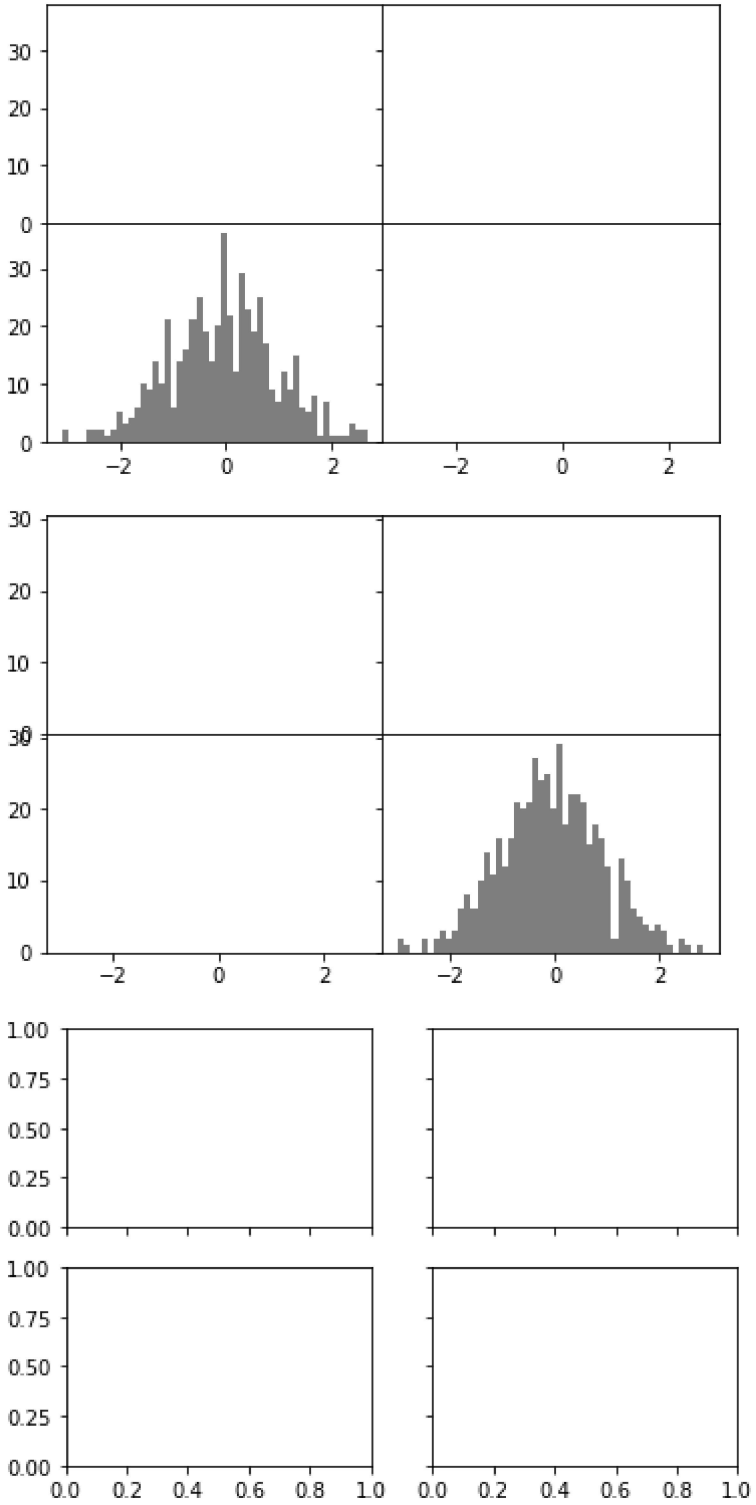
In [22]:
```
fig, axes = plt.subplots(2, 3)
axes
#Adjusting the spacing around subplots
plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspa

fig, axes = plt.subplots(2, 2, sharex=True, sharey=True)
for i in range(2):
    for j in range(2):
        axes[i, j].hist(np.random.randn(500), bins=50, color='k', alpha=0.5)
        plt.subplots_adjust(wspace=0, hspace=0)
        fig, axes = plt.subplots(2, 2, sharex=True, sharey=True)
```

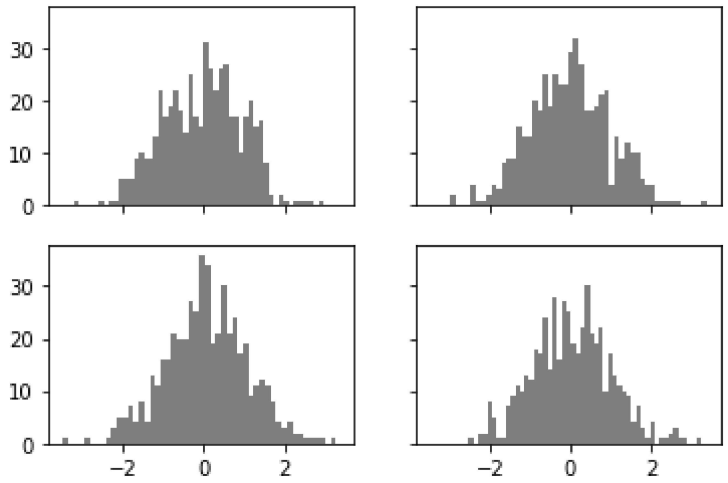






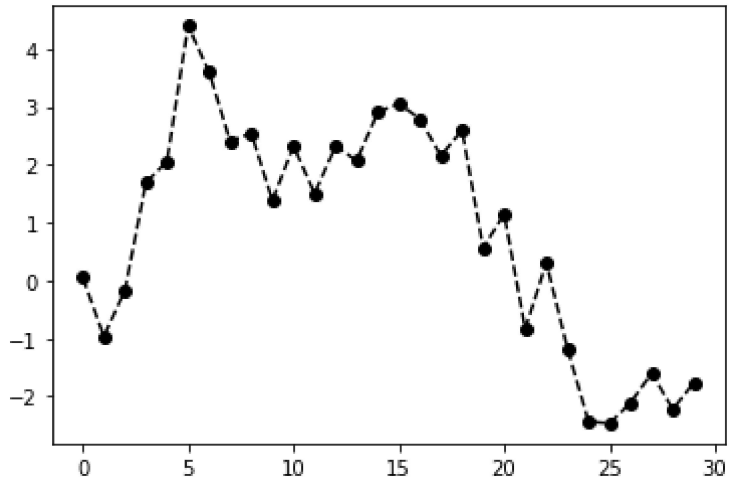


In [58]:
```
fig, axes = plt.subplots(2, 2,sharex=True, sharey=True)
for i in range(2):
    for j in range(2):
        axes[i, j].hist(np.random.randn(500), bins=50, color='k', alpha=0.5)
plt.subplots_adjust() #wspace=0, hspace=0)
```



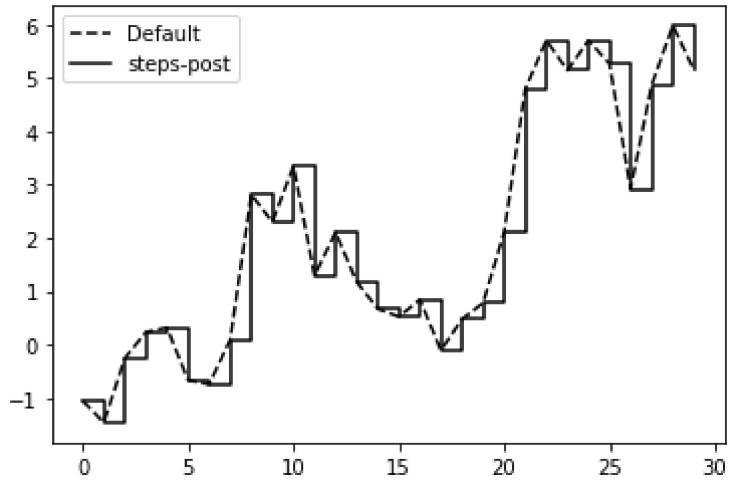
In [61]:
```
from numpy.random import randn
#plt.plot(randn(30).cumsum(), 'ko--')
plt.plot(randn(30).cumsum(), color='k', linestyle='dashed', marker='o')
#plt.close('all')
```

Out[61]:
```
[<matplotlib.lines.Line2D at 0x1c98fdcab50>]
```



In [64]:
```
data = np.random.randn(30).cumsum()
plt.plot(data, 'k--', label='Default')
plt.plot(data, 'k-', drawstyle='steps-post', label='steps-post')
plt.legend(loc='best')
```

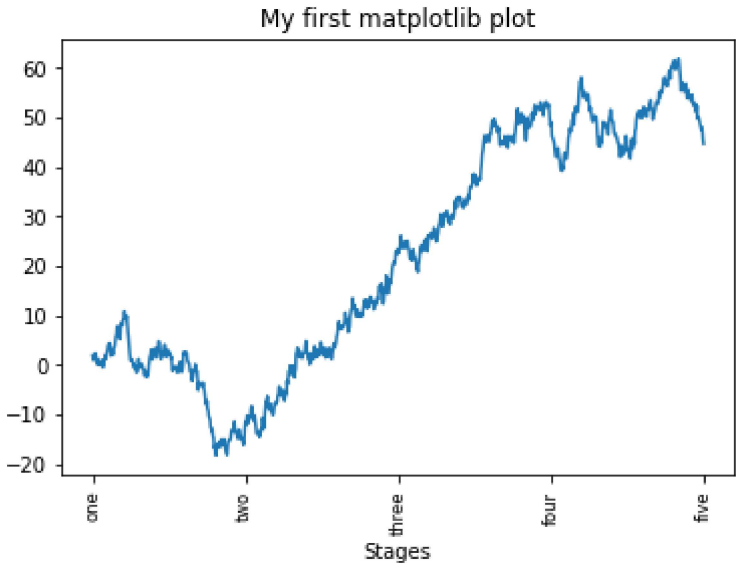
Out[64]:
```
<matplotlib.legend.Legend at 0x1c98ff18fd0>
```



In [66]:
```
#Ticks, Labels, and Legends
```

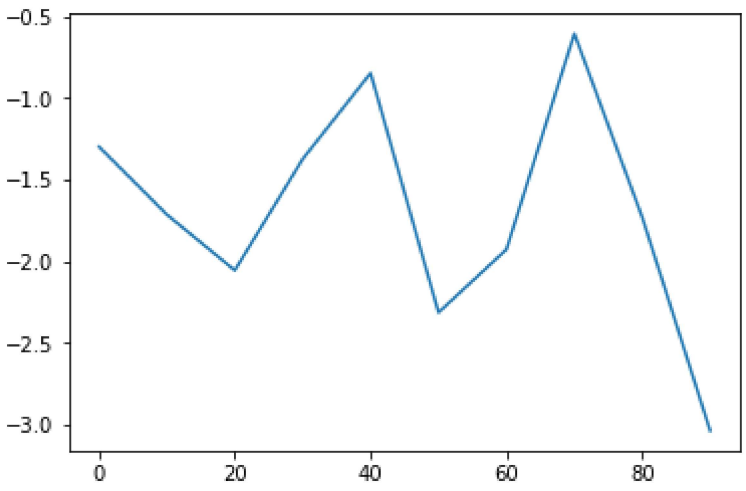
```
#Setting the title, axis labels, ticks, and ticklabels
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(np.random.randn(1000).cumsum())
ticks = ax.set_xticks([0, 250, 500, 750, 1000])
labels = ax.set_xticklabels(['one', 'two', 'three', 'four', 'five'],
                            rotation=90, fontsize='small')
#ax.set_title('My first matplotlib plot')
#ax.set_xlabel('Stages')
props = { 'title': 'My first matplotlib plot', 'xlabel': 'Stages' }
ax.set(**props)
```

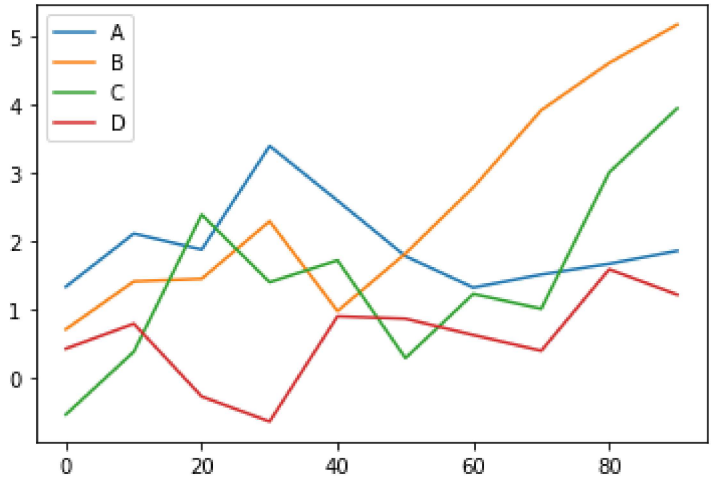
Out[66]:
```
[Text(0.5, 1.0, 'My first matplotlib plot'), Text(0.5, 0, 'Stages')]
```



In [27]:
```
#Plotting with pandas and seaborn
#Line Plots
import pandas as pd
plt.close('all')
s = pd.Series(np.random.randn(10).cumsum(), index=np.arange(0, 100, 10))
s.plot()
df = pd.DataFrame(np.random.randn(10, 4).cumsum(0),
                  columns=['A', 'B', 'C', 'D'],
                  index=np.arange(0, 100, 10))
df.plot()
```

Out[27]:
```
<AxesSubplot:>
```





In [67]:
```
np.random.randn(10, 4)
```

Out[67]:
```
array([[ 0.18245809, -1.32811723, -0.10901602,  1.20444721],
       [-2.11586623, -0.17679549, -2.15728855,  1.74228335],
       [-0.54034815, -0.39059538,  0.54030198, -0.29786307],
       [-1.11885879,  1.06856935, -2.01904623,  0.4771608 ],
       [-2.59236724,  0.65907743,  0.02430087,  0.50700941],
       [ 0.64545113,  0.34356568, -1.36073966, -0.10306488],
       [-1.74073717, -0.17990327,  0.27871154,  0.25880721],
       [ 0.50220751,  1.57877825,  0.36472353,  0.93906044],
       [-0.84882377, -1.7841542 ,  0.55558386, -2.09690213],
       [ 0.38979957, -0.1801363 , -1.37198452,  0.81067169]])
```

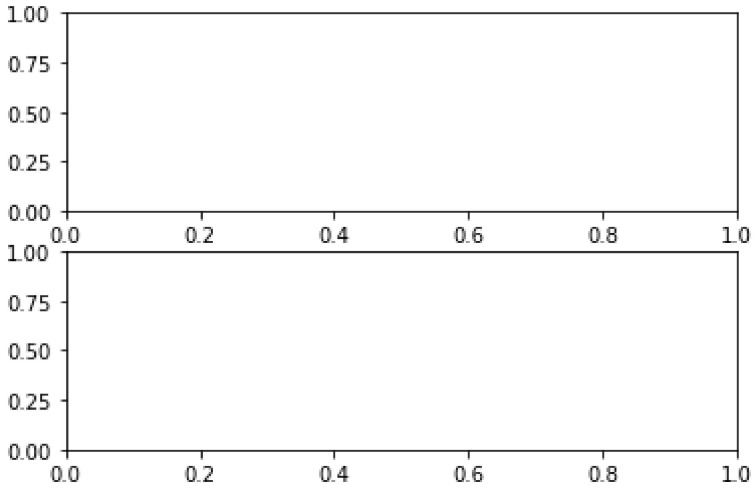
In [69]:
```
#Bar Plots
fig, axes = plt.subplots(2, 1)
data = pd.Series(np.random.rand(16), index=list('abcdefghijklmnop'))
data.plot.bar(ax=axes[0], color='k', alpha=0.7)

data.plot.barh(ax=axes[1], color='k', alpha=0.7)
np.random.seed(12348)

df = pd.DataFrame(np.random.rand(6, 4),
                  index=['one', 'two', 'three', 'four', 'five', 'six'],
                  columns=pd.Index(['A', 'B', 'C', 'D'], name='Genus'))
df
df.plot.bar()
plt.figure()
```

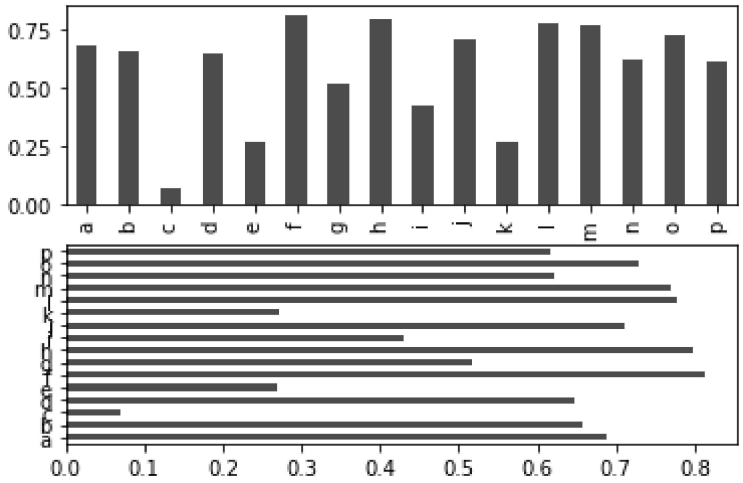
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [69], in <cell line: 3>()
      1 #Bar Plots
      2 fig, axes = plt.subplots(2, 1)
----> 3 data = pd.Series(np.random.rand(16), index=list('abcdefghijklmnop'))
      4 data.plot.bar(ax=axes[0], color='k', alpha=0.7)
      6 data.plot.barh(ax=axes[1], color='k', alpha=0.7)

NameError: name 'pd' is not defined
```



In [74]:
```
#Bar Plots
import pandas as pd
fig, axes = plt.subplots(2, 1)
data = pd.Series(np.random.rand(16), index=list('abcdefghijklmnop'))
data.plot.bar(ax=axes[0], color='k', alpha=0.7)
data.plot.barh(ax=axes[1], color='k', alpha=0.7)
```

Out[74]:
```
<AxesSubplot:>
```



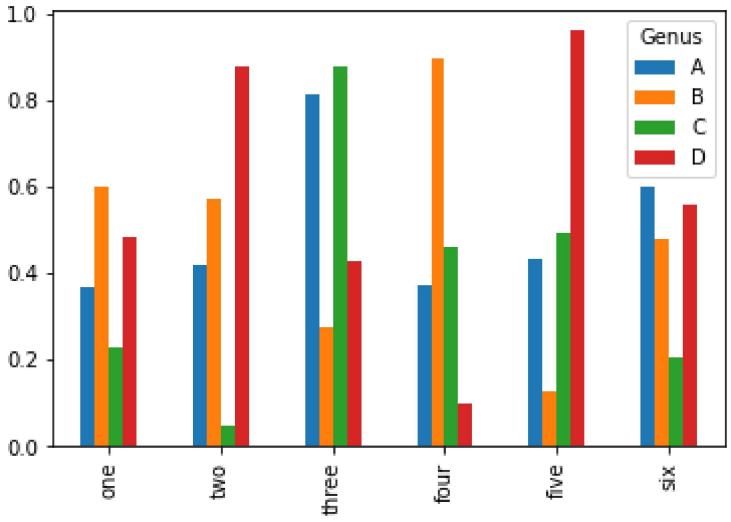
In [30]:
```
np.random.seed(12348)
df = pd.DataFrame(np.random.rand(6, 4),
                  index=['one', 'two', 'three', 'four', 'five', 'six'],
                  columns=pd.Index(['A', 'B', 'C', 'D'], name='Genus'))
df
#df.plot.bar()
#plt.figure()
```

Out[30]:

Genus	A	B	C	D
one	0.370670	0.602792	0.229159	0.486744
two	0.420082	0.571653	0.049024	0.880592
three	0.814568	0.277160	0.880316	0.431326
four	0.374020	0.899420	0.460304	0.100843
five	0.433270	0.125107	0.494675	0.961825
six	0.601648	0.478576	0.205690	0.560547

In [31]:
```
df.plot.bar()
```

Out[31]:
```
<AxesSubplot:>
```



In [32]:
```
plt.figure()
```

Out[32]:
```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

In [75]:
```
plt.close('all')
tips = pd.read_csv('tips.csv')
print(tips.head())
party_counts = pd.crosstab(tips['day'], tips['size'])
print(party_counts)
# Not many 1- and 6-person parties
#party_counts = party_counts.loc[:, 2:5]
# Normalize to sum to 1
#party_pcts = party_counts.div(party_counts.sum(1), axis=0)
#party_pcts
#party_pcts.plot.bar()
#plt.close('all')
```

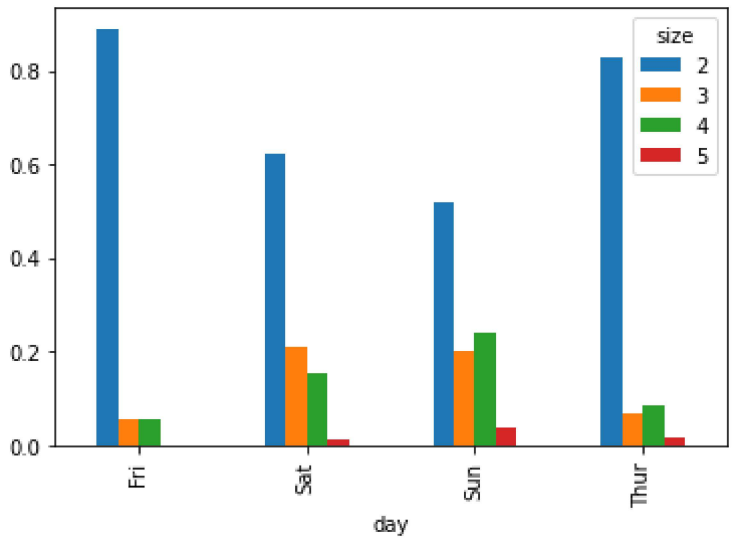
```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4
size  1   2   3   4  5  6
day
Fri   1  16   1   1  0  0
Sat   2  53  18  13  1  0
Sun   0  39  15  18  3  1
Thur  1  48   4   5  1  3
```

In [76]:
```
#party_counts = pd.crosstab(tips['day'], tips['size'])
#party_counts
# Not many 1- and 6-person parties
party_counts = party_counts.loc[:, 2:5]
# Normalize to sum to 1
party_pcts = party_counts.div(party_counts.sum(1), axis=0)
print(party_pcts)
party_pcts.plot.bar()
#plt.close('all')
```

```
size         2         3         4         5
day
Fri   0.888889  0.055556  0.055556  0.000000
Sat   0.623529  0.211765  0.152941  0.011765
Sun   0.520000  0.200000  0.240000  0.040000
Thur  0.827586  0.068966  0.086207  0.017241
```

Out[76]:

```
<AxesSubplot:xlabel='day'>
```



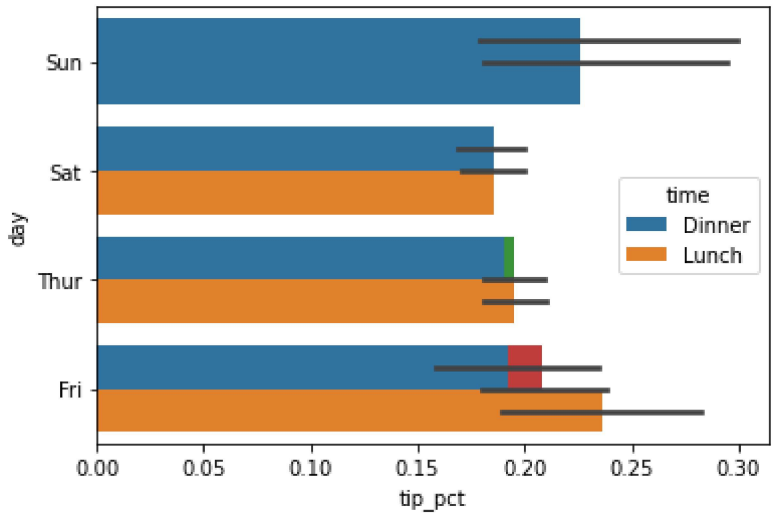
In [77]:

```
import seaborn as sns
tips['tip_pct'] = tips['tip'] / (tips['total_bill'] - tips['tip'])
print(tips.head())
sns.barplot(x='tip_pct', y='day', data=tips, orient='h')
#plt.close('all')
sns.barplot(x='tip_pct', y='day', hue='time', data=tips, orient='h')
#plt.close('all')
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A
NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected
version 1.23.1
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
   total_bill   tip     sex smoker  day    time  size   tip_pct
0       16.99  1.01  Female     No  Sun  Dinner     2  0.063204
1       10.34  1.66    Male     No  Sun  Dinner     3  0.191244
2       21.01  3.50    Male     No  Sun  Dinner     3  0.199886
3       23.68  3.31    Male     No  Sun  Dinner     2  0.162494
4       24.59  3.61  Female     No  Sun  Dinner     4  0.172069
```

Out[77]:

```
<AxesSubplot:xlabel='tip_pct', ylabel='day'>
```



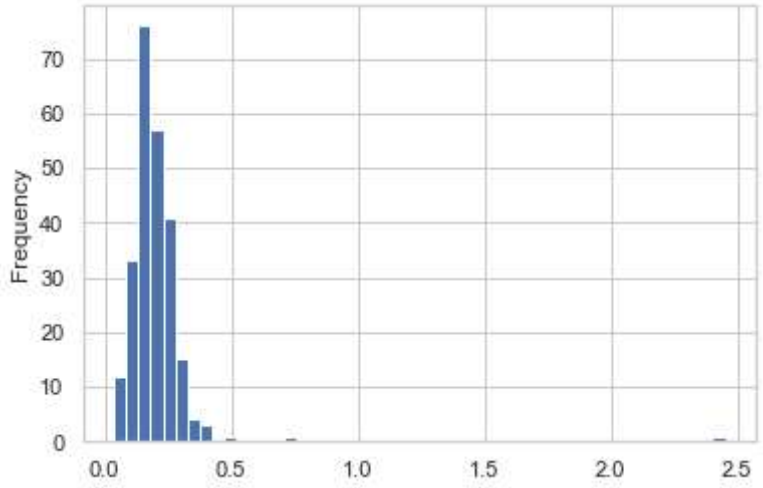
In [79]:

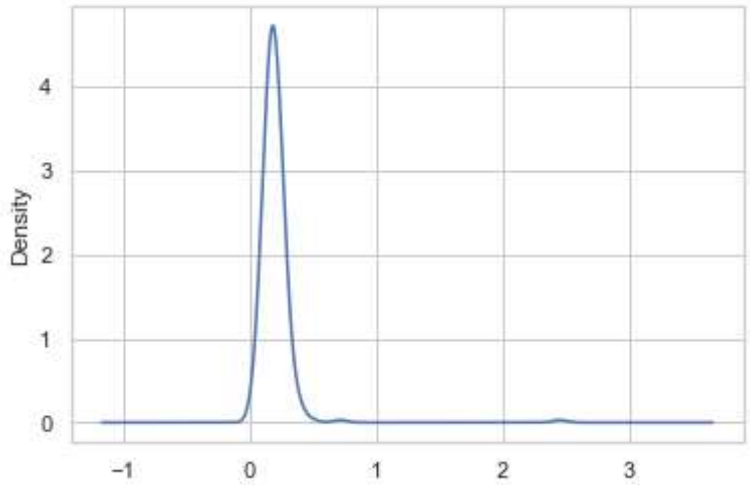
```
sns.set(style="whitegrid")
```

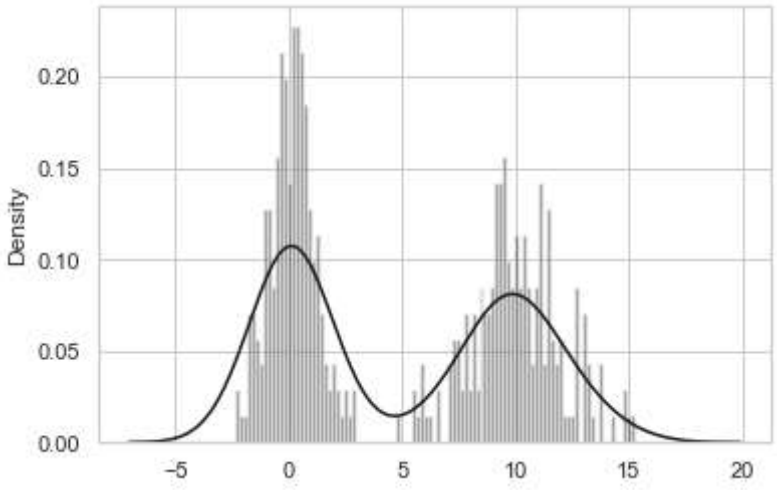
In [80]:
```
#Histograms and Density Plots
plt.figure()
tips['tip_pct'].plot.hist(bins=50)
plt.figure()
tips['tip_pct'].plot.density()
plt.figure()
comp1 = np.random.normal(0, 1, size=200)
comp2 = np.random.normal(10, 2, size=200)
values = pd.Series(np.concatenate([comp1, comp2]))
sns.distplot(values, bins=100, color='k')
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[80]:
```
<AxesSubplot:ylabel='Density'>
```







In [81]:
```
#Scatter or Point Plots
macro = pd.read_csv('Documents/macrodata.csv')
data = macro[['cpi', 'm1', 'tbilrate', 'unemp']]
trans_data = np.log(data).diff().dropna()
trans_data[-5:]
plt.figure()
sns.regplot('m1', 'unemp', data=trans_data)
plt.title('Changes in log %s versus log %s' % ('m1', 'unemp'))
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Input In [81], in <cell line: 2>()
      1 #Scatter or Point Plots
----> 2 macro = pd.read_csv('Documents/macrodata.csv')
      3 data = macro[['cpi', 'm1', 'tbilrate', 'unemp']]
      4 trans_data = np.log(data).diff().dropna()

File ~\anaconda3\lib\site-packages\pandas\util\_decorators.py:311, in deprecate_no
nkeyword_arguments.<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    305 if len(args) > num_allow_args:
    306     warnings.warn(
    307         msg.format(arguments=arguments),
    308         FutureWarning,
    309         stacklevel=stacklevel,
    310     )
--> 311 return func(*args, **kwargs)

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:680, in read_csv(f
ilepath_or_buffer, sep, delimiter, header, names, index_col, usecols, squeeze, pre
fix, mangle_dupe_cols, dtype, engine, converters, true_values, false_values, skipi
nitialspace, skiprows, skipfooter, nrows, na_values, keep_default_na, na_filter, v
erbose, skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_
parser, dayfirst, cache_dates, iterator, chunksize, compression, thousands, decima
l, lineterminator, quotechar, quoting, doublequote, escapechar, comment, encoding,
encoding_errors, dialect, error_bad_lines, warn_bad_lines, on_bad_lines, delim_whi
tespace, low_memory, memory_map, float_precision, storage_options)
    665 kwds_defaults = _refine_defaults_read(
    666     dialect,
    667     delimiter,
   (...)
    676     defaults={"delimiter": ","},
    677 )
    678 kwds.update(kwds_defaults)
--> 680 return _read(filepath_or_buffer, kwds)

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:575, in _read(file
path_or_buffer, kwds)
    572 _validate_names(kwds.get("names", None))
    574 # Create the parser.
--> 575 parser = TextFileReader(filepath_or_buffer, **kwds)
    577 if chunksize or iterator:
    578     return parser

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:933, in TextFileRe
ader.__init__(self, f, engine, **kwds)
    930     self.options["has_index_names"] = kwds["has_index_names"]
    932 self.handles: IOHandles | None = None
--> 933 self._engine = self._make_engine(f, self.engine)

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:1217, in TextFileR
eader._make_engine(self, f, engine)
   1213     mode = "rb"
   1214 # error: No overload variant of "get_handle" matches argument types
   1215 # "Union[str, PathLike[str], ReadCsvBuffer[bytes], ReadCsvBuffer[str]]"
   1216 # , "str", "bool", "Any", "Any", "Any", "Any", "Any"
-> 1217 self.handles = get_handle(  # type: ignore[call-overload]
   1218     f,
   1219     mode,
   1220     encoding=self.options.get("encoding", None),
   1221     compression=self.options.get("compression", None),
   1222     memory_map=self.options.get("memory_map", False),
   1223     is_text=is_text,
   1224     errors=self.options.get("encoding_errors", "strict"),
```

```
   1225     storage_options=self.options.get("storage_options", None),
   1226 )
   1227 assert self.handles is not None
   1228 f = self.handles.handle

File ~\anaconda3\lib\site-packages\pandas\io\common.py:789, in get_handle(path_or_
buf, mode, encoding, compression, memory_map, is_text, errors, storage_options)
    784 elif isinstance(handle, str):
    785     # Check whether the filename is to be opened in binary mode.
    786     # Binary mode does not support 'encoding' and 'newline'.
    787     if ioargs.encoding and "b" not in ioargs.mode:
    788         # Encoding
--> 789         handle = open(
    790             handle,
    791             ioargs.mode,
    792             encoding=ioargs.encoding,
    793             errors=errors,
    794             newline="",
    795         )
    796     else:
    797         # Binary mode
    798         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory: 'Documents/macrodata.csv'
```

In [91]:
```
sns.pairplot(trans_data, diag_kind='kde', plot_kws={'alpha': 0.52})
```

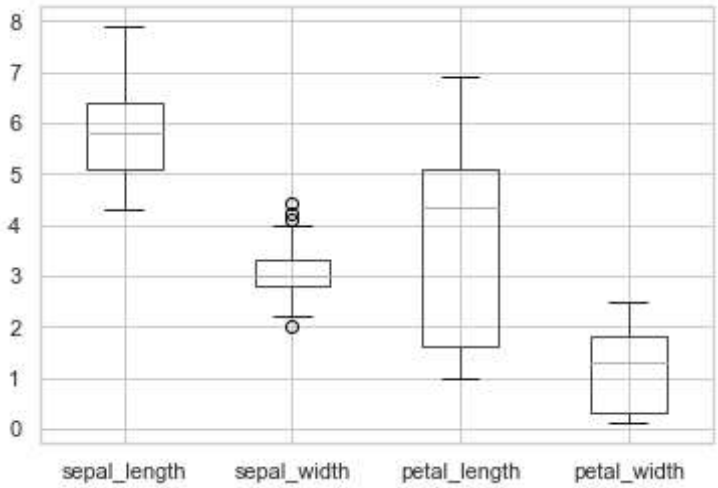
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [91], in <cell line: 1>()
----> 1 sns.pairplot(trans_data, diag_kind='kde', plot_kws={'alpha': 0.52})

NameError: name 'trans_data' is not defined
```

In [83]:
```
ir=pd.read_csv('IRIS.csv')
```

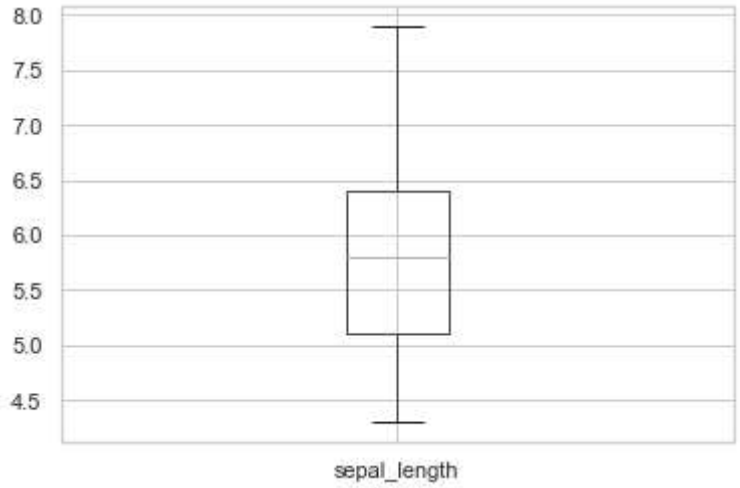
In [84]:
```
ir.boxplot()
```

Out[84]:
```
<AxesSubplot:>
```



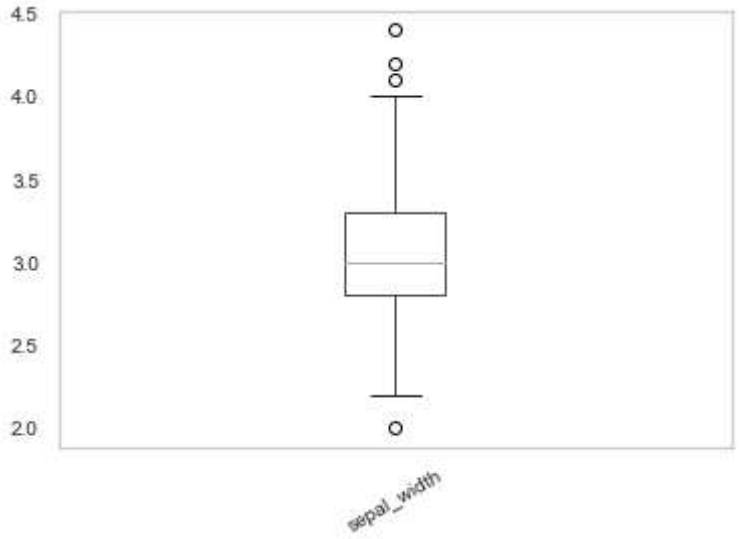
In [85]:
```
ir.boxplot('sepal_length')
```

Out[85]:
```
<AxesSubplot:>
```



In [89]:
```
ir.boxplot('sepal_width', grid=False,rot=30,fontsize=10)
```

Out[89]:
```
<AxesSubplot:>
```



In [90]:
```
ir.describe()
```

Out[90]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [ ]:
```

```