# APSSDC

**Andhra Pradesh State Skill Development Corporation**

(https://apssdc.in)

# Data Analysis Using Python Day11

## Data Visualization using Matplotlib and Seaborn

- Normalization
- Data Imputation - Fill the Missing Values - mean, median, most frequent, constant
- Matplotlib
- line Plot - identify the changes in the data
- Scatter - find the relationship between 2 variable
- Bar Graph - count Categorical data
- Text

```
## Day 11 Objectives
- Sub plots
- Bar Graphs
- Histogram
- Pie chart
- Box plot
- Seaborn
- Styles
- Color Palletes
```

## Sub Plots

In [15]:

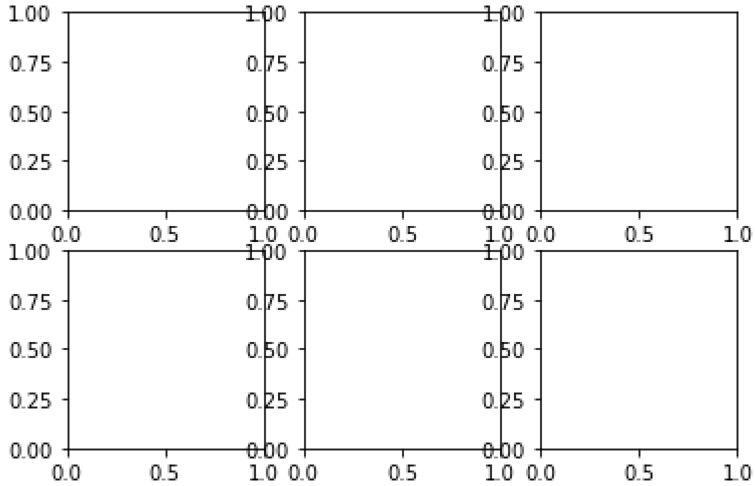
```
import matplotlib.pyplot as plt
```

In [16]:

```
import numpy as np
import pandas as pd
```

In [17]:

```
ax = plt.subplots(2, 3) # row, colum

plt.show()
```



In [18]:

```
help(plt.subplots)
```

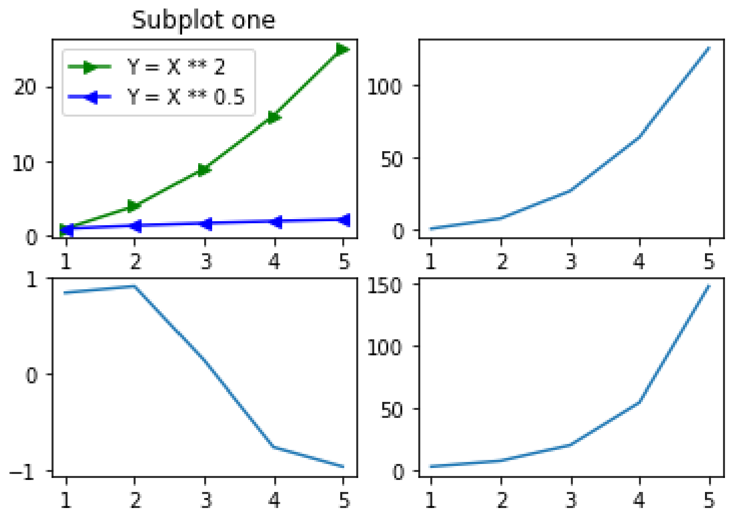
```
Help on function subplots in module matplotlib.pyplot:

subplots(nrows=1, ncols=1, sharex=False, sharey=False, squeeze=True, subpl
ot_kw=None, gridspec_kw=None, **fig_kw)
    Create a figure and a set of subplots.

    This utility wrapper makes it convenient to create common layouts of
    subplots, including the enclosing figure object, in a single call.

    Parameters
    ----------
    nrows, ncols : int, optional, default: 1
        Number of rows/columns of the subplot grid.

    sharex, sharey : bool or {'none', 'all', 'row', 'col'}, default: False
        Controls sharing of properties among x (`sharex`) or y (`sharey`)
        axes:

        - True or 'all': x- or y-axis will be shared among all subplots.
```

In [19]:

```
x = np.array([1, 2, 3, 4, 5])

plt.subplot(2, 2, 1) #row, column, index
plt.plot(x, x **2, marker = '>', c = 'g')
plt.plot(x, x ** 0.5, marker = '<', c = 'b')
plt.xlabel('Y = X **2')
plt.title('Subplot one')
plt.legend(['Y = X ** 2', 'Y = X ** 0.5'])
plt.subplot(2,2,2)
plt.plot(x, x ** 3)
plt.subplot(2,2,3)
plt.plot(x, np.sin(x))
plt.subplot(2,2,4)
plt.plot(x, np.exp(x))
```

Out[19]:

```
[<matplotlib.lines.Line2D at 0x19fe348e970>]
```



In [20]:

```
help(plt.bar)
```

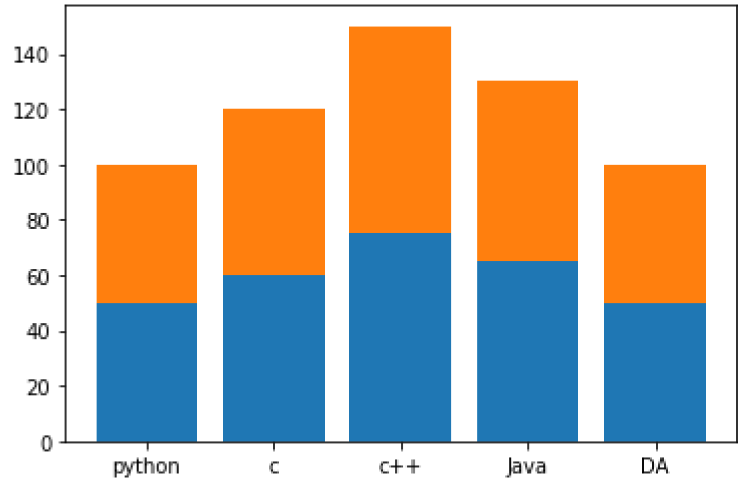
```
Help on function bar in module matplotlib.pyplot:

bar(x, height, width=0.8, bottom=None, *, align='center', data=None, **kwa
rgs)
    Make a bar plot.

    The bars are positioned at *x* with the given *align*\ment. Their
    dimensions are given by *width* and *height*. The vertical baseline
    is *bottom* (default 0).

    Each of *x*, *height*, *width*, and *bottom* may either be a scalar
    applying to all bars, or it may be a sequence of length N providing a
    separate value for each bar.

    Parameters
    ----------
    x : sequence of scalars
        The x coordinates of the bars. See also *align* for the
        alignment of the bars to the coordinates.
```

In [21]:

```
sub = ['python', 'c', 'c++', 'Java', 'DA']
c1 = [55, 65, 80, 70, 60]
c2 = [50, 60, 75, 65, 50]

plt.bar(sub, c1)
plt.bar(sub, c2, bottom = c2)
```

Out[21]:

```
<BarContainer object of 5 artists>
```

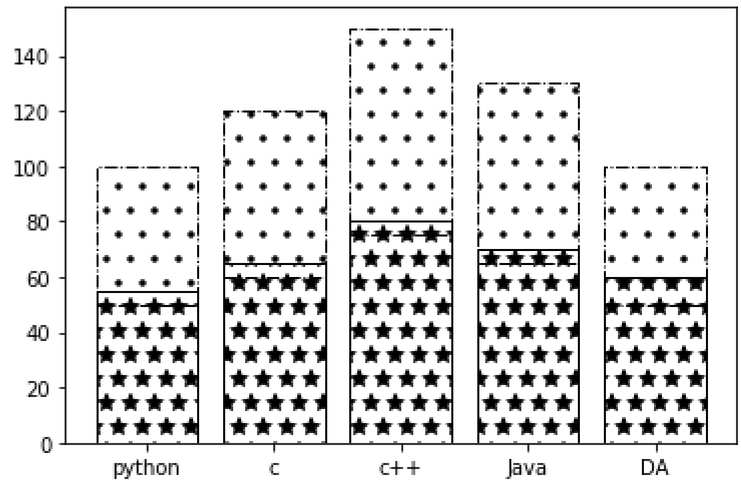


In [22]:

```
sub = ['python', 'c', 'c++', 'Java', 'DA']
c1 = [55, 65, 80, 70, 60]
c2 = [50, 60, 75, 65, 50]

plt.bar(sub, c1, fill = False, hatch = '*')
plt.bar(sub, c2, bottom = c2, fill = False, hatch = '.', linestyle = '-.')
```

Out[22]:

```
<BarContainer object of 5 artists>
```



In [23]:

```python
help(plt.barh)
```

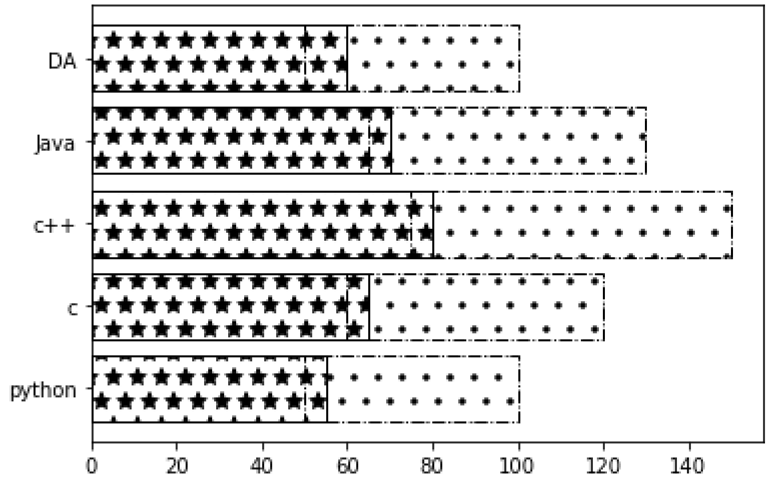
```
Help on function barh in module matplotlib.pyplot:

barh(y, width, height=0.8, left=None, *, align='center', **kwargs)
    Make a horizontal bar plot.

    The bars are positioned at *y* with the given *align*\ment. Their
    dimensions are given by *width* and *height*. The horizontal baseline
    is *left* (default 0).

    Each of *y*, *width*, *height*, and *left* may either be a scalar
    applying to all bars, or it may be a sequence of length N providing a
    separate value for each bar.

    Parameters
    ----------
    y : scalar or array-like
        The y coordinates of the bars. See also *align* for the
        alignment of the bars to the coordinates.
```

In [24]:

```python
sub = ['python', 'c', 'c++', 'Java', 'DA']
c1 = [55, 65, 80, 70, 60]
c2 = [50, 60, 75, 65, 50]

plt.barh(sub, c1, fill = False, hatch = '*')
plt.barh(sub, c2, left = c2, fill = False, hatch = '.', linestyle = '-.')
```

Out[24]:

```
<BarContainer object of 5 artists>
```



In [25]:

```
help(plt.hist)
```

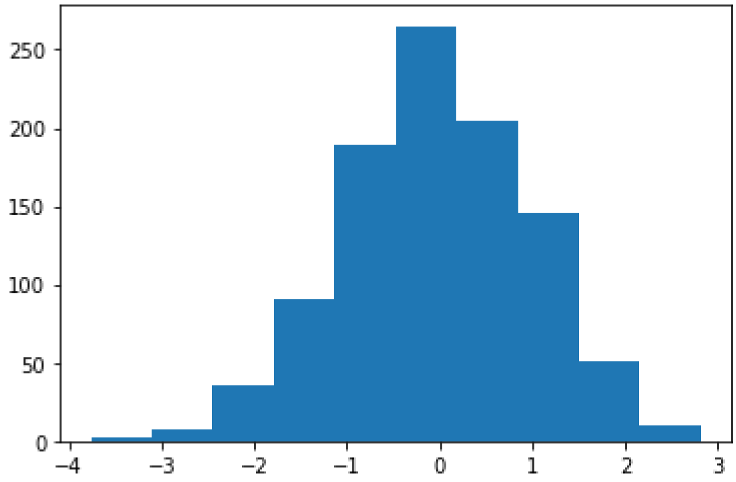
```
    Compute and draw the histogram of *x*.  The return value is a tuple
    (*n*, *bins*, *patches*) or ([*n0*, *n1*, ...], *bins*, [*patches0*,
    *patches1*,...]) if the input contains multiple data.  See the
    documentation of the *weights* parameter to draw a histogram of
    already-binned data.

    Multiple data can be provided via *x* as a list of datasets
    of potentially different length ([*x0*, *x1*, ...]), or as
    a 2-D ndarray in which each column is a dataset.  Note that
    the ndarray form is transposed relative to the list form.

    Masked arrays are not supported.

    The *bins*, *range*, *weights*, and *density* parameters behave as in
    `numpy.histogram`.

    Parameters
    ----------
    x : (n,) array or sequence of (n,) arrays
        Input values, this takes either a single array or a sequence of
```

In [26]:

```
x = np.random.randn(1000)
plt.hist(x)
```

Out[26]:

```
(array([  2.,   7.,  35.,  90., 189., 265., 205., 146.,  51.,  10.]),
 array([-3.76680373, -3.10851272, -2.45022171, -1.79193069, -1.13363968,
        -0.47534867,  0.18294234,  0.84123335,  1.49952437,  2.15781538,
         2.81610639]),
 <a list of 10 Patch objects>)
```



In [67]:

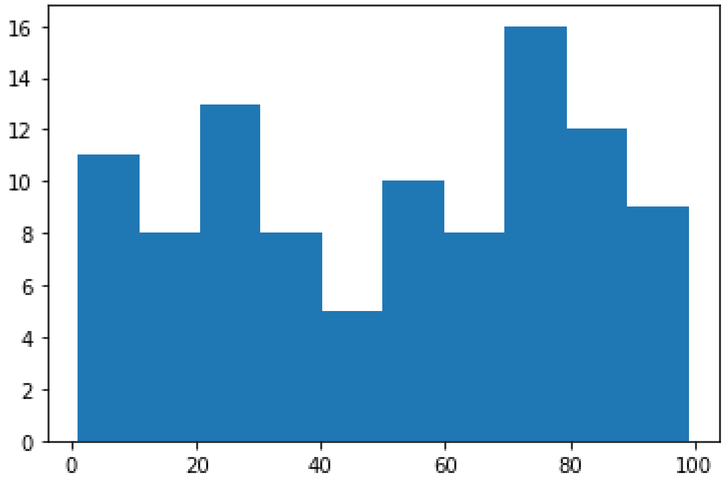
```
python = np.random.randint(1, 100, 100)
```

In [68]:

```
plt.hist(python)
```

Out[68]:

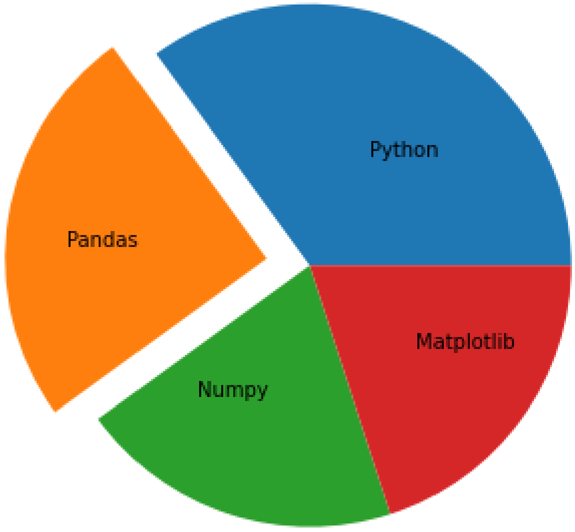
```
(array([11.,  8., 13.,  8.,  5., 10.,  8., 16., 12.,  9.]),
 array([ 1. , 10.8, 20.6, 30.4, 40.2, 50. , 59.8, 69.6, 79.4, 89.2, 99. ]),
 <a list of 10 Patch objects>)
```



## Pie Chart

In [81]:

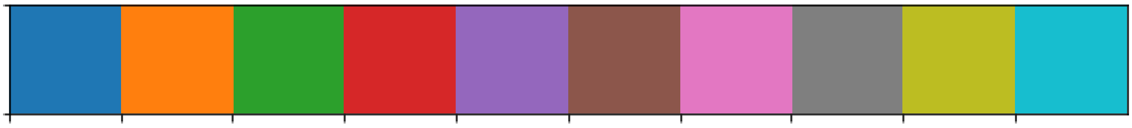
```
sub = ['Python', 'Pandas', 'Numpy', 'Matplotlib']
y = [35, 25, 20, 20]

plt.pie(y, labels=sub,labeldistance=0.5, radius=1.5, explode = [0, 0.25,0,0])
plt.show()
```



# Seaborn - Color Palletes

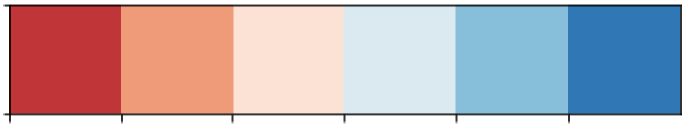
In [27]:

```
#Calling with no arguments returns all colors from the current default color cycle:
#Here, the palplot() is used to plot the array of colors horizontally
import seaborn as sns
sns.palplot(sns.color_palette())
```



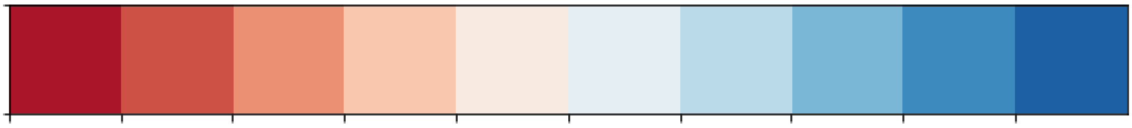
In [28]:

```
from matplotlib import pyplot as plt
current_palette = sns.color_palette("RdBu")
sns.palplot(current_palette)
plt.show()
```



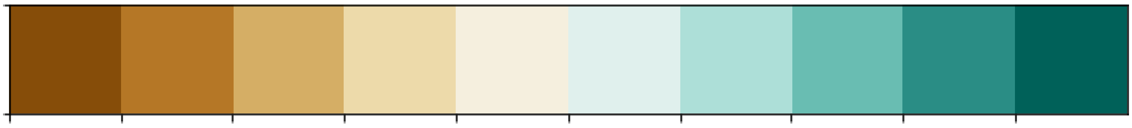
In [29]:

```
sns.palplot(sns.color_palette("RdBu", n_colors=10))
```



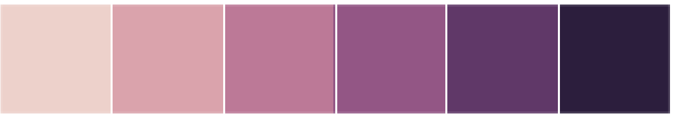
In [30]:

```
#Diverging palettes use two different colors.
#Each color represents variation in the value ranging from a common point in either directi

#Assume plotting the data ranging from -1 to 1.
#The values from -1 to 0 takes one color and 0 to +1 takes another color.
sns.palplot(sns.color_palette("BrBG", 10))
```



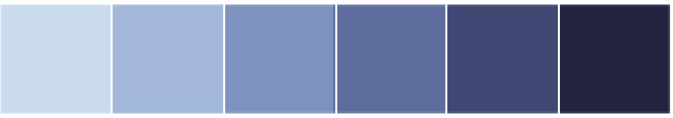
In [31]:

```
#Customized cubixhelix

sns.set()
sns.palplot(sns.cubehelix_palette())
```



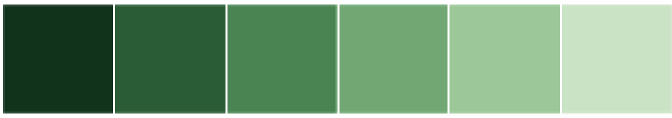
In [32]:

```
sns.palplot(sns.cubehelix_palette(rot=-0.1))
```



In [33]:

```
sns.palplot(sns.cubehelix_palette(start=2.1, rot=-.1,reverse = True))
```



In [34]:

```
sns.palplot(sns.cubehelix_palette(reverse=True))
```