औद्योगिक प्रशिक्षण के लिए राष्ट्रीय संस्थान

National Institute for Industrial Training

One Premier Organization with Non Profit Status | Registered Under Govt. of WB
Inspired By: National Task Force on IT & SD Govt. of India

WITH

Subject -> Python with Artificial Intelligence

Submitted By -> Rohit Shaw

Submitted To -> Sayantan Sir

CONTENTS

Acknowledgement

I would like to express my deepest appreciation to all those who provided me the possibility to complete this project. A special gratitude I give to my project guide, Mr. Sayantan Chakraborty and Ankit Pramanik whose contribution in stimulating suggestions and encouragement helped me to coordinate my project.

Furthermore I would also like to acknowledge with much appreciation the crucial role of the 'National Institute for Industrial Training', which gave the permission to use all required equipment and the necessary materials to complete the course "*Artificial Intelligence with Python*".

I have taken efforts in this project. However, it would not have been possible without the kind support and help of above mentioned individuals and organization. I would like to extend my sincere thanks to all of them.

Student's Profile

- NAME : Rohit Shaw
- ADDRESS : 23/2 Darpa Narayan Tagore Street
- PIN_CODE : 700007
- E-MAIL : rshaw9377@gmail.com
- NATIONALITY : Indian
- DATE OF BIRTH : 07/12/1998
- SEX : Male
- LANGUAGES KNOWN : English, Hindi, Bengali and Bhojpuri
- CARRER OBJECTIVE : Inquisitive, energetic computer science specialist skilled in C, with a strong foundation in math, logic, and cross-platform coding. Seeking to leverage solid skills in collaboration, communication, and development as a programmer for more programming skills (Python , R , C++). Highly interested in ML(Machine Learning) and developing AI.

Introduction

Python comes with a huge amount of inbuilt libraries. Many of the libraries are for Artificial Intelligence and Machine Learning. Some of the libraries are Tensorflow (which is high-level neural network library), scikit-learn (for data mining, data analysis and machine learning), pylearn2 (more flexible than scikit-learn), etc. The list keeps going and never ends.

Python has an easy implementation for OpenCV. What makes Python favorite for everyone is its powerful and easy implementation. For other languages, students and researchers need to get to know the language before getting into ML or AI with that language. This is not the case with python. Even a programmer with very basic knowledge can easily handle python. Apart from that, the time someone spends on writing and debugging code in python is way less when compared to C, C++ or Java. This is exactly what the students of AI and ML want. They don't want to spend time on

debugging the code for syntax errors, they want to spend more time on their algorithms and heuristics related to AI and ML. Not just the libraries but their tutorials, handling of interfaces are easily available online. People build their own libraries and upload them on GitHub or elsewhere to be used by others.

- Python is a powerful open source programming language, which means that it's free to use while having all the properties that a programming language should have.

- It is a versatile programming language that supports Object-Oriented Programming, Structured Programming, and functional programming patterns.

Python has some 72,000 libraries in the Python Package Index that aid in scientific calculations and machine learning applications.

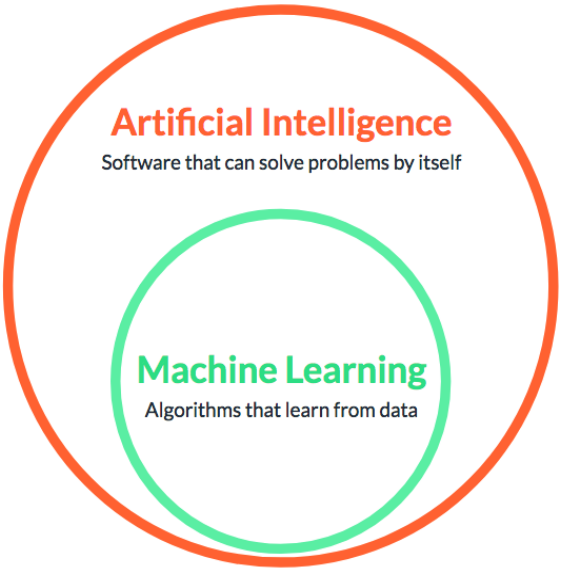
- Python sports an easy to understand and readable syntax that ensures that the development time is cut into half when compared with other programming languages.

- Python enables you to perform data analysis, data manipulation, and data visualization, which are very important in data science.

AI & ML Description

Artificial Intelligence (AI) -the broad discipline of creating intelligent machines

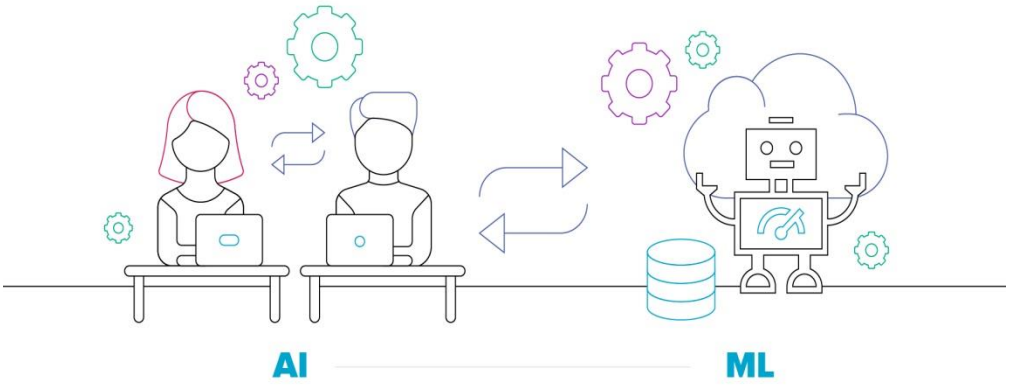
Artificial intelligence (AI) is the overarching discipline that covers anything related to make machines smart. Whether it's a robot, a refrigerator, a car, or a software application, if you are making them smart, then it's AI.



AI refers to systems that can learn from experience

Machine Learning (ML)

- is commonly used alongside AI but they are not the same thing. ML is a subset of AI.



ML refers to systems that can learn by themselves. Systems which gets smarter and smarter over time without human intervention.

Objectives

WHY Python?

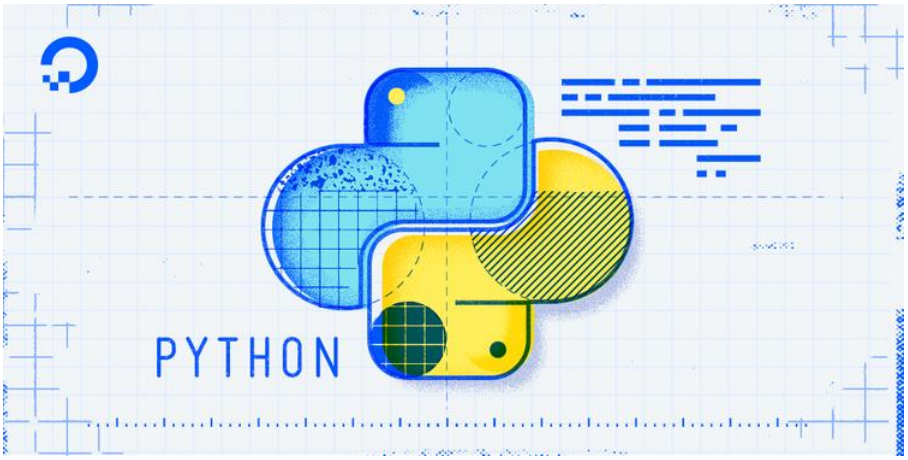
A great choice of libraries is one of the main reasons Python is the most popular programming language used for AI. A library is a module or a group of modules published by different sources like PyPi which include a pre-written piece of code that allows users to reach some functionality or perform different actions. Python libraries provide base level items so developers don't have to code them from the very beginning every time.

These are some of the most widespread libraries you can use for ML and AI:

Scikit-learn -> for handling basic ML algorithms like clustering, linear and logistic regressions, regression, classification, and others.

Pandas -> for high-level data structures and analysis. It allows merging and filtering of data, as well as gathering it from other external sources like Excel, for instance

Matplotlib -> for creating 2D plots, histograms, charts, and other forms of visualization.



Seaborn -> is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures. Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

Working in the ML and AI industry means dealing with a bunch of data that you need to process in the most convenient and effective way. The low entry barrier allows more data scientists to quickly pick up Python and start using it for AI development without wasting too much effort into learning the language.

Python programming language resembles the everyday English language, and that makes the process of learning easier. It’s simple syntax allows you to comfortably work with complex systems, ensuring clear relations between the system elements.

Machine learning and artificial intelligence-based projects are obviously what the future holds. We want better personalization, smarter recommendations, and improved search functionality. Our apps can see, hear, and respond – that’s what artificial intelligence (AI) has brought, enhancing the user experience and creating value across many industries.

Hardware and Software Requirements

1. Python(Latest Version Recommended)
2. Anaconda(Free and open-source distribution of the Python)
3. Operating System(Windows/Linux)

Future Scope

Python has become a formidable language in the data science, artificial intelligence, and machine learning spheres. This is largely due to the language's flexibility and community, but it's also a direct result of the production of many ultra-powerful, high-quality packages and modules.

Further considerations should include the situations where data science tasks (analytics, machine learning, artificial intelligence) are carried out either on a local desktop or laptop machine by a data scientist (for example), or where these tasks are performed on servers (usually in the cloud).

Advantages

The usage of Python is such that it cannot be limited to only one activity.

1. Less Code
2. Prebuilt Libraries
3. Platform Agnostic
4. Flexibility

Snapshots

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

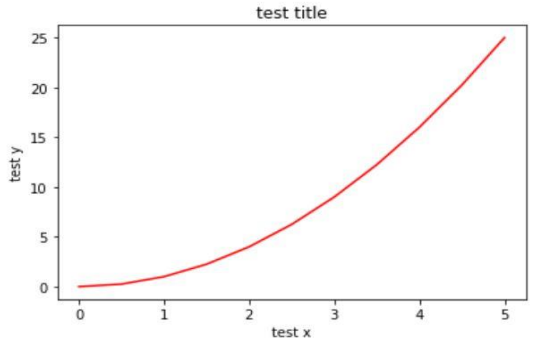
In [2]:
```
x=np.linspace(0,5,11)
x
```

Out[2]:
```
array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. ])
```

In [3]:
```
y=x**2
y
```

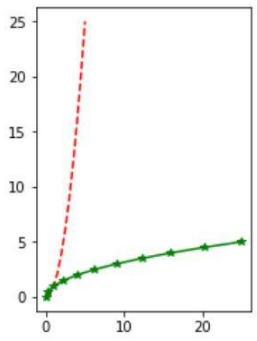
Out[3]:
```
array([ 0.  ,  0.25,  1.  ,  2.25,  4.  ,  6.25,  9.  , 12.25, 16.  ,
       20.25, 25.  ])
```

In [4]:
```
plt.plot(x,y,'r')
plt.xlabel("test x")
plt.ylabel("test y")
plt.title("test title")
plt.show()
```



In [6]:
```
plt.subplot(1,2,1)
plt.plot(x,y,'r--')
plt.subplot(1,2,1)
plt.plot(y,x,'g*-')
```

Out[6]:
```
[<matplotlib.lines.Line2D at 0x1c8e1fa8710>]
```



In [7]:
```
fig=plt.figure()
fig
```

Out[7]:
```
<Figure size 432x288 with 0 Axes>
```

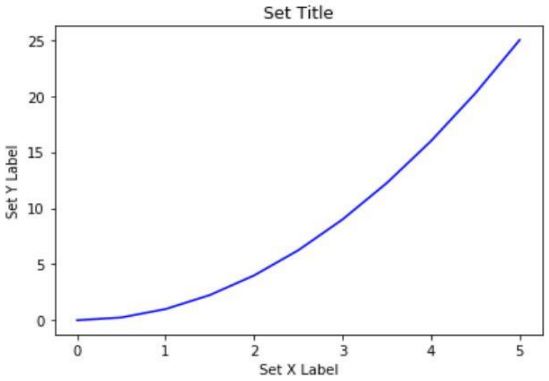
```
<Figure size 432x288 with 0 Axes>
```

In [8]:
```
axes=fig.add_axes([0.1,0.1,0.8,0.8])
axes
```

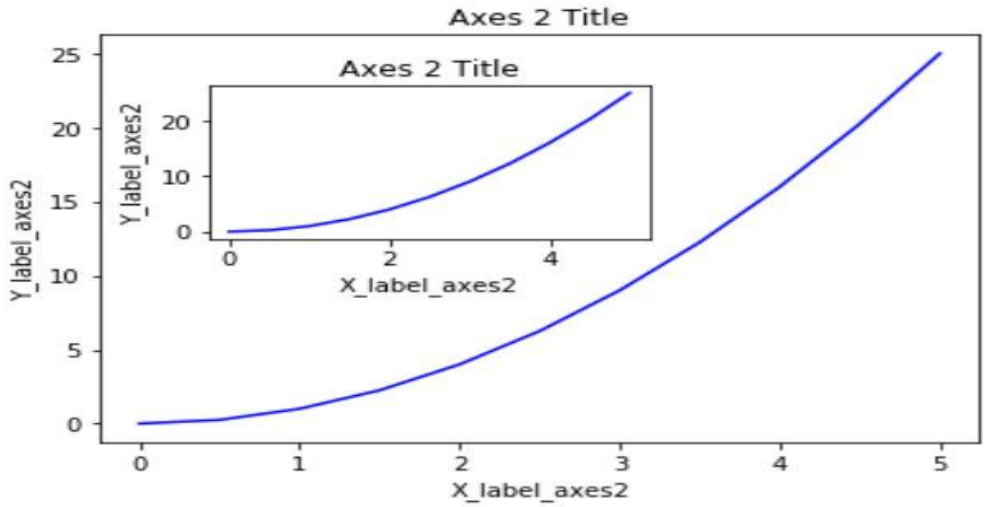
Out[8]:
```
<matplotlib.axes._axes.Axes at 0x1c8e1fcecc0>
```

In [9]:
```
fig=plt.figure()
axes=fig.add_axes([0.1,0.1,0.8,0.8])
axes.plot(x,y,'b')
axes.set_xlabel('Set X Label')
axes.set_ylabel('Set Y Label')
axes.set_title("Set Title")
```

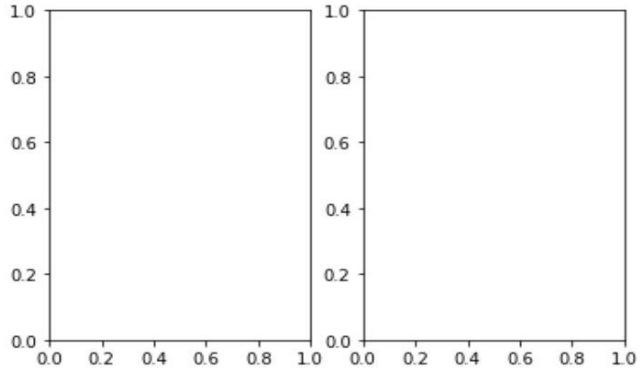
Out[9]:
```
Text(0.5, 1.0, 'Set Title')
```



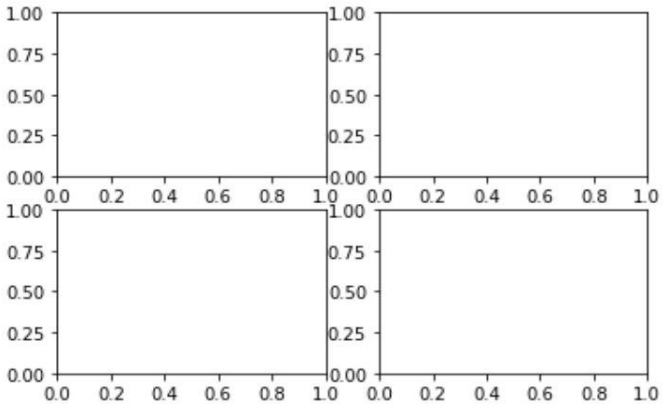
In [10]:
```
fig=plt.figure()
axes1=fig.add_axes([0.1,0.1,0.8,0.8])
axes2=fig.add_axes([0.2,0.5,0.4,0.3])

axes1.plot(x,y,'b')
axes1.set_xlabel('X_label_axes2')
axes1.set_ylabel('Y_label_axes2')
axes1.set_title('Axes 2 Title')

axes2.plot(x,y,'b')
axes2.set_xlabel('X_label_axes2')
axes2.set_ylabel('Y_label_axes2')
axes2.set_title('Axes 2 Title');
```



In [11]:
```
fig,axes=plt.subplots(nrows=1,ncols=2)
```

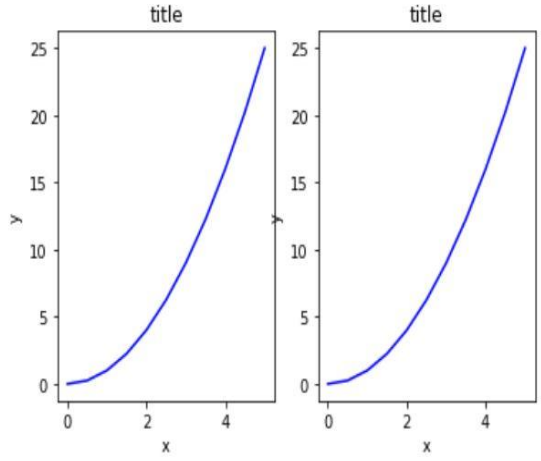


In [14]:
```
fig,axes=plt.subplots(nrows=2,ncols=2)
```



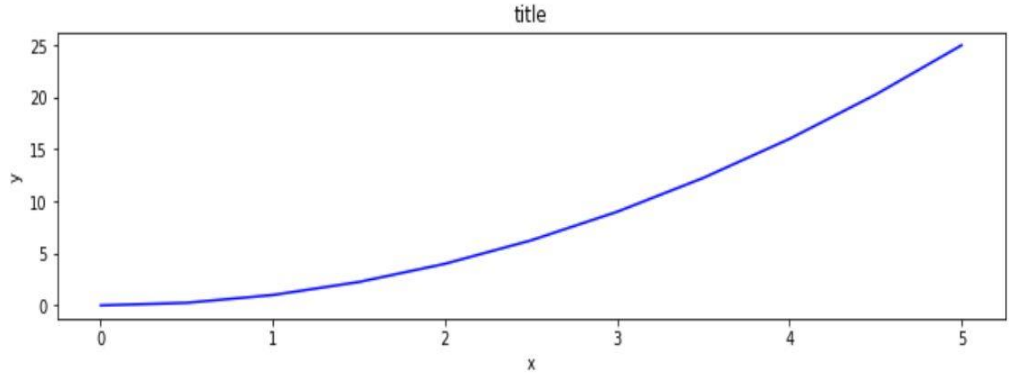
In [16]:
```
for ax in axes:
    ax.plot(x,y,'b')
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_title('title')
fig
```

Out[16]:



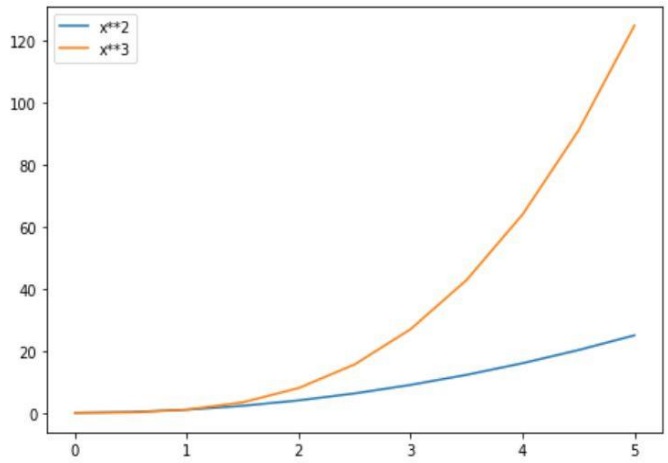
In [18]:
```
#fig=plt.figure(figsize=(8,4),dpi=100)
fig,axes=plt.subplots(figsize=(12,3))
axes.plot(x,y,'b')
axes.set_xlabel('x')
axes.set_ylabel('y')
axes.set_title('title')
```

Out[18]:
```
Text(0.5, 1.0, 'title')
```



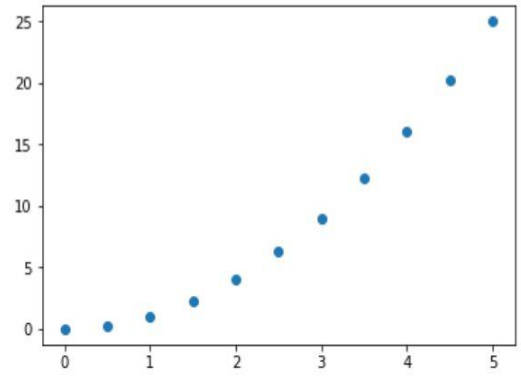
In [22]:
```
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.plot(x,x**2,label="x**2")
ax.plot(x,x**3,label="x**3")
ax.legend()
```

Out[22]: <matplotlib.legend.Legend at 0x1c8e3b32358>



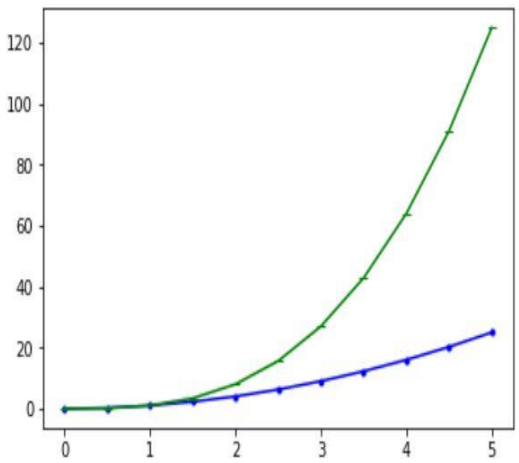
In [25]:
```
plt.scatter(x,y)
```

Out[25]: <matplotlib.collections.PathCollection at 0x1c8e22d3048>



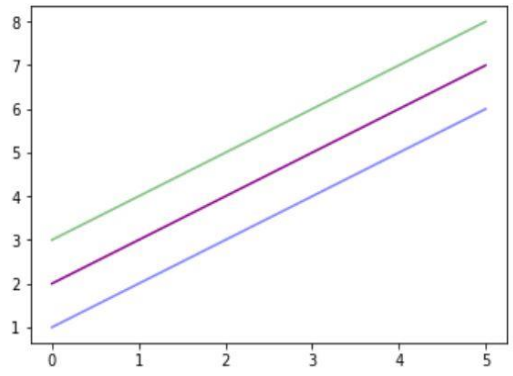
In [23]:
```
fig, ax=plt.subplots()
ax.plot(x,x**2,"b.-")
ax.plot(x,x**3,"g_-")
```

Out[23]: [<matplotlib.lines.Line2D at 0x1c8e3b99ba8>]



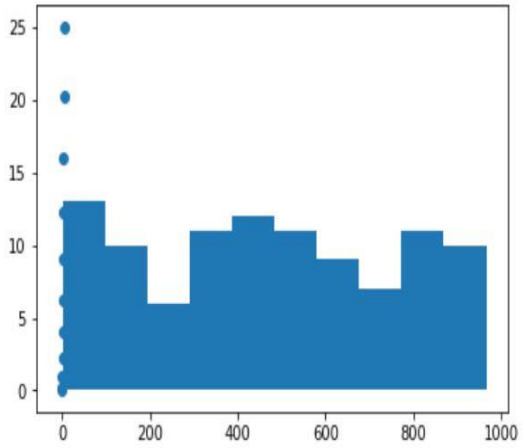
In [24]:
```
fig,ax=plt.subplots()
ax.plot(x,x+1,color="b",alpha=0.5)
ax.plot(x,x+2,color="#8B008B",alpha=1)
ax.plot(x,x+3,color="g",alpha=0.5)
```

Out[24]: [<matplotlib.lines.Line2D at 0x1c8e227e780>]

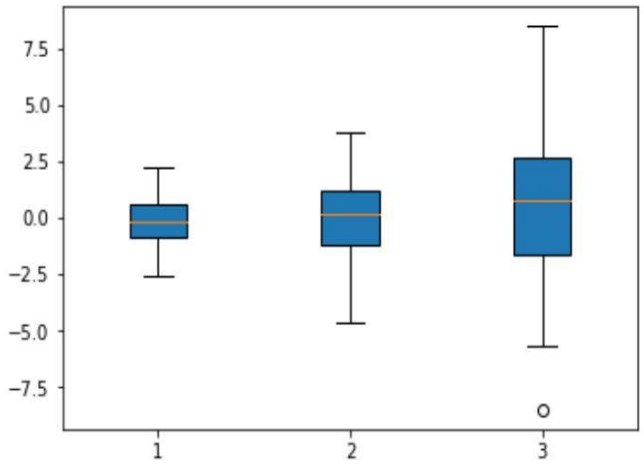


In [26]:
```
from random import sample
data=sample(range(1,1000),100)
plt.hist(data)
plt.scatter(x,y)
```

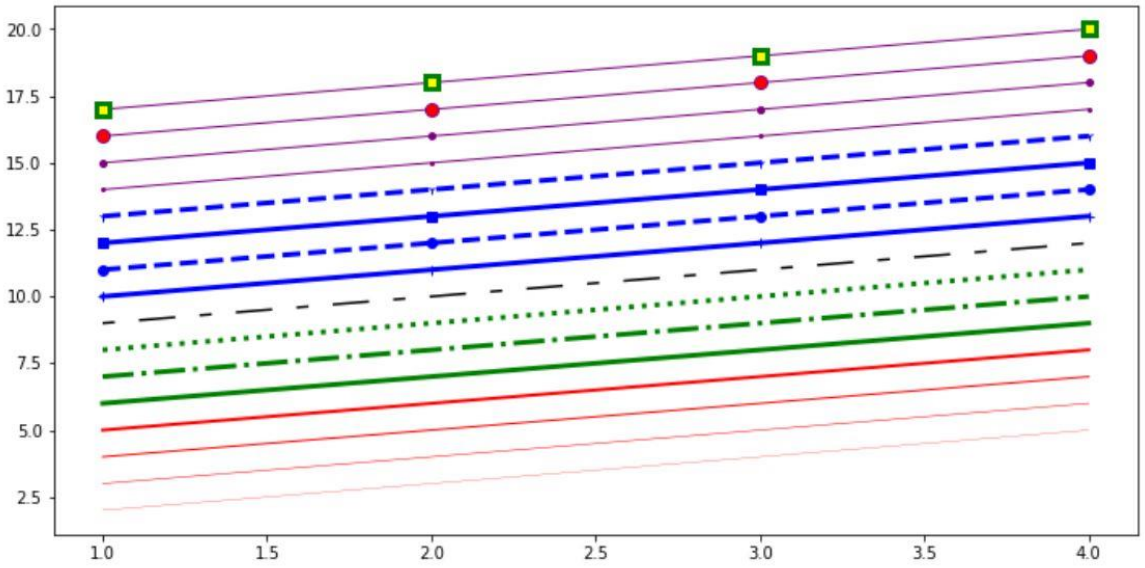
Out[26]: <matplotlib.collections.PathCollection at 0x1c8e2205198>



In [27]:
```
data=[np.random.normal(0,std,100) for std in range(1,4)]
plt.boxplot(data,vert=True,patch_artist=True);
```



In [12]:
```
fig, ax = plt.subplots(figsize=(12,6))

ax.plot(x, x+1, color="red", linewidth=0.25)
ax.plot(x, x+2, color="red", linewidth=0.50)
ax.plot(x, x+3, color="red", linewidth=1.00)
ax.plot(x, x+4, color="red", linewidth=2.00)

# possible linestype options ‘-‘, ‘–’, ‘-.’, ‘:’, ‘steps’
ax.plot(x, x+5, color="green", lw=3, linestyle='-')
ax.plot(x, x+6, color="green", lw=3, ls='-.')
ax.plot(x, x+7, color="green", lw=3, ls=':')

# custom dash
line, = ax.plot(x, x+8, color="black", lw=1.50)
line.set_dashes([5, 10, 15, 10]) # format: line length, space length, ...

# possible marker symbols: marker = '+', 'o', '*', 's', ',', '.', '1', '2', '3', '4', ...
ax.plot(x, x+ 9, color="blue", lw=3, ls='-', marker='+')
ax.plot(x, x+10, color="blue", lw=3, ls='--', marker='o')
ax.plot(x, x+11, color="blue", lw=3, ls='-', marker='s')
ax.plot(x, x+12, color="blue", lw=3, ls='--', marker='1')

# marker size and color
ax.plot(x, x+13, color="purple", lw=1, ls='-', marker='o', markersize=2)
ax.plot(x, x+14, color="purple", lw=1, ls='-', marker='o', markersize=4)
ax.plot(x, x+15, color="purple", lw=1, ls='-', marker='o', markersize=8, markerfacecolor="red")
ax.plot(x, x+16, color="purple", lw=1, ls='-', marker='s', markersize=8,
        markerfacecolor="yellow", markeredgewidth=3, markeredgecolor="green");
```



In [1]:
```
import seaborn as sns
import pandas as pd
import numpy as np
```

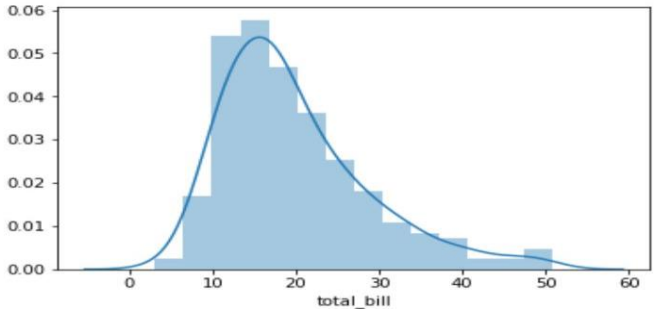
In [2]:
```
tips = sns.load_dataset('tips')
tips.head()
```

Out[2]:

	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

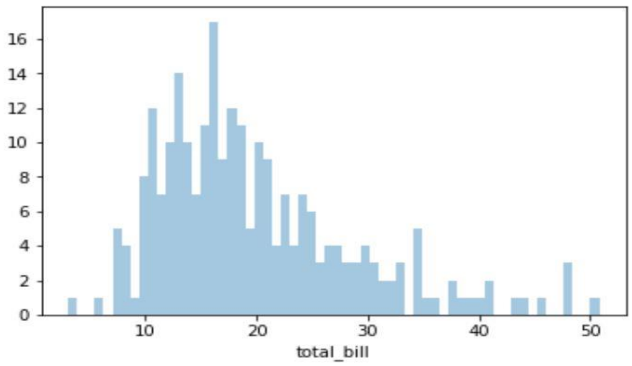
In [4]:
```
sns.distplot(tips['total_bill'])
```

Out[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e7d4b87438>
```



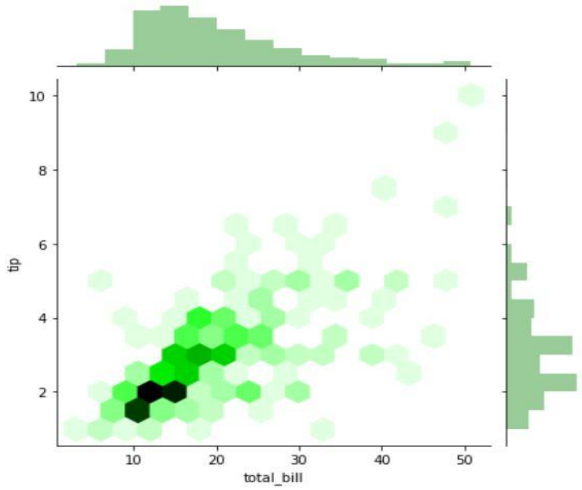
In [5]:
```
sns.distplot(tips['total_bill'],bins=60,kde=False)
```

Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e7d4eaeac8>
```



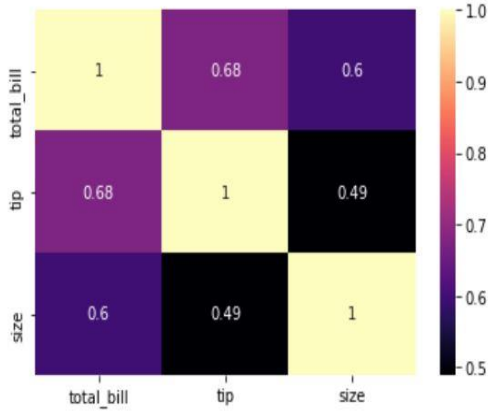
In [6]:
```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='hex',color='green')
```

Out[6]:
```
<seaborn.axisgrid.JointGrid at 0x1e7d5f39828>
```



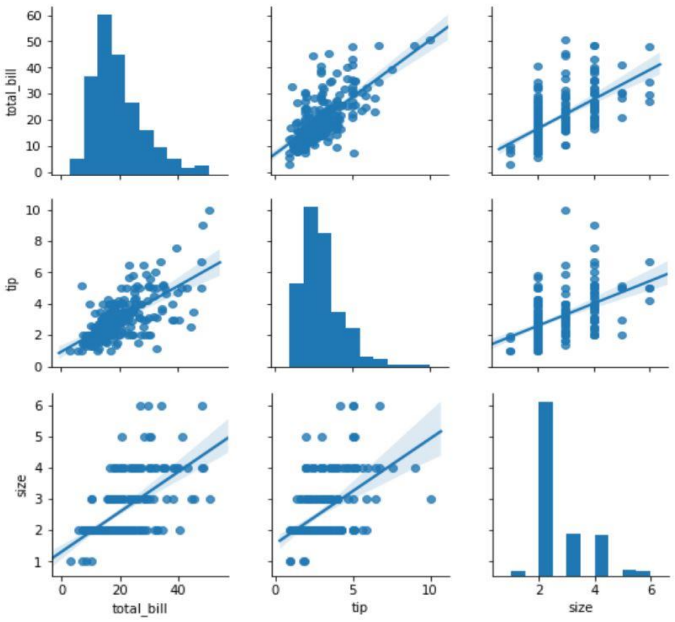
In [11]:
```
sns.heatmap(tips.corr(),annot=True,cmap="magma")
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e7d6f83208>
```



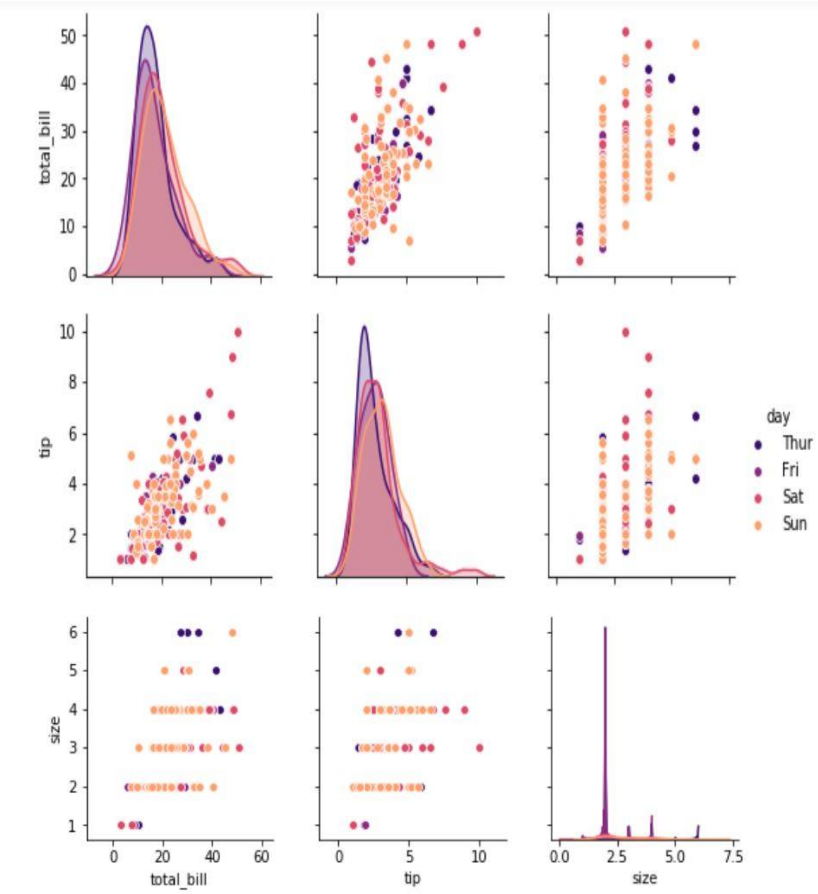
In [7]:
```
sns.pairplot(tips,kind='reg')
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x1e7d608b2e8>



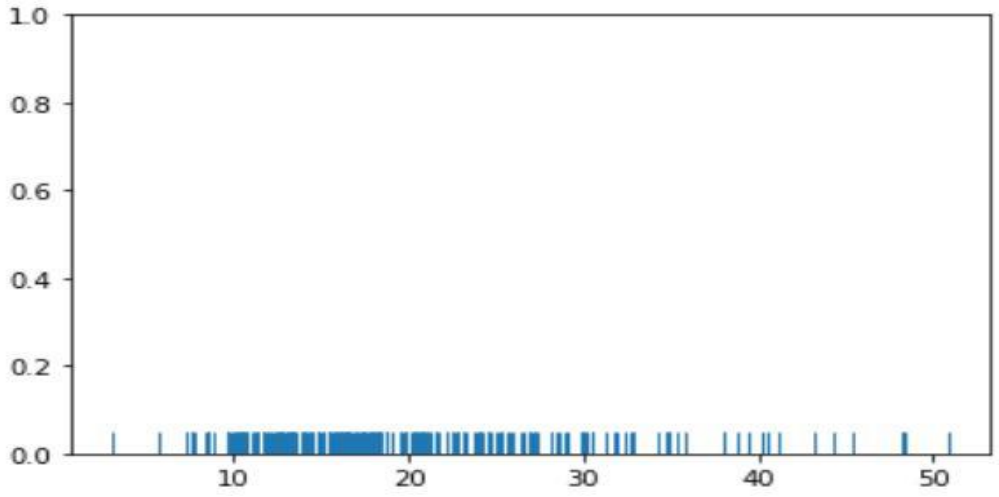
In [8]:
```
sns.pairplot(tips,hue='day',palette='magma')
```

Out[8]: <seaborn.axisgrid.PairGrid at 0x1e7d659c278>



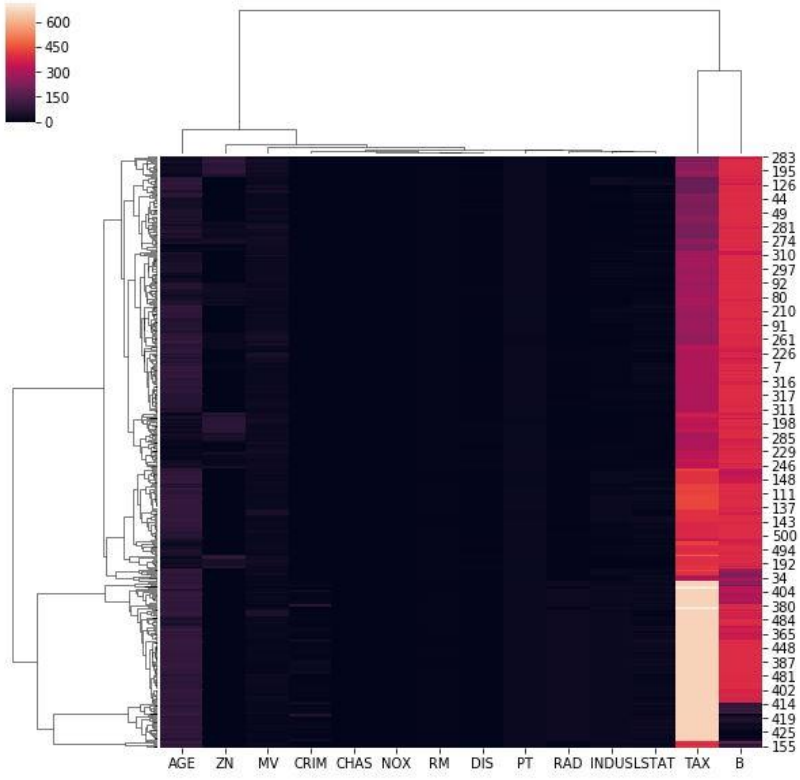
In [9]:
```
sns.rugplot(tips['total_bill'])
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1e7d6d53f60>



In [57]:
```
bos=pd.read_csv('boston - boston.csv.csv')
bos.head()
sns.clustermap(bos)
```

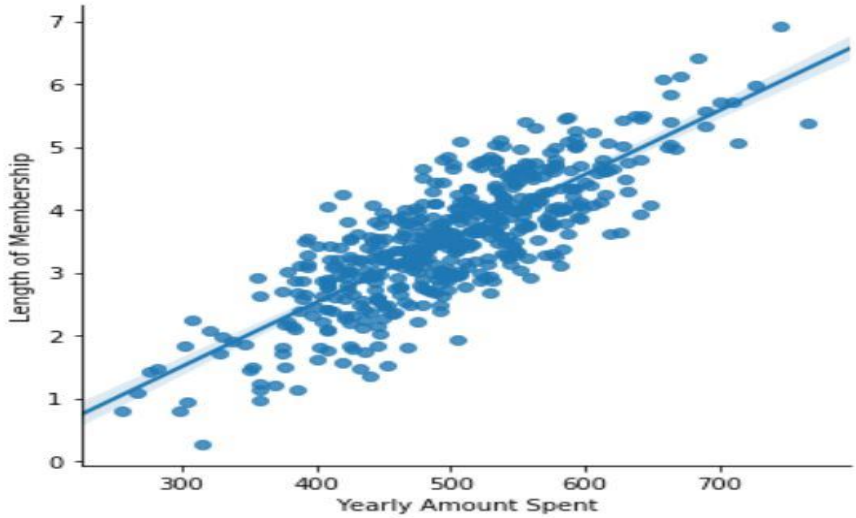
Out[57]: <seaborn.matrix.ClusterGrid at 0x268bd712748>



In [4]:
```
ecom=pd.read_csv('Ecommerce Customers.csv')
ecom.head(3)
sns.lmplot(x='Yearly Amount Spent',y='Length of Membership',data=ecom)
ecom.head()
```

Out[4]:

	Email	Address	Avatar	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
0	mstephenson@fernandez.com	835 Frank Tunnel\nWrightmouth, MI 82180-9605	Violet	34.497268	12.655651	39.577668	4.082621	587.951054
1	hduke@hotmail.com	4547 Archer Common\nDiazchester, CA 06566-8576	DarkGreen	31.926272	11.109461	37.268959	2.664034	392.204933
2	pallen@yahoo.com	24645 Valerie Unions Suite 582\nCobbborough, D...	Bisque	33.000915	11.330278	37.110597	4.104543	487.547505
3	riverarebecca@gmail.com	1414 David Throughway\nPort Jason, OH 22070-1220	SaddleBrown	34.305557	13.717514	36.721283	3.120179	581.852344
4	mstephens@davidson-herman.com	14023 Rodriguez Passage\nPort Jacobville, PR 3...	MediumAquaMarine	33.330673	12.795189	37.536653	4.446308	599.406092



In [5]:
```
X=ecom[['Avg. Session Length','Time on App','Time on Website','Length of Membership']]
y=ecom['Yearly Amount Spent']
```

In [6]:
```
from sklearn.model_selection import train_test_split
```

In [7]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=101)
```

In [8]:
```
from sklearn.linear_model import LinearRegression
```

In [9]:
```
lm=LinearRegression()
```

In [10]:
```
lm
```

Out[10]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [11]:
```
lm.fit(X,y)
```

Out[11]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

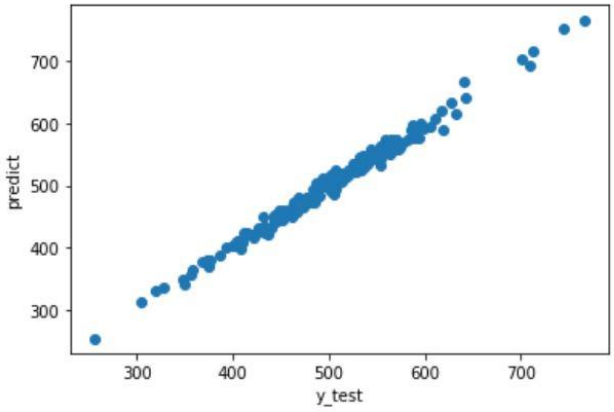
In [12]:
```
lm.coef_
```

Out[12]:
```
array([25.73427108, 38.70915381,  0.43673884, 61.57732375])
```

In [13]:
```
predict=lm.predict(X_test)
```

In [14]:
```
plt.scatter(x=y_test,y=predict)
plt.xlabel('y_test')
plt.ylabel("predict")
```

Out[14]:
```
Text(0, 0.5, 'predict')
```



In [3]:
```
from sklearn.model_selection import train_test_split
```

In [4]:
```
data=pd.read_csv("advertising.csv")
X=data[['Daily Time Spent on Site','Age','Area Income','Daily Internet Usage','Male']]
y=data['Clicked on Ad']
```

In [5]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.33,random_state=42)
```

In [6]:
```
from sklearn.linear_model import LogisticRegression
```

In [7]:
```
log=LogisticRegression()
log.fit(X_train,y_train)
```

```
C:\Users\User\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed t
o 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[7]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [8]:
```
pred=log.predict(X_test)#it should be only X_test but not y because of the dependency
pred
```

Out[8]:
```
array([1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1,
       1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0,
       1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1,
       0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0,
       1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1,
       0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0],
      dtype=int64)
```

In [9]:
```
from sklearn.metrics import confusion_matrix
```

In [10]:
```
print(confusion_matrix(y_test,pred))
```

```
[[156   6]
 [ 24 144]]
```

In [11]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.96      0.91       162
           1       0.96      0.86      0.91       168

   micro avg       0.91      0.91      0.91       330
   macro avg       0.91      0.91      0.91       330
weighted avg       0.91      0.91      0.91       330
```

In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import make_blobs
```

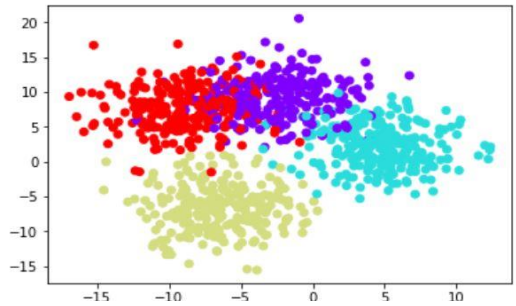
In [3]:
```
data=make_blobs(n_samples=1000,n_features=2,centers=4,cluster_std=3.0,random_state=42)
```

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

Out[6]:
```
(array([[ -7.98846414,   6.54680799],
        [ -4.65334108,  -5.98223185],
        [ -5.20464645,  -6.65209342],
        ...,
        [  1.79168217,   9.87031588],
        [ -8.7081638 ,  -7.4410235 ],
        [-10.52483184,   6.69585729]]),
 array([3, 2, 2, 1, 1, 2, 1, 2, 2, 1, 1, 3, 0, 2, 2, 2, 0, 0, 0, 1, 1, 3,
        3, 3, 1, 1, 0, 0, 2, 1, 2, 2, 2, 0, 0, 3, 2, 1, 3, 3, 1, 2, 1, 3,
        1, 3, 0, 1, 3, 1, 2, 0, 1, 3, 0, 3, 0, 0, 0, 2, 2, 0, 2, 3, 1, 0,
        2, 2, 1, 0, 3, 0, 1, 2, 1, 3, 1, 0, 1, 0, 2, 0, 0, 0, 1, 3, 2, 2,
        0, 0, 0, 0, 1, 1, 3, 1, 3, 0, 1, 2, 1, 3, 3, 0, 3, 1, 1, 0, 2, 0,
        3, 2, 1, 1, 1, 1, 2, 3, 2, 1, 0, 2, 3, 1, 3, 2, 1, 3, 2, 1, 0, 2,
        1, 3, 1, 3, 0, 2, 1, 1, 0, 0, 3, 3, 3, 1, 1, 0, 0, 0, 0, 3, 2, 2,
        0, 1, 0, 1, 1, 3, 2, 0, 1, 2, 0, 0, 1, 2, 3, 2, 1, 0, 0, 1, 0, 3,
        2, 3, 2, 3, 1, 1, 0, 2, 0, 2, 1, 3, 0, 2, 1, 0, 1, 1, 0, 3, 2, 2,
        2, 3, 0, 2, 1, 0, 1, 1, 2, 0, 1, 2, 2, 3, 2, 2, 1, 0, 2, 0, 3, 1,
        3, 3, 2, 0, 3, 0, 1, 2, 2, 0, 0, 2, 0, 3, 2, 2, 3, 2, 2, 1, 2, 3,
        2, 1, 3, 0, 1, 0, 1, 1, 1, 1, 3, 1, 1, 2, 0, 2, 2, 1, 1, 1, 3, 1,
        3, 3, 2, 1, 0, 3, 1, 0, 1, 2, 0, 3, 1, 3, 2, 1, 3, 2, 3, 1, 2, 0,
        0, 2, 0, 3, 3, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 3, 2, 3, 0, 2, 3, 1,
        3, 3, 2, 3, 3, 1, 2, 3, 1, 0, 3, 3, 2, 2, 2, 1, 3, 2, 0, 2, 0, 0,
        2, 2, 3, 3, 2, 0, 3, 1, 3, 1, 3, 2, 2, 1, 0, 1, 3, 3, 0, 0, 0, 0,
        0, 1, 0, 0, 2, 1, 0, 3, 1, 2, 2, 1, 3, 1, 3, 3, 3, 2, 1, 1, 1, 1,
        0, 1, 0, 2, 1, 0, 2, 1, 3, 1, 1, 3, 0, 3, 2, 3, 2, 0, 3, 0, 3, 3,
        2, 1, 1, 1, 0, 3, 0, 1, 1, 3, 1, 0, 0, 3, 1, 0, 3, 2, 3, 2, 2, 0,
        3, 1, 0, 1, 0, 0, 0, 3, 2, 3, 1, 2, 0, 0, 3, 0, 3, 2, 3, 2, 2, 0,
        3, 3, 3, 1, 0, 0, 1, 1, 2, 3, 3, 3, 1, 0, 3, 0, 2, 2, 1, 0, 1, 1,
        0, 0, 1, 1, 2, 2, 1, 0, 3, 1, 1, 3, 3, 2, 3, 0, 1, 0, 3, 1, 3, 2,
        1, 0, 1, 2, 0, 1, 0, 2, 0, 3, 0, 3, 2, 2, 3, 3, 3, 0, 3, 0, 1, 2,
        1, 1, 0, 1, 1, 2, 2, 0, 2, 1, 3, 0, 0, 0, 0, 1, 1, 1, 3, 1, 1, 2,
        1, 0, 0, 1, 0, 3, 1, 0, 0, 1, 1, 3, 2, 3, 3, 0, 3, 1, 0, 1, 3, 3,
        3, 0, 0, 0, 1, 1, 3, 2, 0, 0, 1, 0, 0, 0, 2, 3, 1, 1, 1, 0, 1, 2,
        2, 2, 1, 3, 3, 3, 1, 0, 3, 3, 3, 0, 3, 1, 3, 2, 2, 3, 3, 3, 3, 0,
        2, 2, 0, 2, 0, 1, 1, 1, 1, 0, 2, 3, 2, 0, 3, 0, 1, 3, 0, 2, 3, 2,
        3, 1, 0, 3, 2, 1, 0, 1, 3, 2, 2, 3, 2, 1, 2, 2, 3, 0, 1, 2, 0, 2,
        3, 1, 3, 2, 0, 2, 3, 0, 1, 2, 3, 3, 2, 2, 3, 3, 1, 3, 1, 0, 2, 0,
        1, 3, 0, 3, 1, 2, 0, 0, 3, 0, 2, 2, 3, 0, 3, 3, 3, 1, 3, 0, 1, 2,
        1, 1, 1, 1, 2, 1, 0, 0, 3, 1, 0, 1, 0, 2, 2, 3, 1, 3, 0, 2, 1, 2,
        2, 3, 1, 3, 2, 2, 3, 3, 0, 0, 3, 3, 0, 2, 3, 1, 0, 2, 2, 3, 0, 0,
        3, 2, 3, 0, 1, 3, 1, 1, 2, 0, 2, 3, 1, 2, 2, 3, 3, 2, 2, 0, 0, 2,
        2, 2, 1, 0, 3, 3, 2, 2, 3, 2, 2, 2, 3, 3, 2, 0, 1, 2, 0, 0, 0, 3,
        1, 2, 1, 0, 2, 3, 0, 0, 2, 3, 1, 2, 0, 1, 3, 0, 2, 3, 2, 2, 3, 0,
        3, 2, 0, 0, 0, 3, 2, 3, 3, 1, 3, 1, 0, 1, 1, 3, 0, 1, 0, 0, 1, 3,
        1, 3, 2, 3, 2, 0, 2, 3, 3, 1, 0, 1, 2, 2, 3, 0, 1, 2, 0, 3, 1, 0,
        0, 3, 2, 1, 1, 1, 2, 2, 1, 2, 3, 3, 1, 2, 2, 3, 2, 2, 3, 1, 3, 3,
        2, 2, 0, 3, 0, 2, 3, 0, 3, 2, 3, 0, 3, 2, 1, 1, 2, 2, 2, 0, 0, 0,
        0, 3, 3, 2, 1, 1, 2, 0, 2, 2, 2, 2, 2, 2, 0, 3, 3, 1, 1, 3, 3, 0,
        1, 0, 1, 3, 0, 3, 0, 3, 0, 1, 2, 2, 2, 3, 0, 2, 1, 3, 3, 3, 2, 0,
        1, 3, 1, 0, 0, 1, 2, 0, 0, 3, 1, 3, 3, 1, 0, 0, 3, 0, 2, 2, 2, 1,
        1, 0, 0, 1, 2, 0, 0, 1, 0, 2, 1, 1, 3, 0, 1, 2, 3, 1, 2, 0, 3, 1,
        3, 2, 0, 2, 2, 0, 2, 3, 2, 3, 0, 3, 1, 1, 1, 2, 3, 0, 0, 0, 1, 1,
        1, 0, 1, 2, 1, 2, 0, 1, 2, 3]))
```

In [18]:
```
plt.scatter(data[0][:,0],data[0][:,1],c=data[1],cmap='rainbow')
```

Out[18]:
```
<matplotlib.collections.PathCollection at 0x1a8047165c0>
```



In [30]:
```
from sklearn.cluster import KMeans
```

In [64]:
```
km=KMeans(n_clusters=7)
```

In [65]:
```
km.fit(data[0])
```

Out[65]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=7, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [66]:
```
km.labels_
```

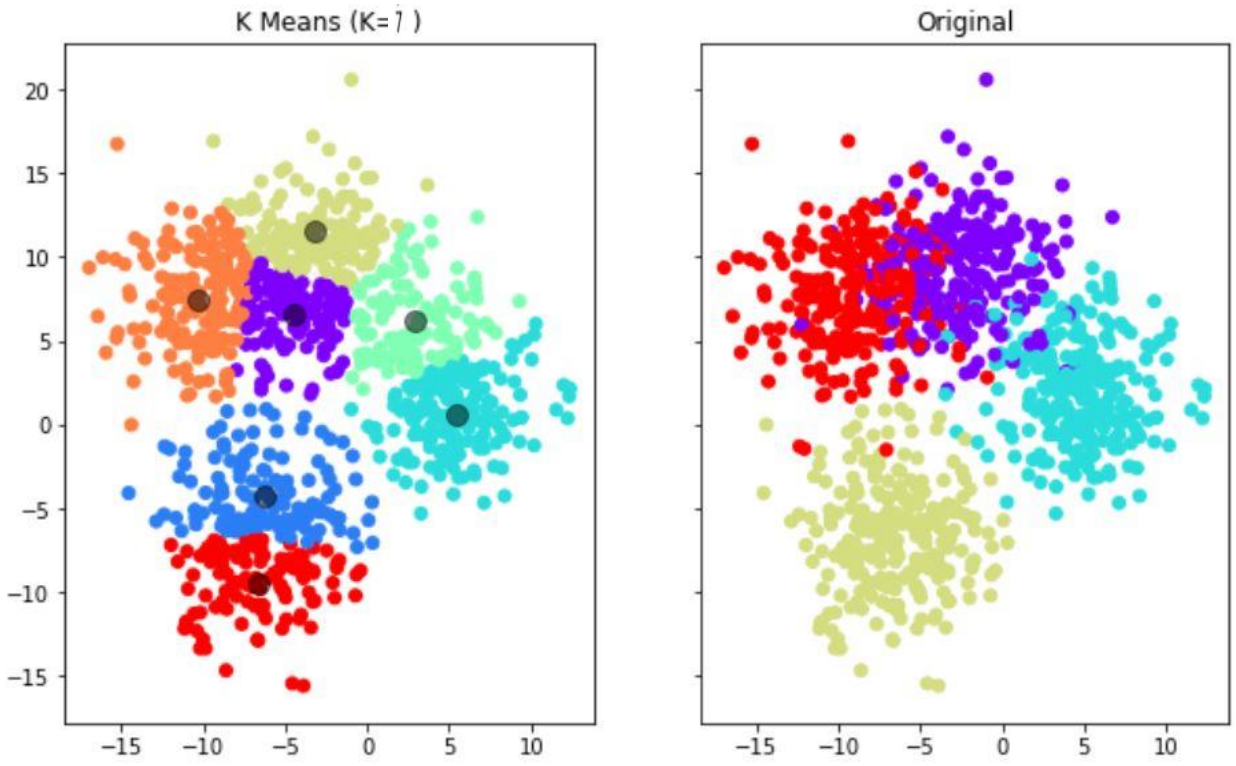
Out[66]:
```
array([5, 1, 1, 2, 2, 1, 2, 1, 1, 2, 2, 0, 0, 6, 6, 1, 0, 4, 3, 2, 2, 5,
       5, 5, 2, 3, 4, 4, 1, 3, 1, 1, 1, 4, 4, 0, 1, 3, 4, 5, 2, 6, 2, 5,
       2, 3, 0, 3, 5, 2, 1, 4, 2, 5, 3, 5, 4, 3, 4, 1, 1, 4, 6, 0, 2, 3,
       6, 1, 2, 4, 0, 4, 2, 6, 2, 5, 3, 0, 3, 4, 6, 0, 0, 4, 2, 5, 6, 1,
       0, 4, 4, 3, 2, 2, 0, 2, 5, 4, 2, 6, 2, 5, 5, 5, 4, 3, 2, 3, 1, 4,
       4, 6, 3, 2, 2, 2, 1, 5, 1, 2, 0, 6, 5, 3, 5, 6, 3, 5, 1, 3, 4, 1,
       2, 4, 2, 4, 4, 1, 2, 3, 4, 0, 0, 5, 5, 2, 3, 0, 0, 0, 0, 5, 6, 1,
       4, 2, 4, 2, 3, 5, 1, 0, 3, 1, 3, 4, 2, 1, 0, 6, 3, 4, 4, 3, 0, 4,
       1, 5, 6, 5, 3, 2, 0, 6, 4, 1, 2, 5, 4, 6, 2, 3, 3, 3, 4, 0, 1, 6,
       6, 5, 0, 1, 3, 0, 3, 3, 6, 0, 2, 1, 1, 0, 6, 6, 3, 4, 1, 0, 5, 2,
       5, 5, 6, 0, 5, 0, 2, 6, 1, 0, 4, 6, 0, 5, 6, 1, 5, 6, 1, 2, 6, 5,
       6, 2, 5, 3, 3, 4, 2, 2, 3, 2, 5, 2, 2, 6, 4, 6, 1, 2, 2, 3, 5, 3,
       5, 4, 6, 2, 3, 5, 2, 2, 3, 1, 0, 0, 3, 4, 6, 3, 5, 1, 4, 2, 6, 4,
       4, 1, 0, 5, 5, 5, 1, 1, 1, 4, 1, 6, 6, 0, 6, 5, 6, 5, 0, 6, 5, 2,
       5, 5, 6, 0, 5, 2, 6, 5, 2, 4, 5, 5, 6, 1, 1, 3, 4, 1, 4, 1, 0, 2,
       1, 1, 5, 4, 1, 4, 0, 2, 5, 2, 5, 1, 6, 1, 0, 2, 0, 0, 4, 3, 4, 0,
       4, 3, 4, 4, 1, 2, 0, 5, 3, 1, 1, 3, 5, 3, 4, 5, 0, 1, 3, 1, 3, 2,
       4, 2, 4, 6, 2, 0, 6, 2, 0, 2, 3, 4, 3, 4, 6, 5, 1, 3, 5, 5, 0, 5,
       1, 3, 3, 2, 4, 5, 4, 2, 2, 5, 3, 3, 0, 5, 3, 3, 5, 1, 5, 1, 1, 4,
       5, 2, 4, 2, 3, 5, 4, 5, 5, 5, 2, 1, 4, 4, 0, 0, 0, 6, 5, 1, 1, 0,
       0, 5, 5, 2, 0, 4, 3, 3, 1, 5, 5, 0, 2, 0, 5, 0, 1, 6, 3, 0, 2, 3,
       3, 3, 3, 2, 6, 6, 2, 4, 5, 3, 2, 5, 5, 6, 0, 5, 2, 4, 4, 2, 0, 6,
       2, 4, 3, 1, 3, 2, 4, 6, 4, 4, 4, 0, 1, 6, 5, 5, 0, 4, 5, 0, 3, 1,
       2, 3, 5, 2, 2, 1, 1, 4, 6, 2, 0, 0, 0, 5, 0, 3, 2, 2, 0, 2, 3, 1,
       2, 4, 4, 1, 0, 5, 2, 4, 4, 2, 2, 4, 6, 5, 5, 3, 5, 2, 4, 2, 5, 0,
       5, 4, 4, 4, 2, 2, 5, 1, 3, 0, 2, 0, 4, 5, 1, 5, 3, 2, 2, 5, 2, 6,
       1, 6, 2, 0, 5, 5, 2, 3, 5, 5, 0, 4, 5, 3, 1, 6, 6, 0, 5, 5, 5, 4,
       1, 6, 3, 6, 3, 3, 2, 0, 3, 3, 6, 0, 6, 0, 0, 3, 2, 5, 0, 6, 0, 1,
       5, 2, 0, 0, 6, 2, 3, 0, 5, 1, 6, 5, 1, 2, 1, 1, 0, 4, 2, 6, 4, 6,
       5, 2, 5, 1, 4, 6, 5, 4, 3, 1, 5, 5, 6, 6, 5, 5, 2, 5, 2, 3, 6, 3,
       2, 5, 4, 5, 2, 1, 5, 3, 5, 4, 6, 1, 5, 3, 5, 5, 5, 2, 5, 4, 2, 1,
       3, 2, 3, 2, 1, 2, 4, 0, 5, 2, 4, 2, 4, 6, 6, 5, 3, 5, 0, 1, 2, 1,
       6, 5, 2, 5, 1, 6, 5, 0, 3, 3, 0, 4, 0, 6, 5, 2, 3, 6, 1, 5, 4, 0,
       5, 6, 5, 0, 3, 5, 2, 2, 1, 3, 1, 0, 2, 1, 6, 5, 5, 6, 1, 4, 4, 6,
       1, 1, 2, 3, 5, 5, 1, 6, 5, 1, 6, 6, 5, 5, 1, 4, 2, 1, 3, 4, 0, 5,
       2, 1, 2, 0, 1, 5, 0, 0, 1, 4, 3, 6, 4, 2, 5, 4, 1, 5, 1, 6, 5, 0,
       0, 6, 4, 0, 0, 5, 6, 0, 5, 2, 5, 2, 4, 2, 2, 0, 4, 2, 0, 0, 2, 0,
       2, 5, 6, 5, 6, 4, 6, 0, 4, 2, 0, 2, 6, 6, 5, 0, 2, 6, 4, 4, 2, 4,
       0, 5, 1, 2, 2, 3, 1, 1, 2, 6, 5, 4, 2, 1, 1, 4, 1, 1, 1, 2, 5, 5,
       1, 6, 4, 5, 3, 6, 5, 0, 0, 6, 0, 4, 5, 1, 2, 3, 1, 1, 1, 0, 0, 0,
       4, 5, 0, 1, 3, 2, 1, 0, 1, 6, 6, 1, 6, 6, 4, 0, 5, 2, 3, 0, 5, 3,
       3, 4, 2, 5, 4, 5, 4, 5, 0, 2, 6, 6, 1, 5, 0, 1, 3, 5, 5, 1, 6, 0,
       3, 0, 2, 3, 0, 2, 1, 4, 0, 5, 3, 0, 4, 2, 4, 0, 5, 0, 6, 1, 6, 2,
       2, 4, 4, 2, 1, 0, 4, 3, 4, 1, 1, 3, 5, 5, 2, 1, 5, 3, 6, 0, 0, 0,
       5, 1, 4, 6, 6, 4, 1, 5, 1, 5, 3, 4, 3, 2, 2, 6, 5, 0, 4, 4, 3, 2,
       3, 0, 2, 6, 2, 1, 4, 3, 6, 5])
```

In [67]:
```
centers=km.cluster_centers_
centers
```

Out[67]:
```
array([[ -4.5251967 ,   6.54920387],
       [ -6.19892348,  -4.21539494],
       [  5.47584074,   0.62951557],
       [  2.92963795,   6.23088366],
       [ -3.23167205,  11.4964361 ],
       [-10.28685374,   7.50289726],
       [ -6.66665895,  -9.41473838]])
```

In [68]:
```
f,(ax1,ax2)=plt.subplots(nrows=1,ncols=2,sharey=True,figsize=(10,6))
ax1.set_title('K Means (K=7)')
ax1.scatter(data[0][:,0],data[0][:,1],c=km.labels_,cmap='rainbow')

ax2.set_title("Original")
ax2.scatter(data[0][:,0],data[0][:,1],c=data[1],cmap='rainbow')

ax1.scatter(x=centers[:,0],y=centers[:,1],c='black',s=100,alpha=0.5)
```

Out[68]: <matplotlib.collections.PathCollection at 0x1a8064e4978>



In [69]:
```
sum_square={}
```

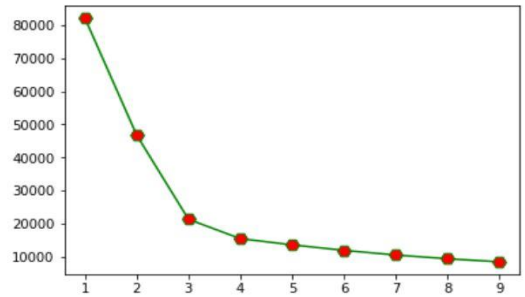
In [73]:
```
for k in range(1,10):
    km=KMeans(n_clusters=k).fit(data[0])
    sum_square[k]=km.inertia_ #.inertia:Computing Sum of Square Distances
```

In [74]:
```
sum_square
```

Out[74]:
```
{1: 82169.48782364259,
 2: 46732.65570616981,
 3: 21233.519961941543,
 4: 15386.308239157897,
 5: 13492.592959109155,
 6: 11841.84251938933,
 7: 10421.411765241739,
 8: 9295.712549522024,
 9: 8356.284798558772}
```

In [84]:
```
plt.plot(list(sum_square.keys()),list(sum_square.values()),linestyle='-',marker='H',color='g',markersize=10,markerfacecolor='r')
```

Out[84]: [<matplotlib.lines.Line2D at 0x1a80797c7b8>]



Conclusion

Some AI experts predict that AI will be able to do anything that humans can but do it better. This is a questionable assumption, but AI will surely surpass humans in specific domains. A chess computer beating the world chess champion was the first example.

If AI were to develop to the point that it can do everything better than humans, it would mean that it would also do better in science and technology. It may decide that it is no longer worthwhile to develop a certain field of research - or it may decide space travel is a waste of time as long as humans on earth are living in poverty and more than a billion people have no access to clean drinking water. Most scenarios about future AI are hypothetical, but AI presents us with existential questions. There is a never ending chase of the Machine and humans, it will run in unparalleled way to achieve in the form of BEST. It is difficult to determine where this technology might create new jobs in the future, yet easier to see which tasks AI might take from humans. It's likely that in future we will adapt technological changes by inventing entirely new types of work, and by taking advantage of our uniquely human capabilities.
That what I can see at present with trends in AI.

Bibliography

1) Information & Images -> Google , Wikipedia and Quora.
2) Codes & its Snapshots -> All are done by me in Anaconda's Jupyter Notebook.