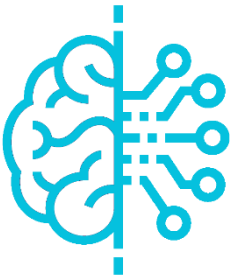
औद्योगिक प्रशिक्षण के लिए राष्ट्रीय संस्थान

National Institute for Industrial Training

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

National Institute for Industrial Training- One Premier Organization with Non Profit Status Registered Under Govt.of West Bengal,Empanelled Under Planning Commission Govt.of India , Empanelled Under Central Social Welfare Board Govt. of India , Registered with National Career Services , Registered with National Employment Services.



Subject: Python with Artificial Intelligence

Submitted By: Erina Karati

Submitted To: Sayantan Chakraborty Sir

CONTENTS

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to all those who provided me the possibility to complete this project. I give special gratitude to my project guides, Mr. Sayantan Chakraborty and Ankit Pramanik, whose contribution in stimulating ideas and encouragement helped me 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 put tremendous effort 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 PROFILE

- NAME: Erina Karati
- ADDRESS: Nandanik Apt, MB 96, Sector V, Salt Lake, Kolkata
- PIN_CODE: 700102
- E-MAIL: erina.karati@gmail.com
- NATIONALITY: Indian
- DATE OF BIRTH: 08/18/1999
- SEX: Female
- LANGUAGES KNOWN: English, Bengali and Hindi
- CARRER OBJECTIVE: A curious problem solver who is in love with Math and Programming. Always trying to learn by getting involved in new projects. Seeking to leverage skills in collaboration, communication, and technical areas such as ML(Machine Learning), AI, Android App Development, and Cloud Computing.

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.

THEORY

#	AI	ML
1	Overarching field	Subset of AI
2	The goal is to simulate human intelligence to solve complex problems	The goal is to learn from data and be able to predict results when new data is presented
3	Leads to intelligence or wisdom	Leads to knowledge
4	Tries to find the optimal solution	Tries to find the only solution whether it is optimal or not

---

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.

-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.

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. Its 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 & SOFTWARE REQUIREMENTS

- ❖ Python (Latest Version Recommended)
- ❖ Anaconda (Free and open-source distribution of the Python)
- ❖ 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

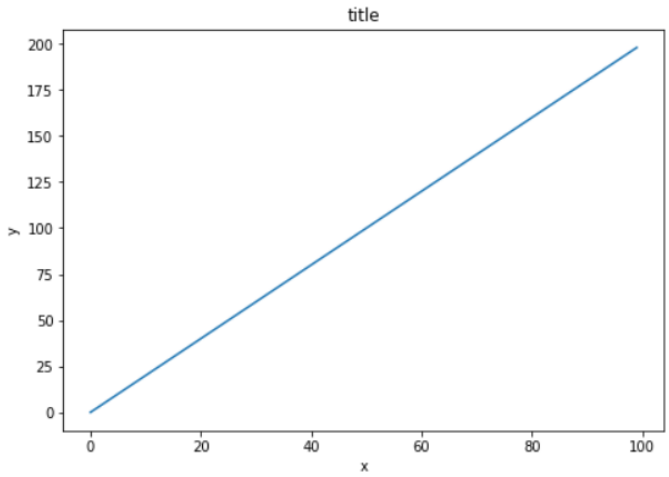
- A great choice of libraries is one of the main reasons Python is the most popular programming language used for AI.
- Working in the ML and AI industry means dealing with a bunch of data that you need to process in the most convenient and effective way. The low entry barrier allows more data scientists to quickly pick up Python and start using it for AI development without wasting too much effort on learning the language.
- Python for machine learning development can run on any platform including Windows, MacOS, Linux, Unix, and twenty-one others.
- Python is very easy to read so every Python developer can understand the code of their peers and change, copy or share it. There's no confusion, errors or conflicting paradigms, and this leads to more a efficient exchange of algorithms, ideas, and tools between AI and ML professionals.
- Python is an open-source language which means that there's a bunch of resources open for programmers starting from beginners and ending with pros.
- For AI developers, it's important to highlight that in artificial intelligence, deep learning, and machine learning, it's vital to be able to represent data in a human-readable format. Python offers a variety of libraries, and some of them are great visualization tools.

SNAPSHOTS

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
x= np.arange(0,100)
y= x*2
z= x**2
```

In [2]:
```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.plot(x,y)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_title('title')
```

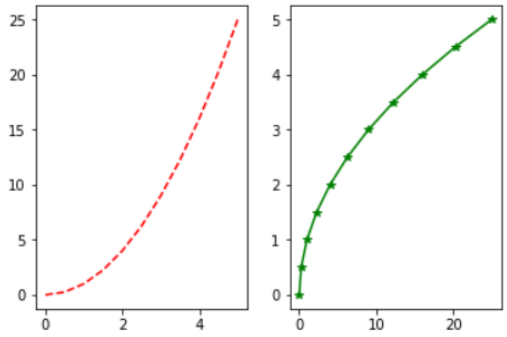
Out[2]: Text(0.5,1,'title')



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

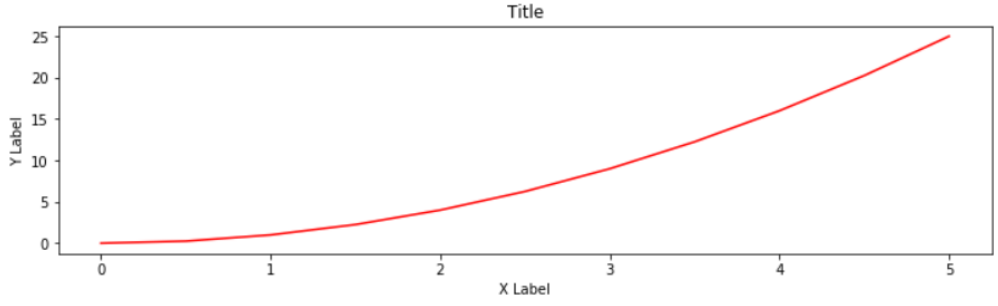
In [11]:
```
plt.subplot(1,2,1)
plt.plot(x,y, 'r--')
plt.subplot(1,2,2)
plt.plot(y,x,'g*-')
```

Out[11]: [<matplotlib.lines.Line2D at 0x1f20ec16ac8>]



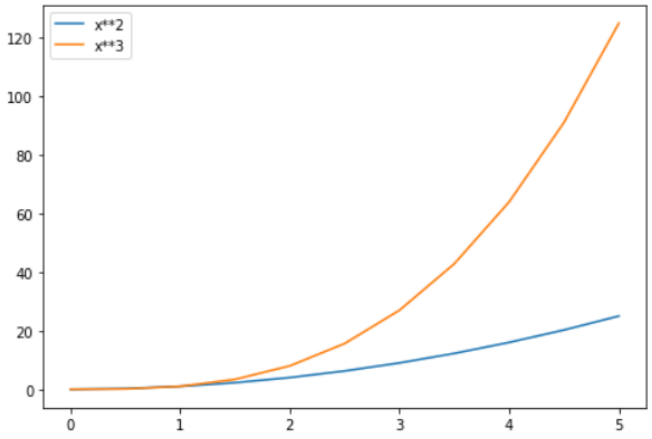
In [32]:
```
fig, axes = plt.subplots(figsize=(12,3))
axes.plot(x,y,'r')
axes.set_xlabel('X Label')
axes.set_ylabel('Y Label')
axes.set_title('Title')
```

Out[32]: Text(0.5,1,'Title')

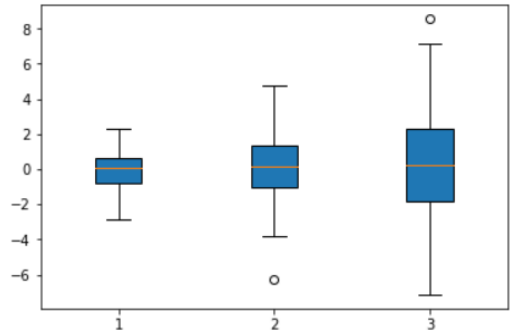


In [5]:
```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.plot(x, x**2, label="x**2")
ax.plot(x, x**3, label="x**3")
ax.legend()
```

Out[5]:
```
<matplotlib.legend.Legend at 0x2661d66ecf8>
```

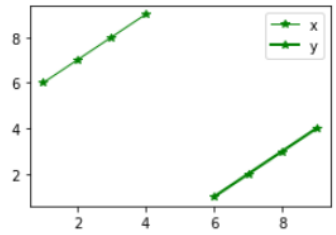


In [59]:
```
data = [np.random.normal(0,std,100) for std in range(1,4)]
plt.boxplot(data, vert=True, patch_artist=True);
```



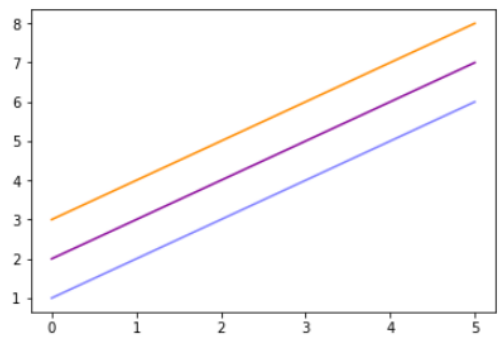
In [11]:
```
fig=plt.figure()
axes=fig.add_axes([0.9,0.9,.5,.5])
axes.plot(x,y,"g*-",label="x",lw=1)
axes.plot(y,x,"g*-",label="y",lw=2)
axes.legend()
```

Out[11]:
```
<matplotlib.legend.Legend at 0x1745a984828>
```

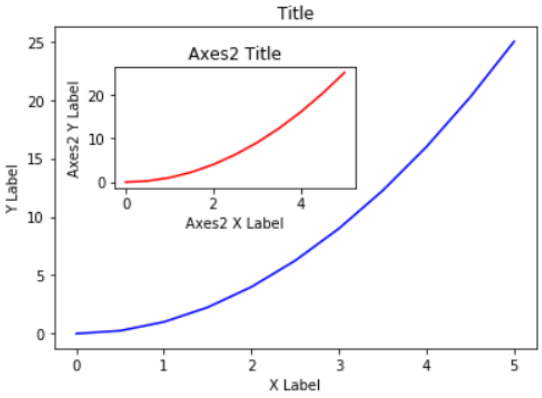


In [49]:
```
fig, ax = plt.subplots()
ax.plot(x, x+1, color="blue",alpha=0.5)
ax.plot(x, x+2,color="#8B099B")
ax.plot(x, x+3,color="#FF8C00")
```

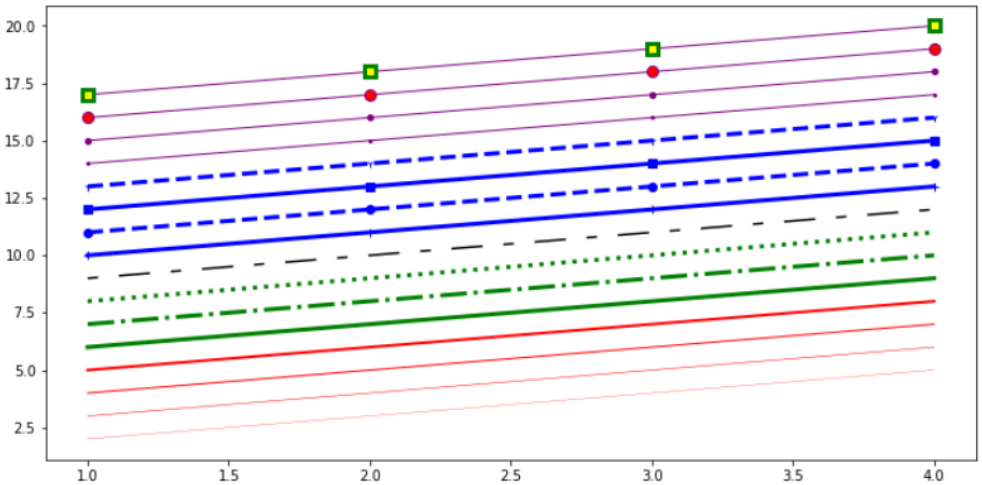
Out[49]:
```
[<matplotlib.lines.Line2D at 0x1f21261dc88>]
```



In [18]:
```
fig = plt.figure()

axes = fig.add_axes([0.1,0.1,0.8,0.8])
axes2 = fig.add_axes([0.2,0.5,0.4,0.3])

axes.plot(x,y,'b')
axes.set_xlabel('X Label')
axes.set_ylabel('Y Label')
axes.set_title('Title')

axes2.plot(x,y,'r')
axes2.set_xlabel('Axes2 X Label')
axes2.set_ylabel('Axes2 Y Label')
axes2.set_title('Axes2 Title');
```



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 [4]:
```
import seaborn as sns
import numpy as np
import pandas as pd
```

In [5]:
```
tips = sns.load_dataset('tips')
```

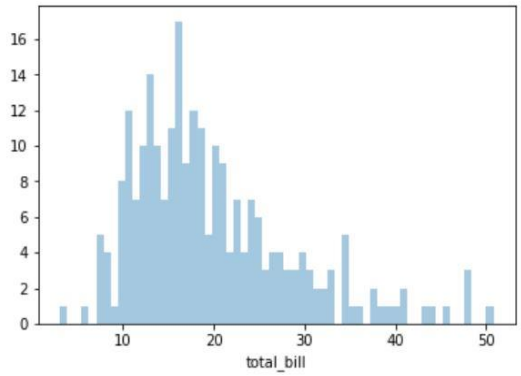
In [7]:
```
tips.head()
```

Out[7]:

	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

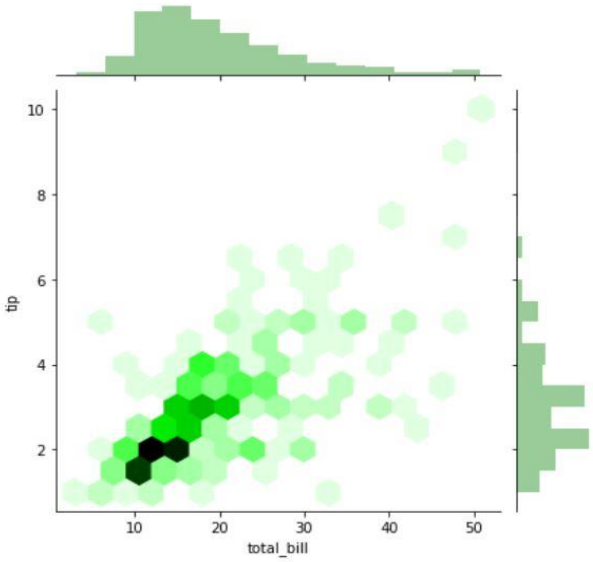
In [5]:
```
sns.distplot(tips['total_bill'],bins=60,kde=False)
```

Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x21b284bbb38>
```



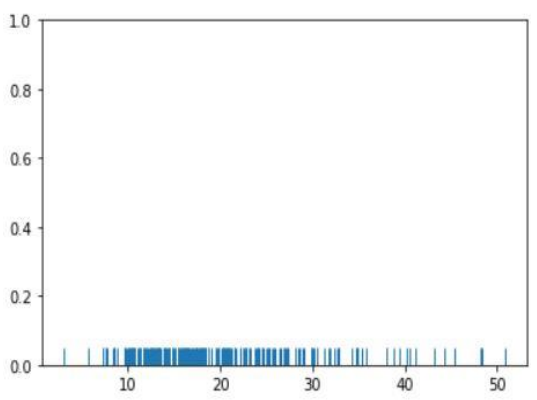
In [6]:
```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='hex',color='green')
```

Out[6]:
```
<seaborn.axisgrid.JointGrid at 0x21b2956e2b0>
```



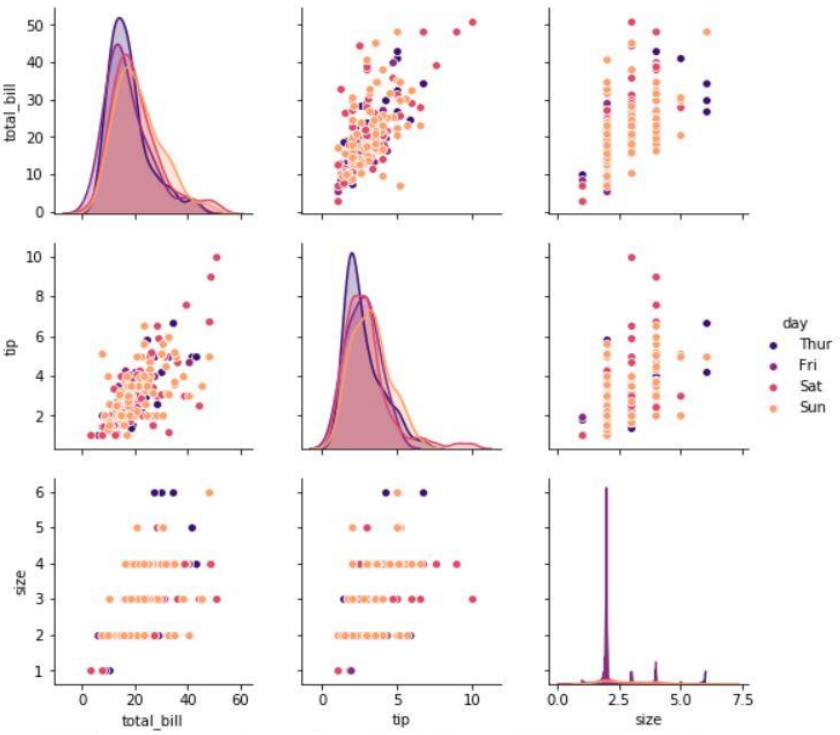
In [57]:
```
sns.rugplot(tips['total_bill'])
```

Out[57]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2526631a828>
```



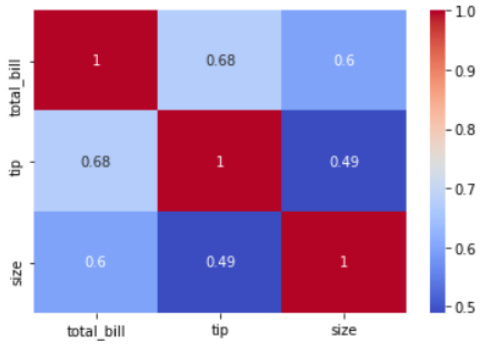
In [12]:
```
sns.pairplot(tips,hue='day',palette='magma')
```

Out[12]: <seaborn.axisgrid.PairGrid at 0x21b2aafe390>



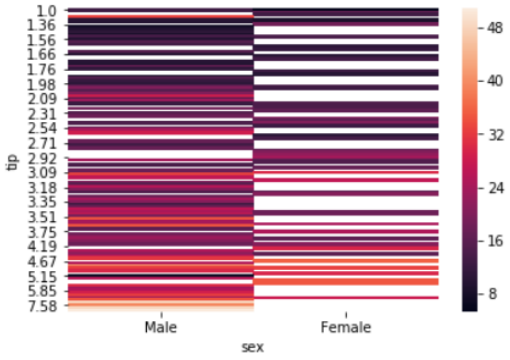
In [30]:
```
sns.heatmap(tips.corr(), cmap='coolwarm',annot=True)
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x23db9bcd748>



In [28]:
```
tipspivot = tips.pivot_table(values='total_bill',index='tip',columns='sex')
sns.heatmap(tipspivot)
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x23db9a95c18>



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

In [2]:
```
us = pd.read_csv('USA_Housing.csv')
```

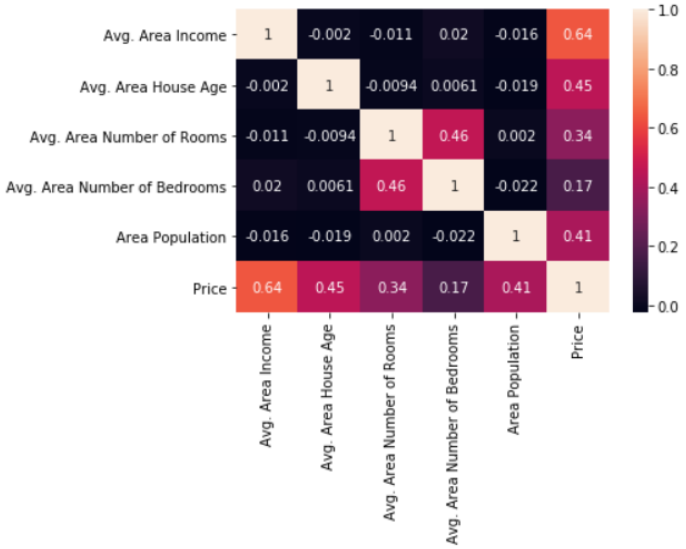
In [4]:
```
USAhousing.head()
```

Out[4]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386

In [6]:
```
sns.heatmap(us.corr(), annot=True)
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x27411f71828>



In [11]:
```
X = us[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
        'Avg. Area Number of Bedrooms', 'Area Population']]
y = us['Price']
```

In [12]:
```
from sklearn.model_selection import train_test_split
```

In [13]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.4, random_state=101)
```

In [14]:
```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
```

In [15]:
```
lm.fit(X_train, y_train)
```

Out[15]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

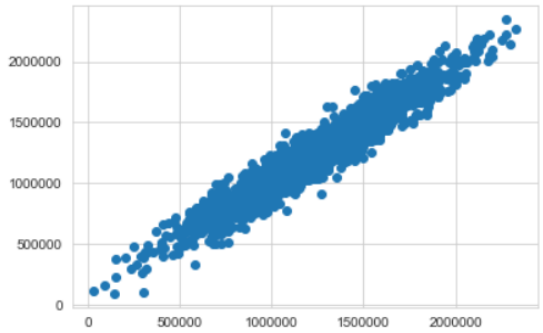
In [16]:
```
lm.intercept_
```

Out[16]:
```
-2640159.796851911
```

In [19]:
```
predictions = lm.predict(X_test)
```

In [20]:
```
plt.scatter(y_test,predictions)
```

Out[20]: <matplotlib.collections.PathCollection at 0x19545be05c0>



In [19]:
```
coeff = pd.DataFrame(lm.coef_,X.columns,columns=['Coefficient'])
coeff
```

Out[19]:

	Coefficient
Avg. Area Income	21.528276
Avg. Area House Age	164883.282027
Avg. Area Number of Rooms	122368.678027
Avg. Area Number of Bedrooms	2233.801864
Area Population	15.150420

In [22]:
```
from sklearn import metrics
```

In [23]:
```
metrics.mean_absolute_error(y_test, predictions)
```

Out[23]: 82288.22251914954

In [24]:
```
metrics.mean_squared_error(y_test, predictions)
```

Out[24]: 10460958907.209503

In [25]:
```
np.sqrt(metrics.mean_squared_error(y_test, predictions))
```

Out[25]: 102278.82922291153

In [6]:
```
ad = pd.read_csv('advertising.csv')
```

In [7]:
```
ad.head()
```

Out[7]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp	Clicked on Ad
0	68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	0	Tunisia	2016-03-27 00:53:11	0
1	80.23	31	68441.85	193.77	Monitored national standardization	West Jodi	1	Nauru	2016-04-04 01:39:02	0
2	69.47	26	59785.94	236.50	Organic bottom-line service-desk	Davidton	0	San Marino	2016-03-13 20:35:42	0
3	74.15	29	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	1	Italy	2016-01-10 02:31:19	0
4	68.37	35	73889.99	225.58	Robust logistical utilization	South Manuel	0	Iceland	2016-06-03 03:36:18	0

In [41]:
```
from sklearn.model_selection import train_test_split
```

In [47]:
```
X = ad[['Daily Time Spent on Site', 'Age', 'Area Income',
       'Daily Internet Usage', 'Male']]
y = ad['Clicked on Ad']
```

In [48]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.33, random_state=42)
```

In [49]:
```
from sklearn.linear_model import LogisticRegression
```

In [50]:
```
logmodel = LogisticRegression()
logmodel.fit(X_train, y_train)
```

Out[50]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [51]:
```
predict = logmodel.predict(X_test)
```

In [52]:
```
from sklearn.metrics import confusion_matrix
```

In [54]:
```
print(confusion_matrix(y_test, predict))
```

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

In [56]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, predict))
```

```
             precision    recall  f1-score   support

          0       0.87      0.96      0.91       162
          1       0.96      0.86      0.91       168

avg / total       0.91      0.91      0.91       330
```

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style='darkgrid')
```

In [2]:
```
from sklearn.datasets import make_blobs
```

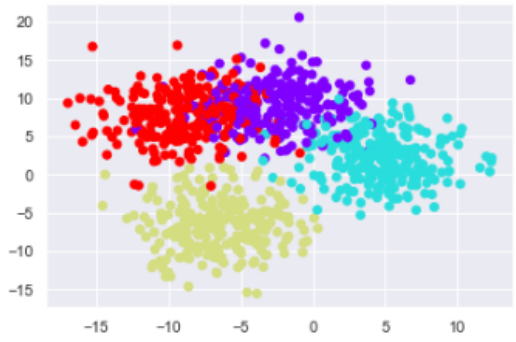
In [3]:
```
data = make_blobs(n_samples=1000, n_features=2, centers=4, cluster_std=3.0, random_state=42)
```

In [4]:
```
data
```

Out[4]:
```
(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 [8]:
```
plt.scatter(data[0][:,0],data[0][:,1], c=data[1], cmap='rainbow')
```

Out[8]:
```
<matplotlib.collections.PathCollection at 0x1b546eec2b0>
```



In [9]:
```
from sklearn.cluster import KMeans
```

In [10]:
```
km = KMeans(n_clusters=6)
```

In [11]:
```
km.fit(data[0])
```

Out[11]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=6, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [12]:
```
km.labels_
```

Out[12]:
```
array([5, 4, 4, 3, 3, 4, 3, 4, 4, 3, 3, 0, 0, 1, 1, 4, 2, 0, 0, 3, 3, 5,
       5, 5, 3, 3, 2, 2, 4, 3, 4, 4, 4, 0, 2, 5, 4, 0, 2, 2, 3, 1, 3, 5,
       3, 0, 2, 3, 5, 3, 4, 0, 3, 5, 0, 5, 2, 0, 2, 4, 4, 2, 1, 5, 3, 0,
       1, 4, 3, 2, 5, 2, 3, 1, 3, 5, 3, 0, 3, 2, 1, 0, 2, 2, 3, 5, 1, 4,
       0, 0, 2, 0, 3, 3, 5, 3, 5, 2, 3, 1, 3, 5, 5, 5, 2, 0, 3, 0, 4, 2,
       2, 1, 0, 3, 3, 3, 4, 5, 4, 3, 0, 1, 5, 3, 5, 1, 0, 5, 4, 0, 2, 4,
       3, 2, 3, 2, 0, 4, 3, 3, 2, 0, 5, 5, 5, 3, 0, 0, 2, 0, 0, 2, 1, 4,
       2, 3, 2, 3, 3, 5, 4, 2, 0, 4, 0, 2, 3, 4, 2, 1, 0, 0, 2, 3, 0, 2,
       4, 5, 1, 5, 0, 3, 2, 1, 2, 4, 3, 5, 2, 1, 3, 0, 0, 0, 2, 2, 4, 1,
       1, 5, 2, 4, 3, 5, 3, 3, 1, 2, 3, 4, 4, 5, 1, 1, 0, 2, 4, 0, 5, 3,
       5, 5, 1, 0, 5, 0, 3, 1, 4, 2, 2, 1, 2, 5, 1, 4, 5, 1, 4, 3, 1, 5,
       1, 3, 5, 0, 0, 2, 3, 3, 0, 3, 5, 3, 3, 1, 2, 1, 4, 3, 3, 0, 5, 3,
       5, 2, 1, 3, 0, 2, 3, 3, 0, 4, 5, 0, 3, 2, 1, 0, 5, 4, 2, 3, 1, 0,
       2, 4, 5, 5, 5, 5, 4, 4, 4, 2, 4, 1, 1, 0, 1, 5, 1, 2, 5, 1, 2, 3,
       5, 2, 1, 2, 5, 3, 1, 5, 3, 2, 5, 5, 1, 4, 4, 3, 2, 4, 0, 4, 0, 3,
       4, 4, 5, 2, 4, 2, 2, 3, 5, 3, 5, 4, 1, 4, 5, 3, 5, 2, 2, 0, 2, 2,
       2, 0, 0, 2, 4, 3, 5, 5, 0, 4, 4, 3, 5, 0, 2, 5, 2, 4, 3, 4, 0, 3,
       2, 3, 2, 1, 3, 2, 1, 3, 5, 3, 0, 2, 0, 2, 1, 5, 4, 0, 5, 2, 2, 5,
       4, 3, 0, 3, 2, 5, 2, 3, 3, 2, 3, 0, 0, 5, 3, 0, 5, 4, 5, 4, 4, 2,
       5, 3, 2, 3, 0, 5, 2, 5, 5, 5, 3, 4, 2, 2, 5, 5, 2, 1, 5, 4, 4, 0,
       5, 2, 5, 3, 0, 2, 3, 0, 4, 5, 5, 5, 3, 5, 5, 0, 4, 1, 0, 0, 3, 0,
       0, 0, 0, 3, 1, 1, 3, 2, 5, 3, 3, 5, 5, 1, 0, 2, 3, 2, 2, 3, 5, 1,
       3, 2, 3, 4, 0, 3, 2, 1, 2, 2, 2, 5, 4, 1, 5, 5, 5, 2, 5, 0, 0, 4,
       3, 0, 2, 3, 3, 4, 4, 0, 1, 3, 2, 0, 0, 5, 2, 0, 3, 3, 2, 3, 3, 4,
       3, 2, 0, 4, 0, 5, 3, 2, 2, 3, 3, 2, 1, 5, 5, 0, 5, 0, 0, 3, 5, 2,
       5, 2, 2, 0, 3, 3, 5, 4, 0, 0, 3, 0, 0, 5, 4, 5, 0, 3, 3, 5, 3, 1,
       4, 1, 3, 5, 5, 5, 3, 0, 5, 5, 2, 2, 5, 3, 4, 1, 1, 2, 5, 5, 5, 2,
       4, 1, 0, 1, 0, 3, 3, 0, 3, 0, 1, 0, 1, 5, 2, 0, 3, 5, 5, 1, 5, 4,
       5, 3, 2, 5, 1, 3, 0, 0, 2, 4, 1, 5, 4, 3, 4, 4, 2, 2, 3, 1, 0, 1,
       2, 3, 5, 4, 2, 1, 5, 2, 0, 4, 5, 5, 1, 1, 5, 5, 3, 5, 3, 0, 1, 0,
       3, 5, 2, 5, 3, 4, 5, 0, 5, 2, 1, 4, 5, 0, 5, 5, 5, 3, 5, 2, 3, 4,
       3, 3, 3, 3, 4, 3, 0, 2, 5, 3, 2, 3, 2, 1, 1, 5, 0, 5, 0, 4, 3, 4,
       1, 5, 3, 5, 4, 1, 5, 2, 0, 0, 5, 2, 0, 1, 5, 3, 0, 1, 4, 5, 2, 0,
       2, 1, 5, 2, 0, 5, 3, 3, 4, 0, 4, 5, 3, 4, 1, 2, 5, 1, 4, 2, 2, 1,
       4, 4, 3, 0, 5, 5, 4, 1, 5, 4, 1, 1, 5, 2, 4, 0, 3, 4, 0, 2, 2, 5,
       3, 4, 3, 5, 4, 5, 0, 2, 4, 2, 3, 1, 2, 3, 5, 2, 4, 5, 4, 1, 5, 2,
       5, 1, 2, 2, 0, 5, 1, 5, 5, 3, 5, 3, 2, 3, 3, 5, 0, 3, 2, 2, 3, 5,
       3, 5, 1, 5, 1, 2, 1, 2, 2, 3, 2, 3, 1, 1, 5, 5, 3, 1, 2, 2, 3, 0,
       0, 2, 4, 3, 3, 3, 4, 4, 3, 1, 5, 2, 3, 4, 4, 2, 4, 4, 4, 3, 5, 5,
       4, 1, 0, 5, 0, 1, 5, 0, 2, 1, 2, 0, 5, 4, 3, 0, 4, 4, 4, 5, 0, 2,
       2, 5, 5, 4, 0, 3, 4, 0, 4, 1, 1, 4, 1, 1, 0, 2, 5, 3, 3, 2, 2, 0,
       0, 2, 3, 5, 2, 5, 0, 5, 2, 3, 1, 1, 4, 5, 0, 4, 3, 5, 5, 4, 1, 2,
       0, 5, 3, 0, 0, 3, 4, 2, 0, 5, 0, 0, 2, 3, 2, 0, 5, 5, 1, 4, 1, 3,
       3, 2, 2, 3, 4, 0, 2, 3, 2, 4, 4, 0, 5, 5, 3, 4, 5, 3, 1, 0, 5, 0,
       5, 4, 2, 1, 1, 2, 4, 5, 4, 5, 0, 2, 0, 3, 3, 1, 5, 0, 0, 2, 0, 3,
       3, 2, 3, 1, 3, 4, 0, 0, 1, 5])
```

In [48]:
```
sum_square={}
for k in range(1,10):
    km = KMeans(n_clusters=k).fit(data[0])
    sum_square[k]=km.inertia_

# .inertia_: Computing sum of squared distances
```

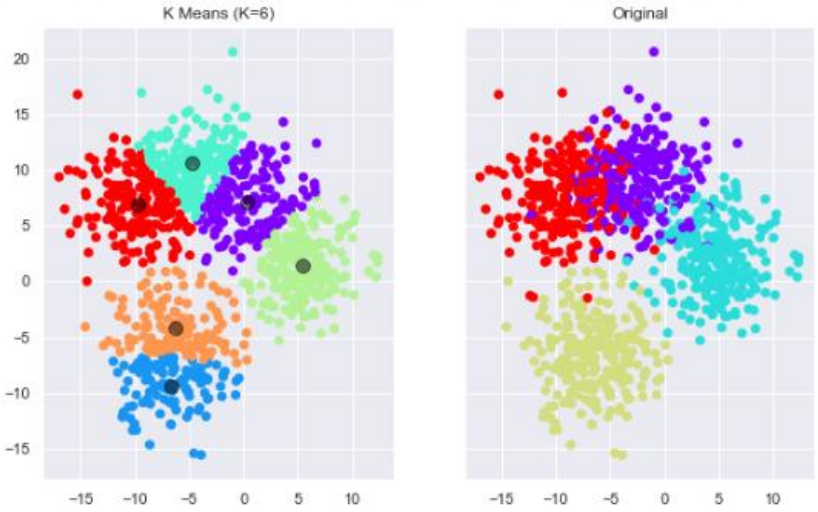
In [49]:
```
sum_square
```

Out[49]:
```
{1: 82169.48782364259,
 2: 46732.65570616981,
 3: 21233.519961941543,
 4: 15386.308239157897,
 5: 13494.927186209077,
 6: 11868.654455819124,
 7: 10423.624403716836,
 8: 9308.384797352886,
 9: 8306.7789982726}
```

In [13]:
```
centers = km.cluster_centers_
centers
```

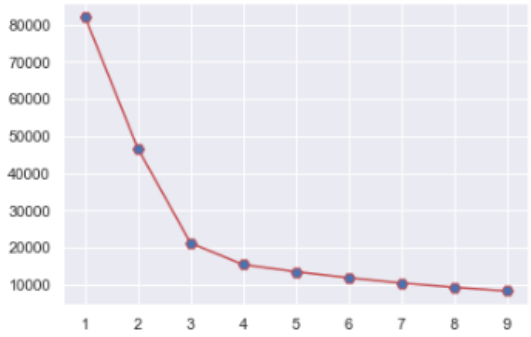
Out[13]:
```
array([[ 0.30873479,  7.12041182],
       [-6.66665895, -9.41473838],
       [-4.67122929, 10.64604694],
       [ 5.39996698,  1.35886454],
       [-6.19892348, -4.21539494],
       [-9.71833619,  6.82327283]])
```

In [14]:
```
f , (ax1, ax2) = plt.subplots(nrows=1, ncols=2, sharey=True, figsize=(10,6))

ax1.set_title('K Means (K=6)')
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);
```



In [53]:
```
plt.plot(list(sum_square.keys()), list(sum_square.values()),
        linestyle='-',
        marker='H',
        color='r',
        markersize=8,
        markerfacecolor='b')
```

Out[53]:
```
[<matplotlib.lines.Line2D at 0x214c3317ba8>]
```



CONCLUSION

It's likely that the upwards trend in capabilities of AI systems will continue; that systems will eventually become capable of solving a wide range of tasks (rather than a new system having to be built for each new problem), and that the adoption of AI within many industries will continue. Evidence suggests AI is currently unable to reproduce human behavior or surpass human thinking; it's likely to stay a complementary workforce tool for a very long time to come. However, steady gradual improvements in AI could reach a point where AI exceeds current expectations. The continued development of AI will depend on moral public opinion regarding the benefits and acceptability of it, on businesses continuing to gain competitive advantage from using it, and continued funding for research and development of it.

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 any routine, repetitive task will be automated. This shift to automation has happened for centuries, but what is different today is that it affects many more industries. It's likely that we will adapt to technological changes by inventing entirely new types of work, and by taking advantage of our uniquely human capabilities.

Most scenarios about future AI are hypothetical, but they present us with existential questions. There is a never ending chase of the Machine and humans, and it will run in unparalleled till one conquers the other.

BIBLIOGRAPHY

- https://ai.stackexchange.com/questions/3494/why-is-python-such-a-popular-language-in-the-ai-field
- https://www.codementor.io/djangostars/reasons-why-python-is-good-for-ai-and-ml-wph92ed5n
- https://www.shponline.co.uk/technology/artificial-intelligence-discussion-and-conclusions/