Variables and Data Types

What is a variable

In [ ]:
```
#VARIABLES

# How to Declare and use a Variable

x = 22

# So the 22 is being stored in memory - not currently being used

print(x)

#When we print or call the variable it pulls that value from memory and displays it for us

# It will automatically assign a data type for your variable.

type(x)

#Note: We will be going much more in depth into data types in the next section

# Case Sensitivity

y = 'Chocolate'

Y = 'Mint Chocolate Chip'

print(Y)
```

In [ ]:

In [ ]:

In [ ]:

Statements & Operators

In [ ]:
```
# IF STATEMENTS

#Let's try this out


price = 75


#We want to categorize this person's age - so we will say

if price < 10:
    print('The price is ' + str(price) + ' dollars. This is Cheap') #Add the str() after showing how it won't
elif price < 40:
    print('The price is ' + str(price) + ' dollars. This is a not very expensive')
elif price < 100:
    print('The price is ' + str(price) + ' dollars. This is somewhat expensive')
else:
    print('The price is ' + price + ' dollars. This is very expensive!')
```

In [ ]:
```
# a list with all of our sales and we want to automatically split those into different lists based on their s

small_sale = []
medium_sale = []
large_sale = []

sale = 200

if sale < 20:
    small_sale.append(sale)
elif sale < 50:
    medium_sale.append(sale)
else:
    large_sale.append(sale)


#print these out and see what we have - each sale is put into the correct lists
print('Small Sale: ', small_sale)
print('Medium Sale: ', medium_sale)
print('Large Sale: ', large_sale)
```

In [ ]:
```
# FOR LOOPS
sale_amount = [1,20.70,35.2,50,100,125,70.59,106.21,700]

small_sale = []
medium_sale = []
large_sale = []


for sale in sale_amount:
    if sale < 50:
        small_sale.append(sale)
    elif sale < 100:
        medium_sale.append(sale)
    else:
        large_sale.append(sale)


#print these out and see what we have - each sale is put into the correct lists
print('Small Sale: ', small_sale)
print('Medium Sale: ', medium_sale)
print('Large Sale: ', large_sale)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Functions

In [1]:
```
def first_func():
    print('We called our first function!')

#how you call your function and it will run your user-defined function - and if it has an output you'll see i

first_func()
```

```
We called our first function!
```

In [ ]:
```
#calling the function, but inputting the correct argument when calling it.

def first_func(name):
    print(name,'did it!')

first_func('Keeno')


#specify that a function has to take an arguement, the user who calls it (or when you call it later) has to i
# an argument to call it properly
```

In [ ]:
```
# PRINT VS RETURN


#So if we write this function like this - the return acts almost like the print statement

def return_example():
    return 100

return_example()




def print_example():
    print(100)


print_example()


#For this we will get an error
print_example() + 100

#This will give us an output and perform the calculation
return_example() + 100
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Working with Packages and Libraries

Popular Data Analysis Libraries Pandas Polars Seaborn Numpy Matplotlib

In [4]:
```
# Now we will try a context manager - the preferred method - it starts with the with statement

with open(r"C:\Users\keenog\OneDrive\Documents\Tutorials\Pandas Folder\laptop_prices.csv", 'r') as read_file:
    print(read_file.read()) #once this block of code finishes the __exit__ automatically gets
    #called and it closes out the file for you.
```

```
laptop_ID,Company,Product,TypeName,Inches,ScreenResolution,Cpu,Ram,Memory,Gpu,OpSys,Weight,Price_usd
1,Apple,MacBook Pro,Ultrabook,13.3,IPS Panel Retina Display 2560x1600,Intel Core i5 2.3GHz,8GB,128GB SSD,Int
el Iris Plus Graphics 640,macOS,1.37kg,1473.659
2,Apple,Macbook Air,Ultrabook,13.3,1440x900,Intel Core i5 1.8GHz,8GB,128GB Flash Storage,Intel HD Graphics 6
000,macOS,1.34kg,988.834
3,HP,250 G6,Notebook,15.6,Full HD 1920x1080,Intel Core i5 7200U 2.5GHz,8GB,256GB SSD,Intel HD Graphics 620,N
o OS,1.86kg,632.5
4,Apple,MacBook Pro,Ultrabook,15.4,IPS Panel Retina Display 2880x1800,Intel Core i7 2.7GHz,16GB,512GB SSD,AM
D Radeon Pro 455,macOS,1.83kg,2791.195
5,Apple,MacBook Pro,Ultrabook,13.3,IPS Panel Retina Display 2560x1600,Intel Core i5 3.1GHz,8GB,256GB SSD,Int
el Iris Plus Graphics 650,macOS,1.37kg,1983.96
6,Acer,Aspire 3,Notebook,15.6,1366x768,AMD A9-Series 9420 3GHz,4GB,500GB HDD,AMD Radeon R5,Windows 10,2.1kg,
440
7,Apple,MacBook Pro,Ultrabook,15.4,IPS Panel Retina Display 2880x1800,Intel Core i7 2.2GHz,16GB,256GB Flash
Storage,Intel Iris Pro Graphics,Mac OS X,2.04kg,2353.967
8,Apple,Macbook Air,Ultrabook,13.3,1440x900,Intel Core i5 1.8GHz,8GB,256GB Flash Storage,Intel HD Graphics 6
000,macOS,1.34kg,1274.57
9,Asus,ZenBook UX430UN,Ultrabook,14,Full HD 1920x1080,Intel Core i7 8550U 1.8GHz,16GB,512GB SSD,Nvidia GeFor
ce MX150,Windows 10,1.3kg,1644.5
```

In [5]:
```
import pandas as pd

pd.set_option('display.max.rows', 1350)
pd.set_option('display.max.columns', 15)

df = pd.read_csv(r"C:\Users\keenog\OneDrive\Documents\Tutorials\Pandas Folder\laptop_prices.csv", encoding =
df
```

Out[5]:

	laptop_ID	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys
0	1	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS
1	2	Apple	Macbook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS
2	3	HP	250 G6	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS
3	4	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS
4	5	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS
5	6	Acer	Aspire 3	Notebook	15.6	1366x768	AMD A9-Series 9420 3GHz	4GB	500GB HDD	AMD Radeon R5	Windows 10

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Manipulating Data in Files

In [6]:
```
df[df['Company'] == 'Apple']
```

Out[6]:

	laptop_ID	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price_u
0	1	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	1473.6
1	2	Apple	Macbook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	988.8
3	4	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg	2791.1
4	5	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	1983.9
6	7	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.2GHz	16GB	256GB Flash Storage	Intel Iris Pro Graphics	Mac OS X	2.04kg	2353.9
7	8	Apple	Macbook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	256GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	1274.5
12	13	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.8GHz	16GB	256GB SSD	AMD Radeon Pro 555	macOS	1.83kg	2683.9
14	15	Apple	MacBook 12"	Ultrabook	12.0	IPS Panel Retina Display 2304x1440	Intel Core M m3 1.2GHz	8GB	256GB SSD	Intel HD Graphics 615	macOS	0.92kg	1388.6
15	16	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	256GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	1670.4
17	18	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.9GHz	16GB	512GB SSD	AMD Radeon Pro 560	macOS	1.83kg	3143.8
26	27	Apple	MacBook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.6GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	Mac OS X	1.35kg	1208.9

	laptop_ID	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price_u
34	35	Apple	MacBook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.6GHz	8GB	256GB Flash Storage	Intel HD Graphics 6000	Mac OS X	1.35kg	1097.8
45	46	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.0GHz	8GB	256GB SSD	Intel Iris Graphics 540	macOS	1.37kg	1560.9
81	83	Apple	MacBook 12"	Ultrabook	12.0	IPS Panel Retina Display 2304x1440	Intel Core i5 1.3GHz	8GB	512GB SSD	Intel HD Graphics 615	macOS	0.92kg	1661.0
249	254	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	512GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	2244.0
270	275	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.9GHz	8GB	512GB SSD	Intel Iris Graphics 550	macOS	1.37kg	2154.7
794	803	Apple	MacBook 12"	Ultrabook	12.0	IPS Panel Retina Display 2304x1440	Intel Core M 1.2GHz	8GB	512GB Flash Storage	Intel HD Graphics 5300	Mac OS X	0.920kg	1281.5
1069	1084	Apple	MacBook 12"	Ultrabook	12.0	IPS Panel Retina Display 2304x1440	Intel Core M 1.1GHz	8GB	256GB Flash Storage	Intel HD Graphics 515	Mac OS X	0.920kg	1430.0
1193	1211	Apple	MacBook 12"	Ultrabook	12.0	IPS Panel Retina Display 2304x1440	Intel Core M 1.1GHz	8GB	256GB Flash Storage	Intel HD Graphics 5300	Mac OS X	0.920kg	1279.3
1210	1228	Apple	MacBook 12"	Ultrabook	12.0	IPS Panel Retina Display 2304x1440	Intel Core M 1.2GHz	8GB	512GB Flash Storage	Intel HD Graphics 515	Mac OS X	0.920kg	1406.9
1234	1252	Apple	MacBook Air	Ultrabook	11.6	1366x768	Intel Core i5 1.6GHz	4GB	256GB Flash Storage	Intel HD Graphics 6000	Mac OS X	1.08kg	1054.9

In [7]:
```
df[(df['Company'] == 'Apple') | (df['TypeName'] == 'Ultrabook')]
```

Out[7]:

	laptop_ID	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight
0	1	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg
1	2	Apple	Macbook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg
3	4	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg
4	5	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg
6	7	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.2GHz	16GB	256GB Flash Storage	Intel Iris Pro Graphics	Mac OS X	2.04kg

In [9]:
```
df.groupby('Company')['Price_usd'].mean().sort_values(ascending=False)
```

Out[9]:
```
Company
Razer        3680.757143
LG           2308.900000
MSI          1901.798963
Google       1845.433333
Microsoft    1773.539167
Apple        1720.618429
Huawei       1566.400000
Samsung      1554.788889
Toshiba      1394.593750
Dell         1304.675889
Xiaomi       1246.808750
Asus         1214.586304
Lenovo       1195.022889
HP           1174.552339
Fujitsu       801.900000
Acer          689.453408
Chuwi         345.726333
Mediacom      324.500000
Vero          239.167500
Name: Price_usd, dtype: float64
```

In [ ]:

In [ ]:

Data Cleaning

In [10]:
```
import pandas as pd

data = {
    'student_id': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'full_name': ['Harry Potter', 'Hermione Granger', 'Ron Weasley', 'Draco Malfoy', 'Luna Lovegood', 'Nevill
    'phone_number': ['(123) 456-7890', '555-123-4567', '9876543210', '123.456.7890', '(123)-4567890', '987-65
}

Bad_Numbers = pd.DataFrame(data)
Bad_Numbers
```

Out[10]:

	student_id	full_name	phone_number
0	1	Harry Potter	(123) 456-7890
1	2	Hermione Granger	555-123-4567
2	3	Ron Weasley	9876543210
3	4	Draco Malfoy	123.456.7890
4	5	Luna Lovegood	(123)-4567890
5	6	Neville Longbottom	987-654-3210
6	7	Ginny Weasley	123 456 7890
7	8	Fred Weasley	1234567890
8	9	George Weasley	+7890123456
9	10	Cho Chang	012-345-6789

In [13]:
```
data = {
    'student_id': [1, 2, 3, 4, 5, 6, 7, 8, 9],
    'full_name': ['Harry Potter', 'Hermione Granger', 'Ron Weasley', 'Draco Malfoy', 'Luna Lovegood', 'Nevill
    'address': ['12 Grimmauld Place, London, England', '4 Privet Drive, Surrey, England', 'The Burrow, Devon,
}

Student_Address = pd.DataFrame(data)
Student_Address
```

Out[13]:

	student_id	full_name	address
0	1	Harry Potter	12 Grimmauld Place, London, England
1	2	Hermione Granger	4 Privet Drive, Surrey, England
2	3	Ron Weasley	The Burrow, Devon, England
3	4	Draco Malfoy	124 Slitheren Street, Wiltshire, England
4	5	Luna Lovegood	869 Light Way, London, England
5	6	Neville Longbottom	4834 Street, Hampshire, England
6	7	Ginny Weasley	The Burrow, Devon, England
7	8	Fred Weasley	The Burrow, Devon, England
8	9	George Weasley	The Burrow, Devon, England

In [14]:
```
# Split the address column into separate columns
Student_Address[['street_address', 'city', 'country']] = Student_Address['address'].str.split(',', expand=Tru
Student_Address
```

Out[14]:

	student_id	full_name	address	street_address	city	country
0	1	Harry Potter	12 Grimmauld Place, London, England	12 Grimmauld Place	London	England
1	2	Hermione Granger	4 Privet Drive, Surrey, England	4 Privet Drive	Surrey	England
2	3	Ron Weasley	The Burrow, Devon, England	The Burrow	Devon	England
3	4	Draco Malfoy	124 Slitheren Street, Wiltshire, England	124 Slitheren Street	Wiltshire	England
4	5	Luna Lovegood	869 Light Way, London, England	869 Light Way	London	England
5	6	Neville Longbottom	4834 Street, Hampshire, England	4834 Street	Hampshire	England
6	7	Ginny Weasley	The Burrow, Devon, England	The Burrow	Devon	England
7	8	Fred Weasley	The Burrow, Devon, England	The Burrow	Devon	England
8	9	George Weasley	The Burrow, Devon, England	The Burrow	Devon	England

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Data Visualization

In [17]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [18]:
```
housing = pd.read_csv(r"C:\Users\keenog\OneDrive\Documents\Tutorials\Pandas Folder\real_estate.csv", encoding
housing
```

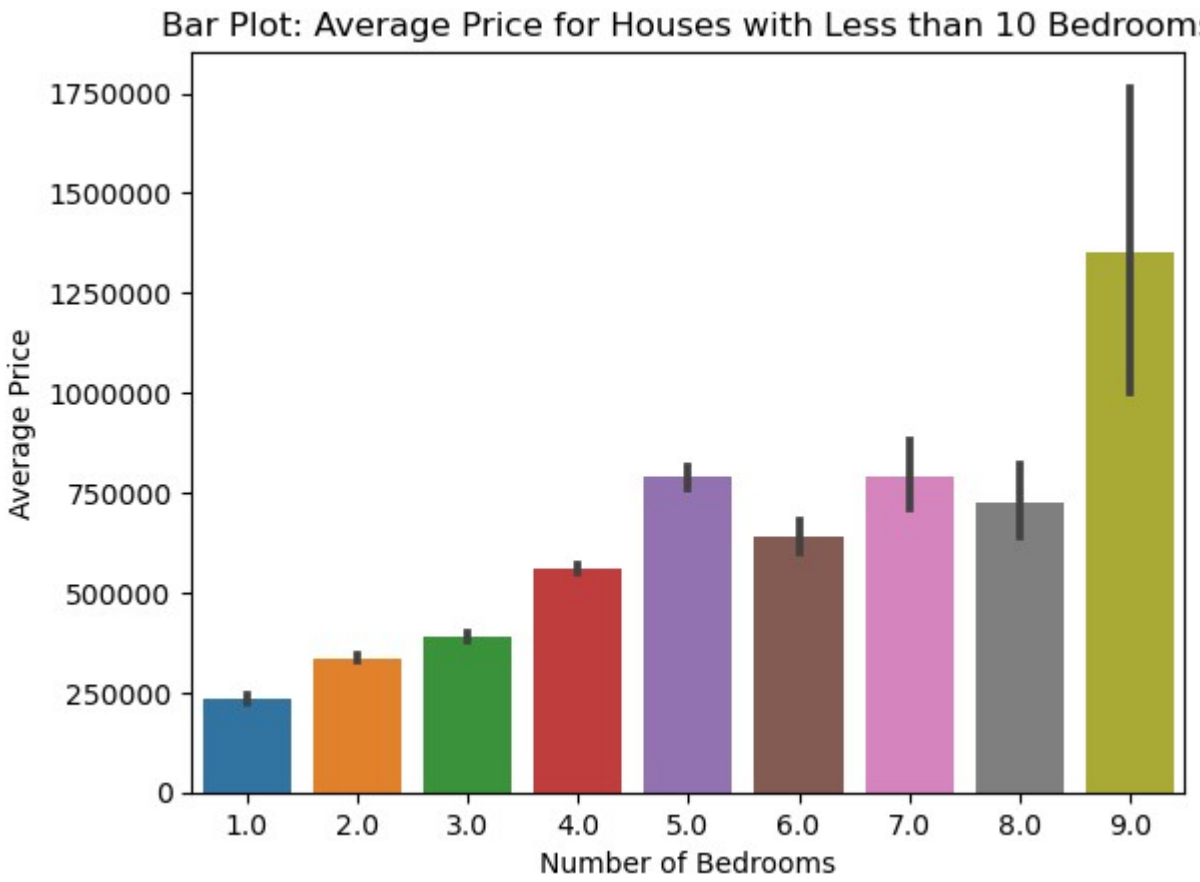
Out[18]:

	status	bed	bath	acre_lot	city	state	zip_code	house_size	prev_sold_date	price
0	for_sale	3.0	2.0	0.12	Adjuntas	Puerto Rico	601.0	920.0	NaN	105000.0
1	for_sale	4.0	2.0	0.08	Adjuntas	Puerto Rico	601.0	1527.0	NaN	80000.0
2	for_sale	2.0	1.0	0.15	Juana Diaz	Puerto Rico	795.0	748.0	NaN	67000.0
3	for_sale	4.0	2.0	0.10	Ponce	Puerto Rico	731.0	1800.0	NaN	145000.0
4	for_sale	6.0	2.0	0.05	Mayaguez	Puerto Rico	680.0	NaN	NaN	65000.0
...	...	...	...	...	...	...	...	...	...	...
99995	for_sale	3.0	3.0	NaN	Hudson	Massachusetts	1749.0	2864.0	NaN	749900.0
99996	for_sale	2.0	1.0	0.34	Auburn	Massachusetts	1501.0	1075.0	1999-06-07	349900.0
99997	for_sale	3.0	2.0	1.01	Shrewsbury	Massachusetts	1545.0	1632.0	1995-09-27	549000.0
99998	for_sale	3.0	2.0	0.12	Worcester	Massachusetts	1604.0	1332.0	2000-09-11	299000.0
99999	for_sale	3.0	3.0	21.67	Grafton	Massachusetts	1536.0	1846.0	2020-10-06	535000.0

100000 rows × 10 columns

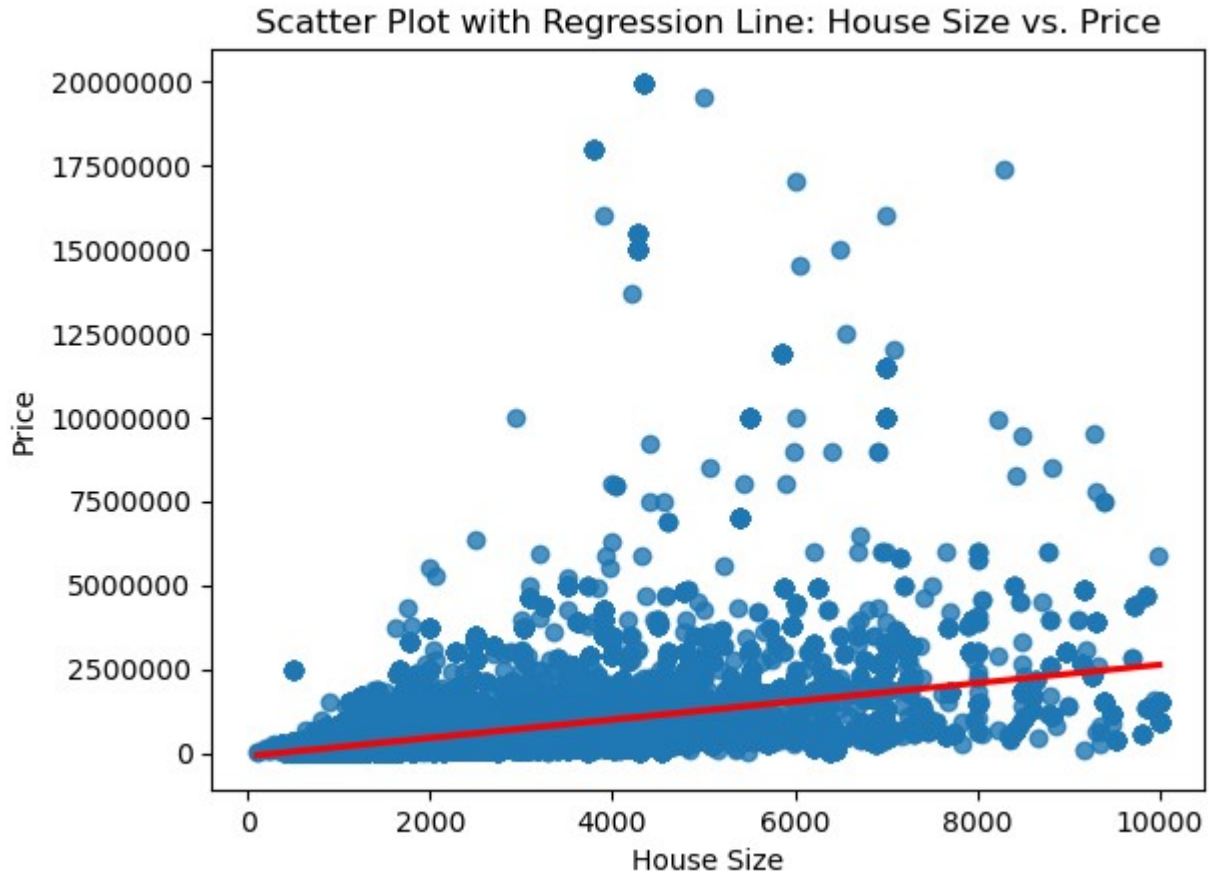
In [19]:

```
# Filter the dataframe to include only houses with less than 10 bedrooms
filtered_df = housing[housing['bed'] < 10]

# Create the bar plot
sns.barplot(x='bed', y='price', data=filtered_df, estimator=np.mean)

# Set the plot labels and title
plt.xlabel('Number of Bedrooms')
plt.ylabel('Average Price')
plt.title('Bar Plot: Average Price for Houses with Less than 10 Bedrooms')

# Remove the "1e6" from the top of the visualization
plt.ticklabel_format(style='plain', axis='y')

# Display the plot
plt.show()



# They are error bars. They can display either confidence intervals or the standard deviation.
# The bar plot shows an aggregation of some values.
```


Bar Plot: Average Price for Houses with Less than 10 Bedrooms
Average Price
1750000
1500000
1250000
1000000
750000
500000
250000
0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
Number of Bedrooms

In [20]:

```
filtered_df = housing[housing['house_size'] < 10000]
sns.regplot(x='house_size', y='price', data=filtered_df,line_kws={'color': 'red'})
plt.xlabel('House Size')
plt.ylabel('Price')
plt.title('Scatter Plot with Regression Line: House Size vs. Price')

# Remove the "1e6" from the top of the visualization
plt.ticklabel_format(style='plain', axis='y')

plt.show()
```



In [ ]:

In [ ]:

In [ ]:

Web Scraping

In [ ]:
```
from bs4 import BeautifulSoup
import requests
```

In [ ]:
```
url = 'https://www.worldometers.info/world-population/population-by-country/'

page = requests.get(url)

soup = BeautifulSoup(page.text, 'html')
```

In [ ]:
```
print(soup)
```

In [ ]:
```
soup.find('table')
```

In [ ]:
```
world_table = soup.find('table', class_ = 'table table-striped table-bordered')
```

In [ ]:
```
world_table.find_all('th')
```

In [ ]:
```
world_titles = world_table.find_all('th')
```

In [ ]:
```
world_table_titles = [title.text for title in world_titles]

print(world_table_titles)
```

In [ ]:
```
import pandas as pd
```

In [ ]:
```
df = pd.DataFrame(columns = world_table_titles)
df
```

In [ ]:
```
world_table.find_all('td')
```

In [ ]:
```
world_table.find_all('tr')
```

In [ ]:
```
column_data = world_table.find_all('tr')
```

In [ ]:
```
for row in column_data:
    row_data = row.find_all('td')
    individual_row_data = [data.text for data in row_data]

print(individual_row_data)
```

In [ ]:
```
for row in column_data[1:]:
    row_data = row.find_all('td')
    individual_row_data = [data.text for data in row_data]

    length = len(df)
    df.loc[length] = individual_row_data

df
```

In [ ]:
```
df
```

In [ ]:
```
df.to_csv(r'C:\Users\keenog\OneDrive\Documents\Python Tutorials\Pandas_Folder\World_Population_Scraped.csv',
```

In [ ]:
```

```