End of session 2

In [25]:
```
my_list1 = [2,3.5,4+2j,"ALI",[1,2,3,4],7,7,7]
my_list2 = [2,7,555,666]
print(my_list1 + my_list2)

my_list2.append([777])
print("my_list2.append:",my_list2)
my_list2.extend([11,22,33])
print("my_list2.extend:",my_list2)


my_set1 = {2,3.5,4+2j,"ALI",7,7,7}
my_set2 = {2,7,555,666}
my_set2.add(888)
print(my_set1)
print("Intersection: ",my_set1 & my_set2)
print("Union: ",my_set1 | my_set2)


my_dic1 = {"firstName":"Ali","lastName":"Reza zadeh"}
my_dic2 = {"age":31, "scores":[17,18.8,16,19]}
my_dic3 = {"set":{"A","B","C"}}

print(my_dic1 | my_dic2)
my_dic1.update(my_dic3)
print(my_dic1)

my_tuple = (1,3,7,9)
print("my_tuple: ",my_tuple)
```

```
[2, 3.5, (4+2j), 'ALI', [1, 2, 3, 4], 7, 7, 7, 2, 7, 555, 666]
my_list2.append: [2, 7, 555, 666, [777]]
my_list2.extend: [2, 7, 555, 666, [777], 11, 22, 33]
{2, 'ALI', 3.5, 7, (4+2j)}
Intersection:  {2, 7}
Union:  {2, 'ALI', 3.5, 7, (4+2j), 555, 888, 666}
{'firstName': 'Ali', 'lastName': 'Reza zadeh', 'age': 31, 'scores': [17, 18.8, 16, 1
9]}
{'firstName': 'Ali', 'lastName': 'Reza zadeh', 'set': {'A', 'C', 'B'}}
my_tuple:  (1, 3, 7, 9)
```

In [26]:
```
my_list1 = [11,22,33,44,55,66,77,88,99,[111,222,333,444]]

print(my_list1[0],my_list1[-1],my_list1[4])
print(my_list1[0:3])
print(my_list1[:3])
print(my_list1[6:])
print(my_list1[-3:-1])
print(my_list1[-1])
print(len(my_list1))
print(my_list1[-1][-2:])

my_list1[2]=300
```

```
my_list1[3:5]= [8888,9999]
print(my_list1)
```

```
11 [111, 222, 333, 444] 55
[11, 22, 33]
[11, 22, 33]
[77, 88, 99, [111, 222, 333, 444]]
[88, 99]
[111, 222, 333, 444]
10
[333, 444]
[11, 22, 300, 8888, 9999, 66, 77, 88, 99, [111, 222, 333, 444]]
```

In [27]:
```
my_str = "11,22,33,44,55,66,77,88,99"

print(my_str[0],my_str[-1],my_str[4])
print(my_str[0:3])
print(my_str[:3])
print(my_str[6:])
print(my_str[-3:-1])
print(len(my_str))
```

```
1 9 2
11,
11,
33,44,55,66,77,88,99
,9
26
```

In [28]:
```
my_dic1 = {"firstName":"Ali","lastName":"Reza zadeh"}
my_dic2 = {"age":31, "scores":[17,18.8,16,19]}

my_dic1.update(my_dic2)
print(my_dic1)
my_dic1["firstName"]="Hossein"
print(my_dic1)

my_dic3 = {}
my_dic3["Insterted"] = "some data"
print(my_dic3)
```

```
{'firstName': 'Ali', 'lastName': 'Reza zadeh', 'age': 31, 'scores': [17, 18.8, 16, 1
9]}
{'firstName': 'Hossein', 'lastName': 'Reza zadeh', 'age': 31, 'scores': [17, 18.8, 1
6, 19]}
{'Insterted': 'some data'}
```

In [ ]:
```
# // Immutable // #
my_tuple = (1,3,7,9)
print("my_tuple: ",my_tuple)
print(my_tuple[0:3])
print(my_tuple[-1])
print(my_tuple[:-2])

my_tuple[0]=111
```

```
my_tuple:  (1, 3, 7, 9)
(1, 3, 7)
9
(1, 3)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[31], line 8
      5 print(my_tuple[-1])
      6 print(my_tuple[:-2])
----> 8 my_tuple[0]=111
     10 my_tuple = (2,3)

TypeError: 'tuple' object does not support item assignment
```

Control Structure

In [47]:
```python
# Start and stop
for i in range(5):
    print(i)
# Output: 0 1 2 3 4
print("***"*10)

# Start and stop
lst1 = []
for i in range(2, 6):
    print(i)
    lst1.append(i)
# Output: 2 3 4 5
print(lst1)
print("***"*10)

# With a step
lst2=[]
for i in range(0, 10, 2):
    lst2+=[i]
    print(i)
# Output: 0 2 4 6 8
print(lst2)
print("***"*10)

# Going backwards
str1 = ""
for i in range(10, 0, -2):
    str1 += str(i)
    print(i)
# Output: 10 8 6 4 2
print(str1)
print("***"*10)

# Going backwards
iters = range(10, 0, -2)
for i in iters:
    print(i)
# Output: 10 8 6 4 2
```

```
0
1
2
3
4
****************************
2
3
4
5
[2, 3, 4, 5]
****************************
0
2
4
6
8
[0, 2, 4, 6, 8]
****************************
10
8
6
4
2
108642
****************************
10
8
6
4
2
```

In [50]:
```
for i in range(0, 10):
    if (i%3)==0:
        continue
    print(i)
    if i>7:
        break
```

```
1
2
4
5
7
8
```

In [52]:
```
print("my_list loop:")
my_list = [11,22,33,[111,222,333,444]]
for item in my_list:
    print(item)

print("my_str loop:")
my_str = "11,22,33"
for item in my_str:
    print(item)
```

```
print("my_tuple loop:")
my_tuple = (1,3,7,9)
for item in my_tuple:
    print(item)
```

```
my_list loop:
11
22
33
[111, 222, 333, 444]
my_str loop:
1
1
,
2
2
,
3
3
my_tuple loop:
1
3
7
9
```

In [62]:
```
my_dic1 = {"firstName":"Ali",
           "lastName":"Reza zadeh",
           "scores":[1,2,3,4,5,6,7,8,9],
           "old_data":my_dic2}

print("my_dic1 loop:")
for item in my_dic1:
    # print(item)
    print("item:",item,"val:",my_dic1[item])

print("\n\n")
print("my_dic1.items() loop:")
for item in my_dic1.items():
    print(item)

print("\n\n")
print("my_dic1.keys() loop:")
for item in my_dic1.keys():
    print(item)

print("\n\n")
print("y_dic1.values() loop:")
for item in my_dic1.values():
    print(item)
```

```
my_dic1 loop:
item: firstName val: Ali
item: lastName val: Reza zadeh
item: scores val: [1, 2, 3, 4, 5, 6, 7, 8, 9]
item: old_data val: {'age': 31, 'scores': [17, 18.8, 16, 19]}
```

```
my_dic1.items() loop:
('firstName', 'Ali')
('lastName', 'Reza zadeh')
('scores', [1, 2, 3, 4, 5, 6, 7, 8, 9])
('old_data', {'age': 31, 'scores': [17, 18.8, 16, 19]})
```

```
my_dic1.keys() loop:
firstName
lastName
scores
old_data
```

```
y_dic1.values() loop:
Ali
Reza zadeh
[1, 2, 3, 4, 5, 6, 7, 8, 9]
{'age': 31, 'scores': [17, 18.8, 16, 19]}
```

In [63]:
```
from pprint import pprint
print(my_dic1)
pprint(my_dic1)
```

```
{'firstName': 'Ali', 'lastName': 'Reza zadeh', 'scores': [1, 2, 3, 4, 5, 6, 7, 8,
9], 'old_data': {'age': 31, 'scores': [17, 18.8, 16, 19]}}
{'firstName': 'Ali',
 'lastName': 'Reza zadeh',
 'old_data': {'age': 31, 'scores': [17, 18.8, 16, 19]},
 'scores': [1, 2, 3, 4, 5, 6, 7, 8, 9]}
```

In [66]:
```
age = 17

if age < 13:
    print("You are a kid.")
elif age < 18:
    print("You are a teenager.")
elif age>=18 and age<=50:
    print("Middle-age")
else:
    print("You are an adult.")
```

```
You are a teenager.
```

In [70]:
```
age_str = input("Please enter your age (only years in integer number):")

try:
```

```
    age = int(age_str)
    if age < 13:
        print("You are a kid.")
    elif age < 18:
        print("You are a teenager.")
    elif age>=18 and age<=50:
        print("Middle-age")
    else:
        print("You are an adult.")
except:
    print("<<<",age_str,">>>","Entered number is not in valid format <integer requi
```

```
Middle-age
```

In [ ]:
```
try:
    x = int(input("Enter a number: "))
    print(10 / x)
except ZeroDivisionError:
    print("You can't divide by zero!")
except ValueError:
    print("That's not a number!")
```

In [81]:
```
try:
    file = open("myfile.txt")
except FileNotFoundError:
    print("File not found!")
finally:
    print("Done trying.")
```

```
File not found!
Done trying.
```

In [84]:
```
try:
    x = 3/0

except Exception as e:
    print(e)
```

```
division by zero
```

In [90]:
```
x = 5
print(type(x))
if type(x)==int:
    print("The var is INT")

type(x)
```

```
<class 'int'>
The var is INT
```

Out[90]:
```
int
```

In [83]:
```
x = [11,22,33,44]
for i,j in enumerate(x):
    print(i,j)
```

```
0 11
1 22
2 33
3 44
```

In [97]:
```python
x = [10,20,30]
print(sum(x))
```

```
60
```

In [100…
```python
def day_of_week(day):
    match day:
        case "11AA":
            return "Monday"
        case 2:
            return "Tuesday"
        case 3:
            return "Wednesday"
        case 4:
            return "Thursday"
        case 5:
            return "Friday"
        case 6:
            return "Saturday"
        case 7:
            return "Sunday"
        case _:
            return "Invalid day"


print(day_of_week("11AA"))  # Output: Wednesday
```

```
Monday
```

In [103…
```python
def describe_point(point):
    match point:
        case (0, 0):
            return "Origin"
        case (0, y):
            return f"Y-axis at {y}"
        case (x, 0):
            return f"X-axis at {x}"
        case (x, y):
            return f"Point at ({x}, {y})"
        case _:
            return "Not a valid point"

print(describe_point((0, 5)))
print(describe_point((0, 0)))
print(describe_point((3, 4)))
print(describe_point("ALI"))
```

```
Y-axis at 5
Origin
Point at (3, 4)
Not a valid point
```

In [ ]:
```python
def handle_data(data):
    match data:
        case {"type": "person", "name": name, "age": age}:
            return f"Person {name}, {age} years old"
        case {"type": "company", "name": name}:
            return f"Company {name}"
        case _:
            return "Unknown data"

print(handle_data({"type": "person", "name": "Alice", "age": 30}))
print(handle_data({"type": "company", "name": "apple"}))
print(handle_data({"type": "company", "name": "apple"}))
```

```
Person Alice, 30 years old
Company apple
Unknown data
```

In [17]:
```python
fruits = {"name": "Alice", "age": 25, "city": "London"}


# with open("fruits.txt", "w") as f:
#     # for fruit in fruits:
#     #     f.write(fruit + "\n")  # write each on a new line
#      f.write(str(fruits))
# print("List saved!")
dict=str(fruits)
```

In [ ]:
```python
# Example string
text = "Hello, this is some sample text."

# Open a file in write mode ('w')
with open("myfile.txt", "w") as f:
    f.write(text)

print("String saved!")




fruits = ["apple", "banana", "cherry"]

with open("fruits.txt", "w") as f:
    for fruit in fruits:
        f.write(fruit + "\n")  # write each on a new line

print("List saved!")




import json

person = {"name": "Alice", "age": 25, "city": "London"}

with open("person.json", "w") as f:
```

```
    json.dump(person, f, indent=10)

print("Dictionary saved as JSON!")
```

```
String saved!
List saved!
Dictionary saved as JSON!
```

In [21]:
```
with open("myfile.txt", "r") as f:
    data = f.read()
    print(data)
```

```
Hello, this is some sample text.
```

Part 2 Python & Data science

In [37]:
```
data = [10, 15, 20, 25, 30]
average = sum(data) / len(data)
print("Average:", average)


import numpy as np

numbers = np.array([10, 15, 20, 25, 30])
print("Mean:", np.mean(numbers))
print("Squares:", numbers ** 2)

print(np.random.rand(2,2))
print(np.sum(np.random.rand(2,2),0))

x = np.ones((10,20,3))
# print(x)
print(x.shape)
y = np.ones_like(x)
print(y.shape)
```

```
Average: 20.0
Mean: 20.0
Squares: [100 225 400 625 900]
[[0.33409636 0.41330367]
 [0.45143866 0.23786457]]
[0.04425143 0.70424817]
(10, 20, 3)
(10, 20, 3)
```

In [44]:
```
from time import time

t1 = time()
for i in range(100000):
    x=100**100

print(time()-t1)
```

```
0.06453204154968262
```

In [46]:
```
eval("10+10")
```

Out[46]: 20

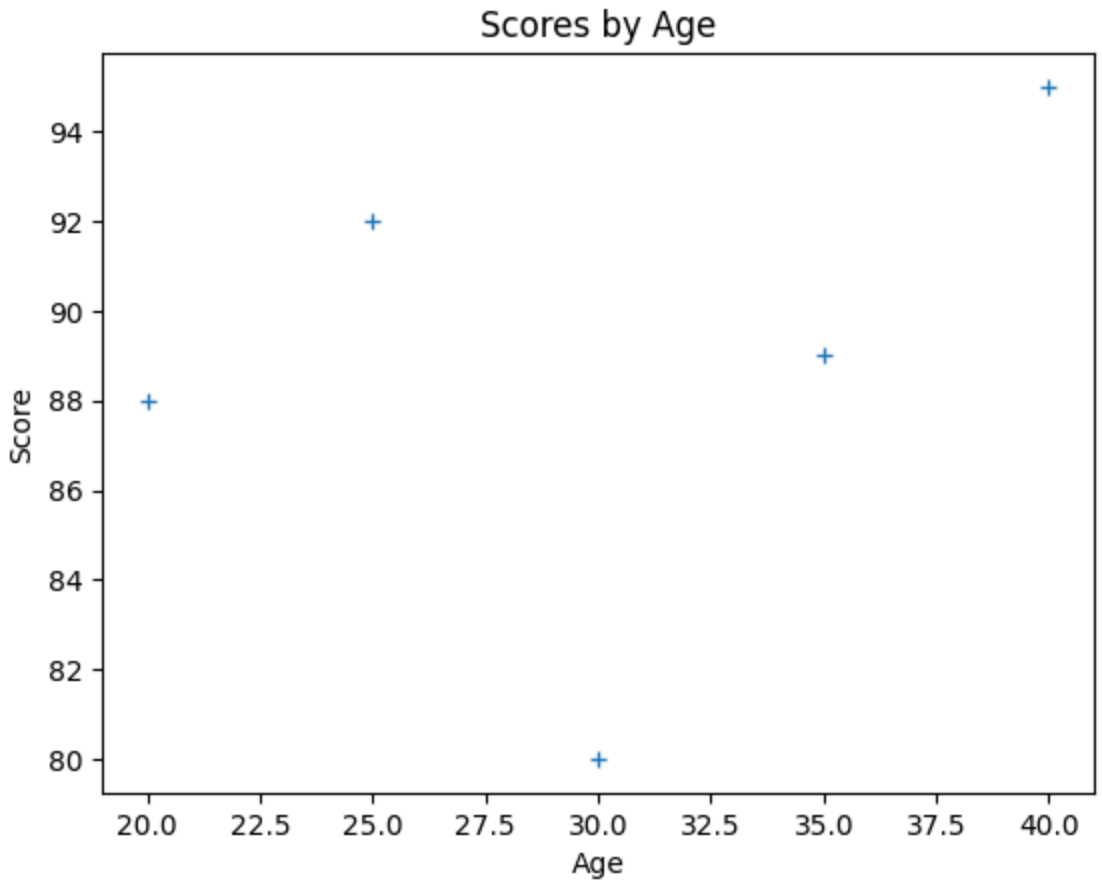
In [45]:
```
from datetime import datetime
print(datetime.now())
```

```
2025-10-30 16:39:54.272559
```

In [49]:
```
from pprint import pprint

data = {
    "Name": ["Alice", "Bob", "Charlie"],
    "Age": [25, 30, 35],
    "Score": [85, 90, 95]
}

pprint(data)


import pandas as pd

df = pd.DataFrame(data)
print(df)
print(df["Age"].mean())

### data = pd.read_csv("data.csv")
```

```
{'Age': [25, 30, 35],
 'Name': ['Alice', 'Bob', 'Charlie'],
 'Score': [85, 90, 95]}
      Name  Age  Score
0    Alice   25     85
1      Bob   30     90
2  Charlie   35     95
30.0
```

In [55]:
```
import matplotlib.pyplot as plt

ages = [20, 25, 30, 35, 40]
scores = [88, 92, 80, 89, 95]

plt.plot(ages, scores,'+')
plt.title("Scores by Age")
plt.xlabel("Age")
plt.ylabel("Score")
plt.show()
```

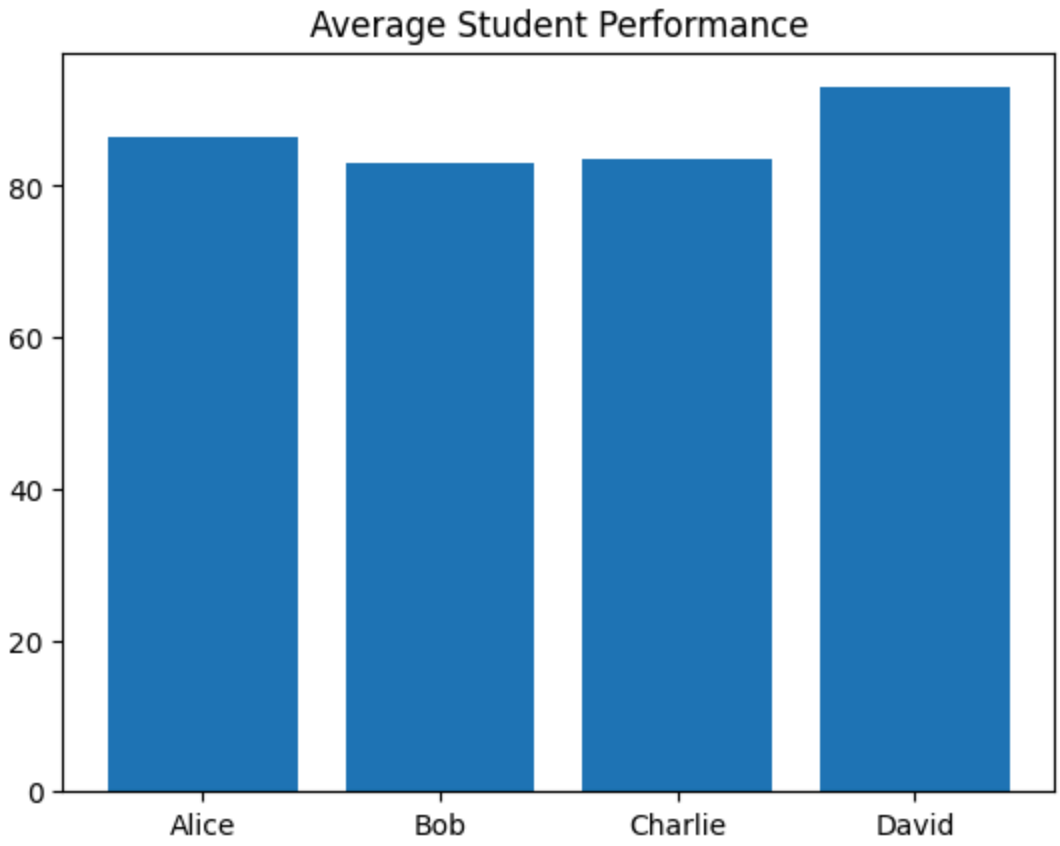


In [60]:
```
import numpy as np

n = np.array([1,2,3,4])
m = np.array([10,20,30,40])
print((n+m)/2)
```

```
[ 5.5 11.  16.5 22. ]
```

In [61]:
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.DataFrame({
    "Name": ["Alice", "Bob", "Charlie", "David"],
    "Math": [85, 90, 78, 92],
    "Science": [88, 76, 89, 94]
})

print(df)
df["Average"] = (df["Math"] + df["Science"]) / 2
print(df)

plt.bar(df["Name"], df["Average"])
plt.title("Average Student Performance")
plt.show()
```

```
      Name  Math  Science
0    Alice    85       88
1      Bob    90       76
2  Charlie    78       89
3    David    92       94
      Name  Math  Science  Average
0    Alice    85       88     86.5
1      Bob    90       76     83.0
2  Charlie    78       89     83.5
3    David    92       94     93.0
```



Mini Project: Data Queries & Casting with a Default Dataset

In [ ]:
```python
import pandas as pd
import seaborn as sns

# Load the dataset
df = sns.load_dataset("tips")

# Show first few rows
print(df.head())
```

```
   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 [11]:
```
print(df.info())       # data types and nulls
print("/#/"*10)
print(df.describe())   # numeric summary
print("/#/"*10)
print(df.shape)        # rows and columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype   
---  ------      --------------  -----   
 0   total_bill  244 non-null    float64 
 1   tip         244 non-null    float64 
 2   sex         244 non-null    category
 3   smoker      244 non-null    category
 4   day         244 non-null    category
 5   time        244 non-null    category
 6   size        244 non-null    int64   
dtypes: category(4), float64(2), int64(1)
memory usage: 7.4 KB
None
/#//#//#//#//#//#//#//#//#/
       total_bill         tip        size
count  244.000000  244.000000  244.000000
mean    19.785943    2.998279    2.569672
std      8.902412    1.383638    0.951100
min      3.070000    1.000000    1.000000
25%     13.347500    2.000000    2.000000
50%     17.795000    2.900000    2.000000
75%     24.127500    3.562500    3.000000
max     50.810000   10.000000    6.000000
/#//#//#//#//#//#//#//#//#/
(244, 7)
```

In [12]:
```
# 1️⃣ Filter rows (tips greater than $5)
big_tips = df[df["tip"] > 5]
print(big_tips.head())

# 2️⃣ Select specific columns
subset = df[["total_bill", "tip", "day"]]
print(subset.head())

# 3️⃣ Conditional query — dinner time, non-smokers only
dinner_ns = df[(df["time"] == "Dinner") & (df["smoker"] == "No")]
print(dinner_ns.head())

# 4️⃣ Average tip by day
avg_tips = df.groupby("day")["tip"].mean()
print(avg_tips)
```

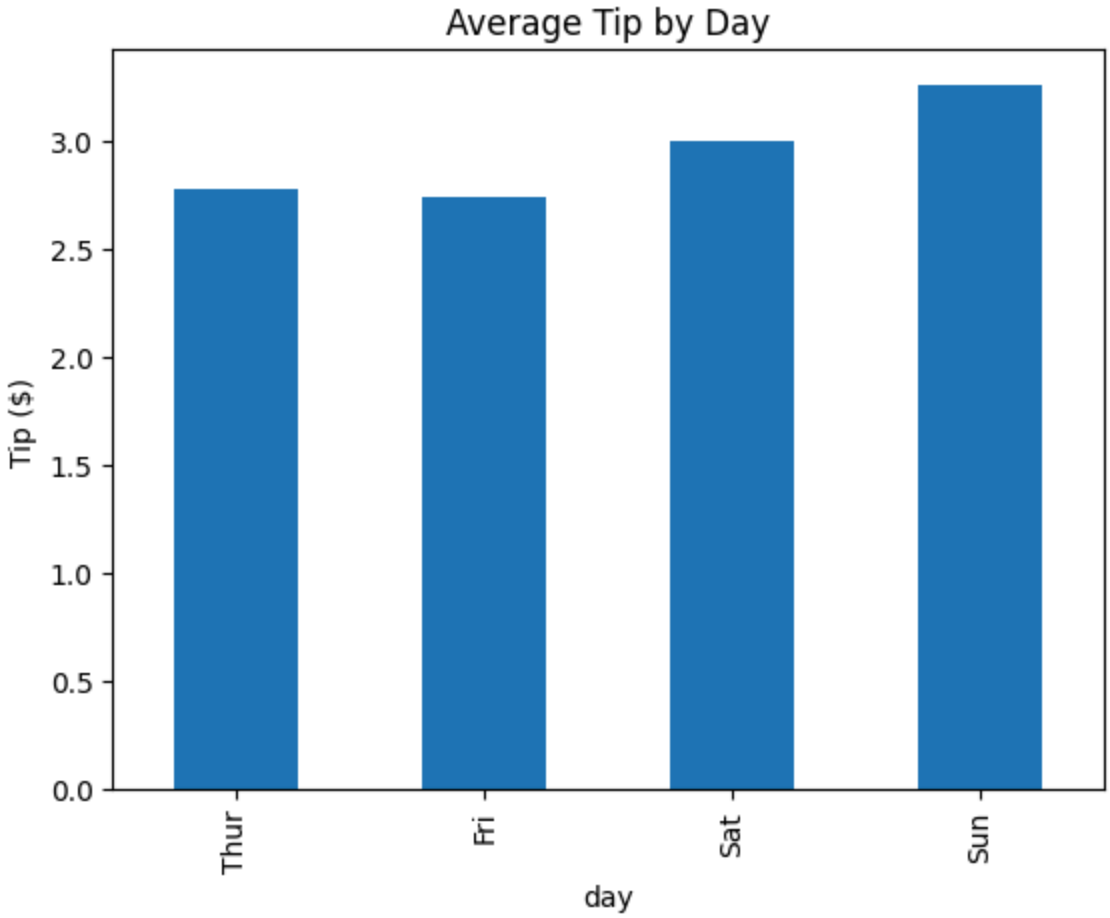
```
     total_bill   tip     sex smoker  day    time  size
23        39.42  7.58    Male     No  Sat  Dinner     4
44        30.40  5.60    Male     No  Sun  Dinner     4
47        32.40  6.00    Male     No  Sun  Dinner     4
52        34.81  5.20  Female     No  Sun  Dinner     4
59        48.27  6.73    Male     No  Sat  Dinner     4
   total_bill   tip  day
0       16.99  1.01  Sun
1       10.34  1.66  Sun
2       21.01  3.50  Sun
3       23.68  3.31  Sun
4       24.59  3.61  Sun
   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
day
Thur    2.771452
Fri     2.734737
Sat     2.993103
Sun     3.255132
Name: tip, dtype: float64
C:\Users\Admin\AppData\Local\Temp\ipykernel_8536\449550018.py:14: FutureWarning: The
default of observed=False is deprecated and will be changed to True in a future vers
ion of pandas. Pass observed=False to retain current behavior or observed=True to ad
opt the future default and silence this warning.
  avg_tips = df.groupby("day")["tip"].mean()
```

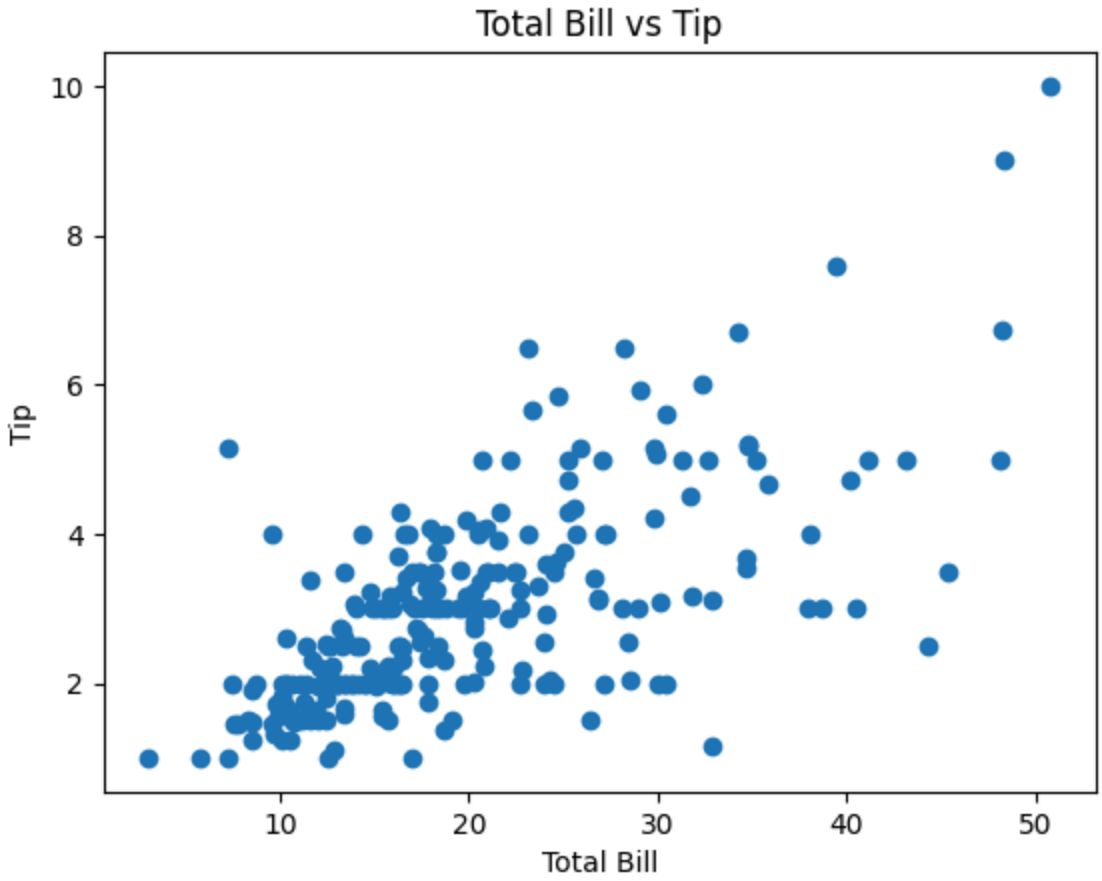
In [14]:

```python
# Check types
print(df.dtypes)
print("/##/"*10)
# Convert 'size' from int to float
df["size"] = df["size"].astype(float)

# Convert 'day' to category (saves memory)
df["day"] = df["day"].astype("category")

# Convert 'total_bill' to string (for demonstration)
df["total_bill_str"] = df["total_bill"].astype(str)

print(df.dtypes)
```

```
total_bill         float64
tip                float64
sex               category
smoker            category
day               category
time              category
size               float64
total_bill_str      object
dtype: object
/##//##//##//##//##//##//##//##//##//##/
total_bill         float64
tip                float64
sex               category
smoker            category
day               category
time              category
size               float64
total_bill_str      object
dtype: object
```

In [15]:
```python
import matplotlib.pyplot as plt

# Average tip by day
avg_tips.plot(kind="bar", title="Average Tip by Day")
plt.ylabel("Tip ($)")
plt.show()

# Relationship between total bill and tip
plt.scatter(df["total_bill"], df["tip"])
plt.title("Total Bill vs Tip")
plt.xlabel("Total Bill")
plt.ylabel("Tip")
plt.show()
```


Average Tip by Day
Tip ($)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Thur
Fri
Sat
Sun
day


Total Bill vs Tip
Tip
2
4
6
8
10
10
20
30
40
50
Total Bill

In [ ]:
```
df.to_csv("cleaned_tips.csv", index=False)
print("Saved to cleaned_tips.csv")
```