1. Write a Python program covering the following topics with concise examples

Q1.) installing python and Running python program

In [ ]:
```
"""Guide the user through installing Python from the official website (python.c
recommend two popular IDEs: PyCharm and Visual Studio Code."""
```

In [ ]:
```
#introduction to python and IDEs
Visit the Python official website:
Go to https://www.python.org in your web browser.

# on PYCHARM
https://www.jetbrains.com/pycharm/download/

## on VS CODE
https://code.visualstudio.com
```

In [1]:
```
## RUNNING PYTHON PRGORAMMING
a=4+4
print(a)
```

```
8
```

In [2]:
```
print("Hello world")
```

```
Hello world
```

Q.2) Variables and Data Types, Basic Input and Output

In [ ]:
```
"""Declare variables of different data types (int, float, str, list, tuple, dic
along with its data type. Prompt the user to enter their name using input(), th
message along with their name. """
```

In [4]:
```
integer= 3
type(integer)
```

Out[4]:
```
int
```

In [1]:
```
string="Mayuri"
type(string)
```

Out[1]:
```
str
```

In [6]:
```
float_value=18.3
type(float_value)
```

Out[6]:
```
float
```

In [7]:
```
list1=[1,2,3,4,5,6,7,8,9]
type(list1)
```

Out[7]:
```
list
```

In [8]:
```
set1= {"python", 'data science', 'ml', 'dl'}
type(set1)
```

Out[8]:
```
set
```

In [2]:
```
tuple1=("Mayuri","Esha")
type(tuple1)
```

Out[2]:
```
tuple
```

In [3]:
```
dictonary= {"name": "Mayuri", "age":'22',"std":"data science"}
type(dictonary)
```

Out[3]:
```
dict
```

In [4]:
```
name=input("enter your name:")
print("Name: ", name)
```

```
enter your name:Mayuri
Name:  Mayuri
```

In [5]:
```
# Simple greeting message
name = input("Please enter your name: ")
print("Hello, " + name + "!")
```

```
Please enter your name: Mayuri
Hello, Mayuri!
```

Q.3) Operators and Expressions, Conditional Statements

In [ ]:
```
"""Demonstrate arithmetic operations (+, -, *, /, //, %) and logical operations
if statement to check if a number entered by the user is positive, negative, or
```

In [1]:
```
num1=3
num2=18
Addition=num1+num2
multiply=num1*num2
substraction=num1-num2
division=num1/num2
reminder=num1//num2
divesir=num1%num2

print("Addition:", Addition)
print("substraction:", substraction)
print("multiplication:", multiply)
print("division:", division)
print("reminder:",reminder)
print("reminder:",reminder)
print("divesir:",divesir)
```

```
Addition: 21
substraction: -15
multiplication: 54
division: 0.16666666666666666
reminder: 0
reminder: 0
divesir: 3
```

Q.4) Loops, Functions and Parameters, Returning Values from Functions:

In [ ]:
```
"""Implement a for loop to iterate over a list of numbers and print each number
add that takes two parameters and returns their sum. Call the add function with
4, and print the result. Then, define a recursive function factorial to calcula
number (5) and print the result."""
```

In [44]:
```
integer=int(input("Enter the integer:"))
if integer >0:
    print("positive")
elif integer==0:
    print("zero")

else:
    print("negetive")
```

```
Enter the integer:18
positive
```

In [66]:
```
numbers=[1,2,3,4,5,6,7,8,9]
for i in numbers:
    print(i)

def add(a,b):
    return a + b
addition=add(3,4)
print("addition:",addition)
```

```
1
2
3
4
5
6
7
8
9
addition: 7
```

In [70]:
```
numbers=[1,2,3,4,5,6,7,8,9]
def factorial(n):
    if n == 0:  # Base case: factorial of 0 is 1
        return 1
    else:
        return n * factorial(n - 1)  # Recursive case: n! = n * (n-1)!
# Call the factorial function with argument 5 and print the result
factorial_result = factorial(5)
print("Factorial of 5:", factorial_result)
```

```
Factorial of 5: 120
```

In [23]:
```
def factorial(n):
    try:
        if n < 0:
            raise ValueError("Factorial is not defined for negative numbers.")
        elif n == 0:  # Base case: factorial of 0 is 1
            return 1
        else:
            return n * factorial(n - 1)  # Recursive case: n! = n * (n-1)!
    except ZeroDivisionError:
        print("Error: Attempted to divide by zero.")

# Test the factorial function
try:
    result = factorial(0)
    print("Factorial of num:", result)

    # Attempt to divide by zero
    result = 10 / 0
except ZeroDivisionError:
    print("Error: Attempted to divide by zero.")
except ValueError as ve:
    print(ve)
```

```
Factorial of num: 1
Error: Attempted to divide by zero.
```

Q.5) Recursion, Exception Handling

In [3]:
```
def recursive_function(n):
    try:
        # Base case: n is 0
        if n == 0:
            return 1

        # Recursive case
        return n * recursive_function(n - 1)

    except ZeroDivisionError:
        # Handle ZeroDivisionError
        print("Error: Cannot divide by zero.")
        return None

# Test cases
print("recursive_function:",recursive_function(5))  # Output should be 120 (5!
print("recursive_function:",recursive_function(0))  # Output should be 1 (base
```

```
recursive_function: 120
recursive_function: 1
```

2. Create a Python program covering the following topics with concise examples:

1) list

In [ ]:
```
"""Storing and Accessing Data in Lists:
Initialize a list containing integers and demonstrate how to access elements us
Print the first and last elements of the list. """
```

In [ ]:
```
# list index
list1 = [2,3,4,5,6,7,8,9,10]
print("list1:", list1)
first_integer=list1[0]
print("first_integer : ",first_integer)
last_element=list1[8]
print("last_element : ",last_element)
```

In [ ]:
```
"""Modifying Lists:
Append a new element to the list and print the modified list. Then, update an e
print the list again."""
```

In [10]:
```
list1.append(1)
print(list1)
```

```
list1: [2, 3, 4, 5, 6, 7, 8, 9, 10]
first_integer :  2
last_element :  10
[2, 3, 4, 5, 6, 7, 8, 9, 10, 1]
```

In [ ]:
```
"""Operations on Lists:
Perform list concatenation with another list and print the result. Also, demons
by multiplying the list by a scalar and print the resulting list. """
```

In [11]:
```
list2=[11,12,13,14,15,16,17,18,19,20]
list1=[1,2,3,4,5,6,7,8,9,10]
new_list=(list1+list2)
print("new_list:",new_list)
scalar=2
scalar_list=[element * scalar for element in new_list]
print("scalar_list:",scalar_list)
```

```
new_list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20]
scalar_list: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34,
36, 38, 40]
```

In [ ]:
```
"""Aliasing, List Methods:
Create a new list and alias it to the original list. Modify the alias list and
original list. Additionally, demonstrate commonly used list methods such as app
remove(), pop(), and sort(). """
```

In [14]:
```
#ALIAS THE LIST
fruit=['Aapple','Banana','Grapes','Watermelon','Cherry']
print(fruit)
FRUIT=fruit
print(FRUIT)
FRUIT.append('Orange')  #append the orange
print("Appended item:",FRUIT)
# does not any changes in our original list
FRUIT.insert(4,'Pineapple')
print("Inserted item:",FRUIT)
FRUIT.remove('Cherry')
print("removed item:",FRUIT)
FRUIT.pop(3)
print("Poped item:",FRUIT)
FRUIT.sort()
print("sorted item:",FRUIT)
```

```
['Aapple', 'Banana', 'Grapes', 'Watermelon', 'Cherry']
['Aapple', 'Banana', 'Grapes', 'Watermelon', 'Cherry']
Appended item: ['Aapple', 'Banana', 'Grapes', 'Watermelon', 'Cherry', 'Orang
e']
Inserted item: ['Aapple', 'Banana', 'Grapes', 'Watermelon', 'Pineapple', 'Che
rry', 'Orange']
removed item: ['Aapple', 'Banana', 'Grapes', 'Watermelon', 'Pineapple', 'Oran
ge']
Poped item: ['Aapple', 'Banana', 'Grapes', 'Pineapple', 'Orange']
sorted item: ['Aapple', 'Banana', 'Grapes', 'Orange', 'Pineapple']
```

In [ ]:
```
#with a List of Lists:
Create a list of lists, each inner list representing a student's information (e
Access and print the information of a specific student from the list of lists.
```

In [7]:
```
specific_student_info = [['Mayuri', 22, 'A'], ['Saiely', 22, 'A'], ['Esha', 20,

# Iterating through each student in specific_student_info and printing their de
for student in specific_student_info:
    print("Name:", student[0])
    print("Age:", student[1])
    print("Grade:", student[2])
    print()  # Print a blank line for better readability between students
```

```
Name: Mayuri
Age: 22
Grade: A

Name: Saiely
Age: 22
Grade: A

Name: Esha
Age: 20
Grade: A
```

In [ ]:
```
"""Processing Lists Using Indices:
Iterate over the elements of the list using indices and perform some operation
such as doubling its value or converting it to uppercase. Print the modified li
```

In [61]:
```
fruit=['APPLE','BANANA','GRAPES','WATERMELON','CHERRY']
for i in fruit:
    print(i)
newlist=[i.lower() for i in fruit]
print(newlist)
```

```
APPLE
BANANA
GRAPES
WATERMELON
CHERRY
['apple', 'banana', 'grapes', 'watermelon', 'cherry']
```

2) set

In [ ]:
```
"""Storing Data Using Sets:
Initialize a set containing unique elements and demonstrate basic set operation
intersection, and difference with another set."""
```

In [15]:
```
set1 = {'rose','lily','watermelon','merrigold','jasmin','cherry'}
set2={'apple', 'banana', 'grapes', 'watermelon', 'cherry','juee'}
print("set1:",set1)
print("set2:", set2)
type(set1)
Sets=set2.intersection(set1)
print("intersected set:",Sets)
sets=set2.union(set1)
print("union set:",sets)
difference_with_set=set1.difference(set2)
print("difference_with_set:",difference_with_set)
```

```
set1: {'lily', 'jasmin', 'watermelon', 'rose', 'merrigold', 'cherry'}
set2: {'banana', 'apple', 'watermelon', 'grapes', 'juee', 'cherry'}
intersected set: {'watermelon', 'cherry'}
union set: {'banana', 'jasmin', 'grapes', 'lily', 'apple', 'watermelon', 'ros
e', 'juee', 'merrigold', 'cherry'}
difference_with_set: {'lily', 'jasmin', 'rose', 'merrigold'}
```

3) TUPLE

In [ ]:
```
"""Storing Data Using Tuples:
Declare a tuple containing heterogeneous data types (e.g., integer, float, stri
element individually. """
```

In [8]:
```
mixed_data=('Mayuri', 'Saiely',3,18,2, 22.0, 22.0)
type(mixed_data)
for element in mixed_data:
    print(element)
```

```
Mayuri
Saiely
3
18
2
22.0
22.0
```

4) DICTONARIES

In [ ]:
```
"""Storing Data Using Dictionaries:
Create a dictionary representing a student's information (e.g., name, age, grad
values. Access and print specific information from the dictionary."""
```

In [9]:
```
my_dict={'name':'Mayuri','age':22,'grades':'A'}
for key, value in my_dict.items():
    print(key, ":", value)
```

```
name : Mayuri
age : 22
grades : A
```

3. Develop a Python program covering the following topics with concise examples:

IMPORTING LIBRARIES

In [ ]:
```
"""Import the math, random, and datetime modules.
Demonstrate importing specific functions or classes from these modules and usir
program. """
```

In [20]:
```
import math
import random
import datetime
```

In [3]:
```
import datetime

x = datetime.datetime(2002,3,9)
print(x)
```

```
2002-03-09 00:00:00
```

Commonly Used Python Libraries

In [ ]:
```
"""Utilize the math module to perform mathematical
operations such as calculating square roots and trigonometric functions. Use th
to generate random numbers and select random elements from a list. Additionally
getting the current date and time using the datetime module."""
```

In [4]:
```
import math
a=64
squareroot=math.sqrt(a)
print(squareroot)
```

```
8.0
```

In [ ]:
```
"""Working with a List of Lists:
Create a list of lists, each inner list representing a student's information (e
Access and print the information of a specific student from the list of lists.
```

In [5]:
```
# Angle in radians
angle = math.pi / 4  # 45 degrees in radians

# Sine of the angle
sin_value = math.sin(angle)
print("Sine:", sin_value)
```

```
Sine: 0.7071067811865476
```

In [12]:
```
import random
alphanumeric=[1,2,3,4,5,6,7,8,9,0,'A','B','C','D','d','e','f','g','h','i','j']
random_list=[]
for i in alphanumeric:
    list_random=random.choice(alphanumeric)
    random_list.append(list_random)
print(random_list)
```

```
['g', 'A', 'A', 7, 7, 'j', 'i', 1, 'g', 'f', 'g', 9, 'D', 2, 1, 'h', 'd',
'i', 5, 'e', 1]
```

In [1]:
```
import datetime

x = datetime.datetime(2002,3,9)
print(x)
```

```
2002-03-09 00:00:00
```

introduction to third party libraries

In [ ]:
```
"""Briefly introduce the NumPy and Pandas libraries.
Import NumPy and create a NumPy array to store numeric data. Then, import Panda
demonstrate creating a DataFrame to organize tabular data. """
```

1) NUMPY

In [16]:
```
# import numpy and numpy array store numerical data

import numpy as np

# Create a NumPy array from a Python list
my_list = [1, 2, 3, 4, 5]
my_array = np.array(my_list)

# Print the array
print("NumPy Array:", my_array)
```

```
NumPy Array: [1 2 3 4 5]
```

2) Pandas

In [22]:
```
import pandas as pd
left = pd.DataFrame({
   'id':[1,2,3,4,5],
   'subject_id':['sub1','sub2','sub4','sub','sub5']})
right = pd.DataFrame(
   {'id':[1,2,3,4],
   'subject_marks':[40,30,50,30]})
right.head()
pd.merge(left,right,on="id",how="right")
```

Out[22]:

	id	subject_id	subject_marks
0	1	sub1	40
1	2	sub2	30
2	3	sub4	50
3	4	sub6	30

Reading and Writing Files:

In [ ]:
```
"""Open a file in# read mode, read its contents, and print them to the
console. Then, open a file in write mode, write some text to it, and close the
same file in append mode, add more text to it, and close the file. """
```

In [2]:
```
import pandas as pd
file=pd.read_csv(r"C:\Users\jades\Downloads\quikr_car.csv")
file
```

Out[2]:

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing XO eRLX Euro III	Hyundai	2007	80,000	45,000 kms	Petrol
1	Mahindra Jeep CL550 MDI	Mahindra	2006	4,25,000	40 kms	Diesel
2	Maruti Suzuki Alto 800 Vxi	Maruti	2018	Ask For Price	22,000 kms	Petrol
3	Hyundai Grand i10 Magna 1.2 Kappa VTVT	Hyundai	2014	3,25,000	28,000 kms	Petrol
4	Ford EcoSport Titanium 1.5L TDCi	Ford	2014	5,75,000	36,000 kms	Diesel
...	...	...	...	...	...	...
887	Ta	Tara	zest	3,10,000	NaN	NaN
888	Tata Zest XM Diesel	Tata	2018	2,60,000	27,000 kms	Diesel
889	Mahindra Quanto C8	Mahindra	2013	3,90,000	40,000 kms	Diesel
890	Honda Amaze 1.2 E i VTEC	Honda	2014	1,80,000	Petrol	NaN
891	Chevrolet Sail 1.2 LT ABS	Chevrolet	2014	1,60,000	Petrol	NaN

In [4]:
```
f = open("quikr_car.csv", "a")
f.write("Now the file has more content!")
f.close()

#open and read the file after the appending:
f = open("quikr_car.csv", "r")
print(f.read())
```

```
Now the file has more content!
```

In [5]:
```
f=open("quikr_car.csv", "a")
f.write("now the file has moer content!")
f.close
f=open("quikr_car.csv",'r')
print(f.read())
```

```
Now the file has more content!now the file has moer content!
```

In [31]:
```
try:
    f = open("myfile.txt", "x")  # Try to create a new file
    print("File created successfully:", f)
    f.close()  # Close the file after using it
except FileExistsError:
    print("File 'myfile.txt' already exists.")
```

```
File 'myfile.txt' already exists.
```

In [18]:
```
import csv

# Reading from the diabetes.csv file
with open('diabetes.csv', mode='r') as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)

# Writing to the diabetes.csv file
data_to_write = [
    ['Patient ID', 'Glucose Level', 'Blood Pressure', 'BMI'],
    [101, 120, 80, 25],
    [102, 140, 90, 30],
    [103, 160, 95, 28]
]

with open('diabetes.csv', mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerows(data_to_write)
print("Data has been written to diabetes.csv")
```

```
['now the file has moer content!now the file has moer content!']
Data has been written to diabetes.csv
```

csv and json processing

In [ ]:
```
"""CSV and JSON Data Processing: Read data from a CSV file using the csv module
to the console. Next, use the JSON module to load JSON data from a file, manipu
the modified data. """
```

import csv import json

Read data from CSV file and print it

def read_csv(filename): with open(filename, 'r', newline='') as file: reader = csv.reader(file) for row in reader: print(row)

Load JSON data from file, manipulate it, and print it

def manipulate_json(filename): with open(filename, 'r') as file: data = json.load(file) print("Original JSON data:") print(json.dumps(data, indent=4)) # Pretty-printing JSON data

```
        # Manipulate JSON data (For demonstration, we'll just print it aga
    in)
        print("\nManipulated JSON data:")
        print(json.dumps(data, indent=4))  # Pretty-printing JSON data
```

CSV file processing

csv_filename = 'data.csv' print("Data from CSV file:") read_csv(csv_filename)

JSON file processing

ison filename = 'data.ison' print("\nData from JSON file:") manipulate ison(ison filename)

Data Manipulation and Analysis:

In [ ]:
```
"""Utilize Pandas to perform basic data manipulation and analysis
tasks. Load a dataset into a DataFrame, filter rows based on conditions, calcul
statistics, and perform basic data transformations such as sorting and grouping
```

In [44]:
```
import pandas as pd

dataset=pd.read_csv(r"C:\Users\jades\Downloads\Titanic-Dataset.csv")
dataset

df = dataset
print(df.head())
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name  Gender   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

In [47]:
```
import pandas as pd

# Load dataset into a DataFrame
dataset=pd.read_csv(r"C:\Users\jades\Downloads\Titanic-Dataset.csv")


# Filter rows based on conditions (e.g., age greater than 30 and survived)
filtered_df = df[(df['Age'] > 30) & (df['Survived'] == 1)]

# Display the filtered DataFrame
print("\nFiltered DataFrame (Age > 30 and Survived):")
print(filtered_df.head())

# Calculate summary statistics for numerical columns
summary_stats = df.describe()

# Display the summary statistics
print("\nSummary Statistics:")
print(summary_stats)

# Sort DataFrame by a specific column (e.g., Age)
sorted_df = df.sort_values(by='Age', ascending=False)

# Display the sorted DataFrame
print("\nSorted DataFrame by Age:")
print(sorted_df.head())

# Grouping by a categorical variable (e.g., Gender) and calculating mean age
grouped_df = df.groupby('Gender')['Age'].mean()

# Display the grouped DataFrame
print("\nMean Age by Gender:")
print(grouped_df)
```

```
Filtered DataFrame (Age > 30 and Survived):
    PassengerId  Survived  Pclass  \
1             2         1       1
3             4         1       1
11           12         1       1
15           16         1       2
21           22         1       2

                                                 Name  Gender   Age  SibSp  \
1   Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
11                           Bonnell, Miss. Elizabeth  female  58.0      0
15                 Hewlett, Mrs. (Mary D Kingcome)   female  55.0      0
21                             Beesley, Mr. Lawrence    male  34.0      0

    Parch    Ticket     Fare Cabin Embarked
1       0  PC 17599  71.2833   C85        C
3       0    113803  53.1000  C123        S
11      0    113783  26.5500  C103        S
15      0    248706  16.0000   NaN        S
21      0    248698  13.0000   D56        S

Summary Statistics:
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200

Sorted DataFrame by Age:
     PassengerId  Survived  Pclass                                 Name  \
630          631         1       1  Barkworth, Mr. Algernon Henry Wilson
851          852         0       3                  Svensson, Mr. Johan
493          494         0       1               Artagaveytia, Mr. Ramon
96            97         0       1             Goldschmidt, Mr. George B
116          117         0       3                  Connors, Mr. Patrick

    Gender   Age  SibSp  Parch    Ticket     Fare Cabin Embarked
630   male  80.0      0      0     27042  30.0000   A23        S
851   male  74.0      0      0    347060   7.7750   NaN        S
493   male  71.0      0      0  PC 17609  49.5042   NaN        C
96    male  71.0      0      0  PC 17754  34.6542    A5        C
```

```
116    male  70.5      0      0    370369   7.7500   NaN        Q

Mean Age by Gender:
Gender
female    27.915709
male      30.726645
Name: Age, dtype: float64
```

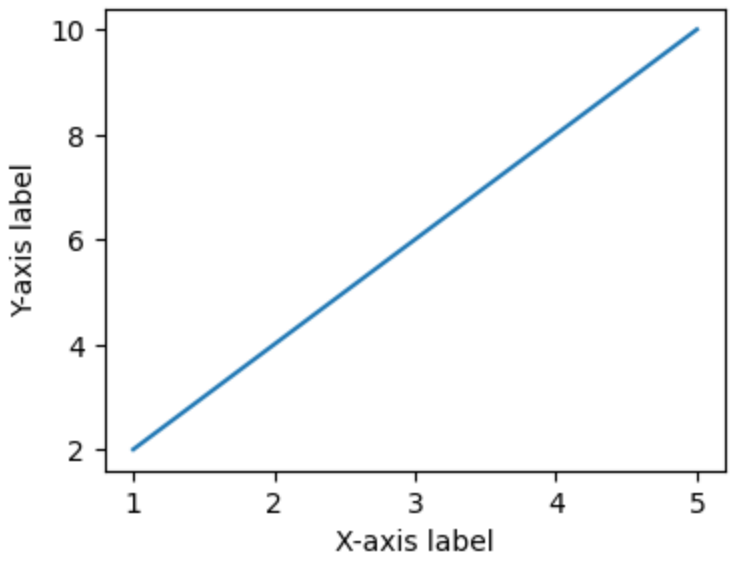
Data Visuallization

In [ ]:
```
 """Use the matplotlib or plotly library to create simple visualizations. Plot graph, histogram, and bar diagram, boxplot, etc. representing some sample data customize the plot with appropriate labels, titles, and legends. """
```

In [ ]:
```
"""USING MATPLOTLIB LIBRARY"""
```

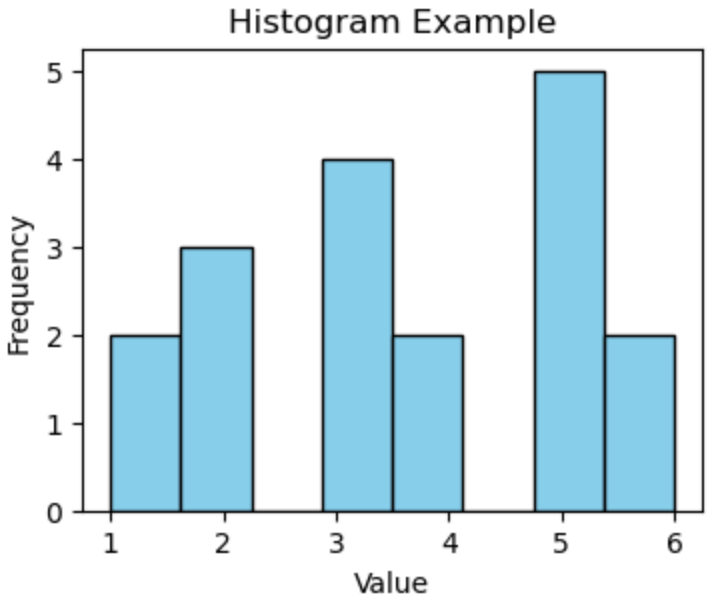
Line graph

In [10]:
```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
plt.figure(figsize=(4, 3))
plt.plot(x, y)
plt.xlabel('X-axis label')
plt.ylabel('Y-axis label')
plt.show()
```



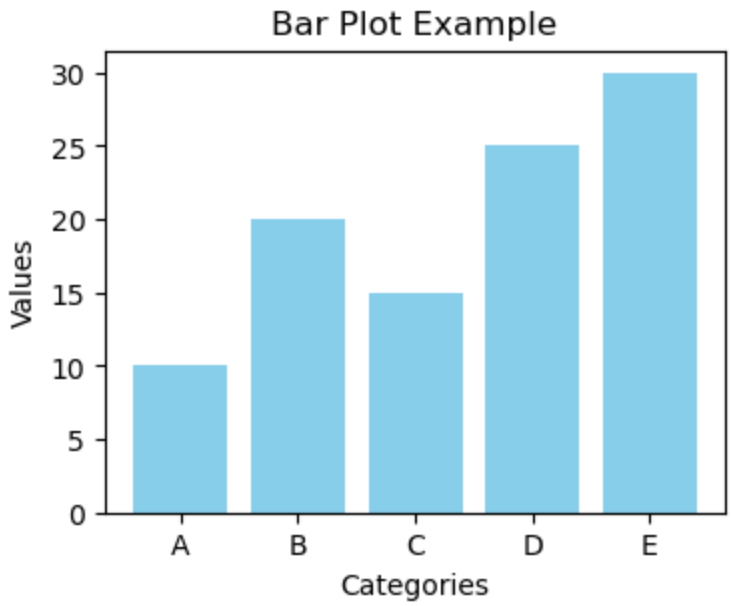
Histogram

In [17]:
```
import matplotlib.pyplot as plt
data = [1, 1, 2, 2, 2, 3, 3, 3, 3, 4, 4, 5, 5, 5, 5, 5, 6, 6]
plt.figure(figsize=(4, 3))
plt.hist(data, bins=8, color='skyblue', edgecolor='black')
plt.title('Histogram Example')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```



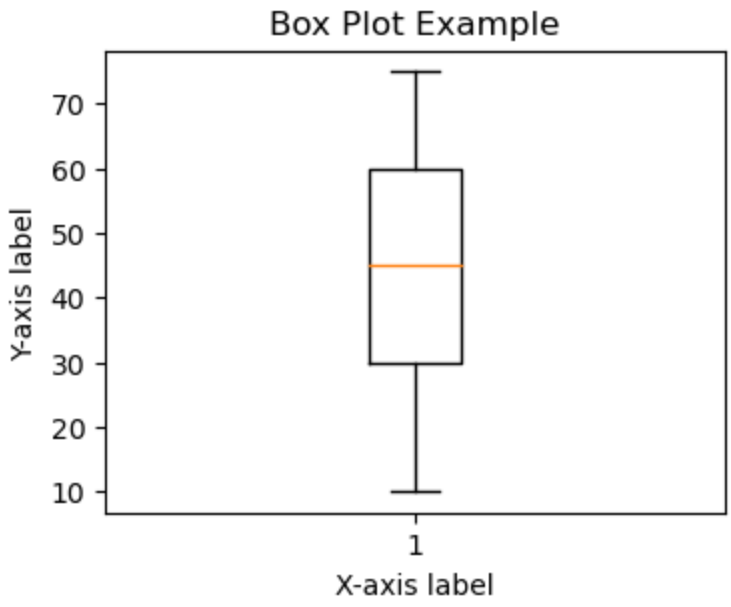
Bar Plot

In [16]:
```
import matplotlib.pyplot as plt
categories = ['A', 'B', 'C', 'D', 'E']
values = [10, 20, 15, 25, 30]
plt.figure(figsize=(4, 3))
plt.bar(categories, values, color='skyblue')
plt.title('Bar Plot Example')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.show()
```



Box Plot

In [19]:
```
import matplotlib.pyplot as plt
data = [10, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75]
plt.figure(figsize=(4, 3))
plt.boxplot(data)
plt.title('Box Plot Example')
plt.xlabel('X-axis label')
plt.ylabel('Y-axis label')
plt.show()
```



GUI interface

In [ ]:
```
"""Write a Python program to develop a GUI application for calculating insuranc
on user input and generating premium receipts"""
```

In [ ]:
```
"""1. Program Objective:
The goal of the GUI application is to provide users with a convenient way to ca
premiums based on their personal information and generate premium receipts. It
the needs of users by offering a user-friendly interface to input their details
premium calculations tailored to their specific factors."""
```

In [ ]:

```
import tkinter as tk
from tkinter import messagebox

# Function to calculate premium and generate receipt
def calculate_premium():
    # Get user input from entry fields
    name = name_entry.get()
    age = int(age_entry.get())
    gender = gender_var.get()
    smoking_status = smoking_var.get()
    phone_number = phone_entry.get()

    # Base premium amount
    base_premium = 1000

    # Calculate adjustment factors
    age_adjustment_factor = 1 - (age / 100)
    gender_adjustment_factor = 0.95 if gender == "Female" else 1
    smoking_adjustment_factor = 1.20 if smoking_status == "Smoker" else 1

    # Calculate final premium
    final_premium = base_premium * age_adjustment_factor * gender_adjustment_fa

    # Display premium receipt
    receipt = f"Policyholder's Name: {name}\nAge: {age}\nGender: {gender}\nSmol
    messagebox.showinfo("Premium Receipt", receipt)

# Create main window
root = tk.Tk()
root.title("Insurance Premium Calculator")

# Label and entry for Name
name_label = tk.Label(root, text="Name:")
name_label.grid(row=0, column=0)
name_entry = tk.Entry(root)
name_entry.grid(row=0, column=1)

# Label and entry for Age
age_label = tk.Label(root, text="Age:")
age_label.grid(row=1, column=0)
age_entry = tk.Entry(root)
age_entry.grid(row=1, column=1)

# Label and radio buttons for Gender
gender_label = tk.Label(root, text="Gender:")
gender_label.grid(row=2, column=0)
gender_var = tk.StringVar(value="Male")
male_radio = tk.Radiobutton(root, text="Male", variable=gender_var, value="Male
male_radio.grid(row=2, column=1, sticky="w")
female_radio = tk.Radiobutton(root, text="Female", variable=gender_var, value="
female_radio.grid(row=2, column=1, sticky="e")

# Label and radio buttons for Smoking Status
smoking_label = tk.Label(root, text="Smoking Status:")
smoking_label.grid(row=3, column=0)
smoking_var = tk.StringVar(value="Non-Smoker")
non_smoker_radio = tk.Radiobutton(root, text="Non-Smoker", variable=smoking_var
```

```
non_smoker_radio.grid(row=3, column=1, sticky="w")
smoker_radio = tk.Radiobutton(root, text="Smoker", variable=smoking_var, value=
smoker_radio.grid(row=3, column=1, sticky="e")

# Label and entry for Phone Number
phone_label = tk.Label(root, text="Phone Number:")
phone_label.grid(row=4, column=0)
phone_entry = tk.Entry(root)
phone_entry.grid(row=4, column=1)

# Button to calculate premium
calculate_button = tk.Button(root, text="Calculate Premium", command=calculate_
calculate_button.grid(row=5, columnspan=2)

root.mainloop()
```

In [ ]: