Veermata Jijabai Technological Institute

(An Autonomous Institute of Government of Maharashtra)

Department: Electronics and Telecommunication Engineering

(Data Science and Analysis Lab- R4ET3104P)

Experiment No.1

Aim: Introduction to Python Programming.

Name of Students: 1. Nandini Junghare (211091012)

2. Astha Shankar Shinde (211091044)

Year & Semester: Third year sixth semester

Branch: Electronics and Telecommunication

EXPERIMENT NO. 1

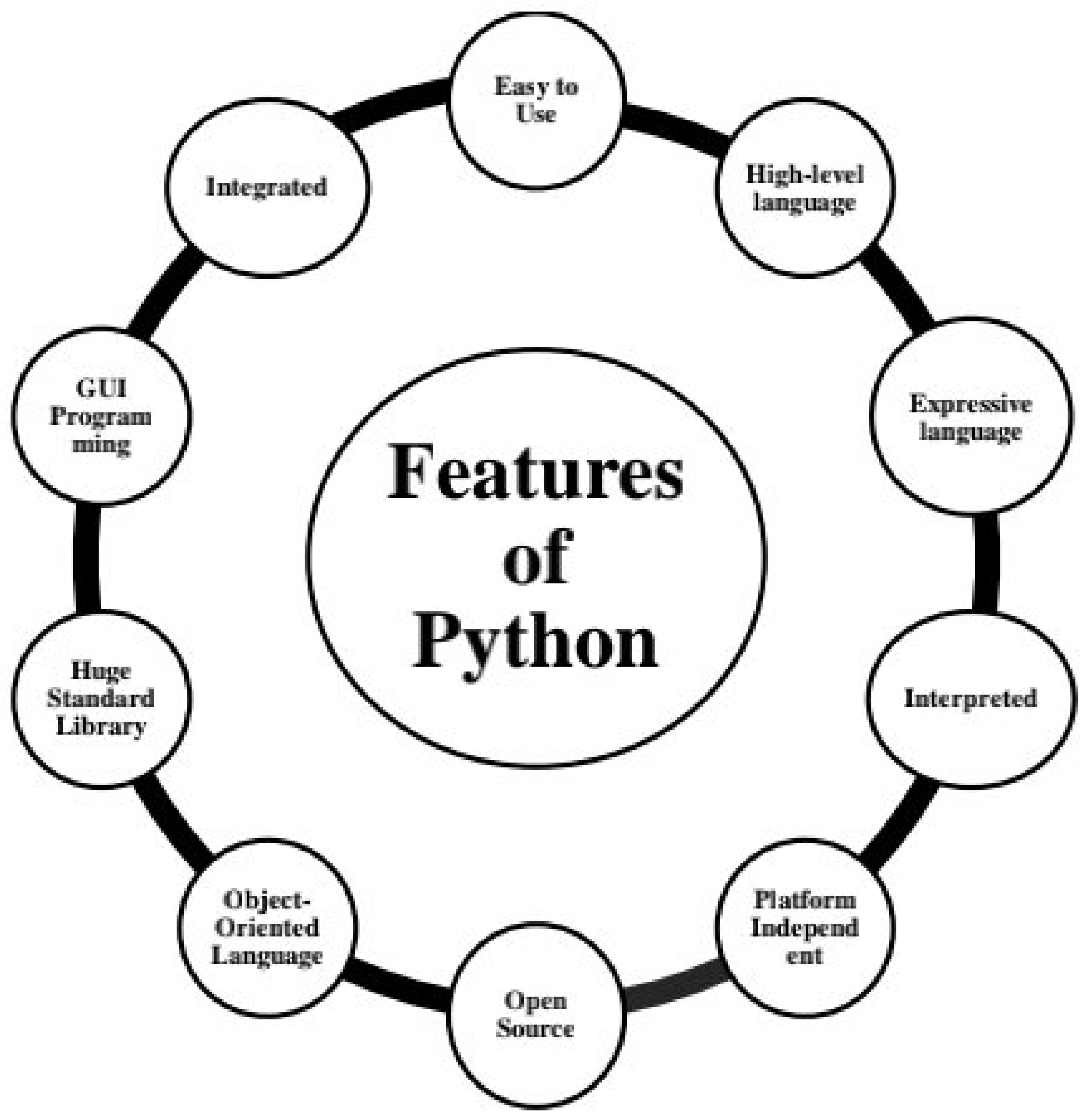
Aim: Introduction to Python Programming.

Objective: To learn Python programming language basics for data science and analysis.

Software used: Jupyter Notebook.

Theory: Python is a versatile programming language that is easy to learn and use. It was created by Guido van Rossum between 1985 and 1990. Python is known for its simplicity, readability, and powerful features.

Features of Python:



1. Easy to Use:
   Python has a simple and straightforward syntax, making it easy to learn and use, particularly for beginners.
2. High-Level Language:
   Python is a high-level language, abstracting away low-level details like memory management and providing a more user-friendly environment for programming.
3. Expressive Language:
   Python emphasizes readability and simplicity, allowing developers to express ideas in fewer lines of code compared to other programming languages.
4. Interpreted:
   Python is an interpreted language, which means that code is executed line by line, allowing for easier debugging and rapid development.
5. Platform Independent:
   Python code can run on various platforms and operating systems without modification, ensuring cross-platform compatibility.
6. Open Source:
   Python is open-source software, meaning its source code is freely available for modification and redistribution, fostering collaboration and innovation within the community.
7. Object-Oriented Language:
   Python supports object-oriented programming (OOP) principles, allowing developers to create reusable and modular code through classes and objects.
8. Huge Standard Library:
   Python comes with an extensive standard library, offering modules and packages for a wide range of tasks, reducing the need for external dependencies.

9. GUI Programming:

Python supports graphical user interface (GUI) programming through libraries like Tkinter, PyQt, and wxPython, enabling the development of desktop applications with interactive interfaces.

10. Integrated:

Python can be easily integrated with other languages and technologies, allowing for seamless interoperability and integration into existing systems and workflows.

11. Databases:

Python provides interfaces to all major commercial databases.

Python supports functional and structural programming methods as well as OOP. It can be used as a scripting language or can be compiled to byte-code for building large applications, it provides very high-level dynamic data types and supports dynamic type checking supporting automatic garbage collection. It can be easily integrated with C, C++, COM, ActiveX, COBRA and JAVA.

In Python, there are primarily four basic **data types**:
1. **Integers** (int): Represents whole numbers without decimal points, such as 5, -10, 100, etc.
2. **Floating-point numbers** (float): Represents real numbers with decimal points, such as 3.14, -0.5, 2.718, etc.
3. **Strings** (str): Represents sequences of characters enclosed in single (") or double ("") quotes, such as "hello", 'Python', "123", etc.
4. **Boolean** (bool): Represents the two truth values, True and False, used for logical operations and comparisons.

Python also provides several built-in **data structures** to store collections of data, including lists, tuples, sets, and dictionaries. These data structures can hold multiple values of various types.
1. **Lists**: Ordered, mutable collections enclosed in square brackets (`[]`). They can contain various data types. Example: `[1, 2, 3]`, `['apple', 'banana', 'cherry']`.
2. **Tuples**: Ordered, immutable collections enclosed in parentheses (`()`). They can contain various data types. Example: `(1, 2, 3)`, `('apple', 'banana', 'cherry')`.
3. **Sets**: Unordered, unique collections enclosed in curly braces (`{}`). They contain only unique elements and do not support duplicates. Example: `{1, 2, 3}`, `{'apple', 'banana', 'cherry'}`.
4. **Dictionaries**: Collections of key-value pairs enclosed in curly braces (`{}`). They are mutable and consist of unique keys and their corresponding values. Example: `{'name': 'John', 'age': 30, 'city': 'New York'}`.

Python Libraries:

Python libraries are collections of pre-written code that offer ready-to-use functions and tools for specific tasks. They help developers by providing shortcuts to perform common operations without needing to write code from scratch.

1. NumPy:

NumPy is a library for numerical computing in Python, providing support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

- Application: Used for scientific computing, data analysis, and numerical simulations.

2. Pandas:

Pandas is a library for data manipulation and analysis in Python, offering data structures and functions for easily handling structured data such as tables and time series.

- Application: Used for data cleaning, transformation, and analysis in data science projects and data-driven applications.

3. Matplotlib:

Matplotlib is a plotting library for creating static, interactive, and animated visualizations in Python, allowing users to generate various types of charts, plots, and graphs.

- Application: Used for data visualization to explore and communicate insights from data in a visually appealing manner.

4. Seaborn:

Seaborn is a statistical data visualization library in Python, built on top of Matplotlib, that provides high-level functions for creating informative and attractive statistical graphics.

- Application: Used for visualizing statistical relationships and patterns in data, particularly in exploratory data analysis and statistical modeling.

5. Scikit-learn:

Scikit-learn is a machine learning library in Python, offering simple and efficient tools for data mining and data analysis, including various algorithms for classification, regression, clustering, and dimensionality reduction.

- Application: Used for implementing machine learning models and performing tasks such as classification, regression, clustering, and model evaluation.

6. OS (Operating System Interface):

Helps interact with the operating system for tasks like file and directory operations, and executing system commands.

-Application: File manipulation, directory operations, and executing system commands.

7. Datetime:

Manipulates dates and times for tasks like parsing, formatting, arithmetic operations, and time zone handling.

-Application: Handling date and time data, including parsing, formatting, and arithmetic operations.

8. Statsmodels:

Estimates and analyzes statistical models for tasks like hypothesis testing, regression analysis, and time series analysis in fields such as economics and data science.

-Application: Statistical modeling and analysis, including hypothesis testing, regression analysis, and time series analysis.

In [3]:
```
# DSA Experiment no.1
# Nandini Junghare (211091012)
# Astha Shankar Shinde (211091044)

import numpy as np

b =np.empty(2,dtype=int)
print("Matrix b: \n",b)

a=np.empty([2,2],dtype=int)
print("Matrix a: \n",a)

c=np.empty([3,3])
print("\nMatrix c: \n",c)
```

```
Matrix b:
 [0 0]
Matrix a:
 [[-1616105815 -2066821942]
 [-1865321288   954055979]]

Matrix c:
 [[0.000e+000 0.000e+000 0.000e+000]
 [0.000e+000 0.000e+000 8.735e-321]
 [0.000e+000 0.000e+000 0.000e+000]]
```

In [6]:
```
f =np.zeros(2,dtype=int)
print("Matrix b: \n",f)
d=np.zeros([2,2],dtype=int)
print("Matrix d: \n",d)
e=np.zeros([3,3])
print("\nMatrix c: \n",e)
```

```
Matrix b:
 [0 0]
Matrix d:
 [[0 0]
 [0 0]]

Matrix c:
 [[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

In [7]:
```
d=np.zeros([2,5],dtype=int)
print("Matrix d: \n",d)
```

```
Matrix d:
 [[0 0 0 0 0]
 [0 0 0 0 0]]
```

In [11]:
```
#How many types of data ?.....HOMEWORK Search it.
import numpy as np
#Defining both the matrices
a= np.array([5,72,13,100])
b=np.array([2,3,4,5])

#Performing addition using arithematic operator
add_ans = a+b
print(add_ans)

#Same function can be use for multiplication
c=np.array([1,2,3,4])
add_ans = a+b+c
print(add_ans)

add_ans = np.add(a,b,c)
add_ans = np.add(a,b,c)
print(add_ans)
```

```
[  7  75  17 105]
[  8  77  20 109]
[  7  75  17 105]
```

In [12]:
```
import numpy as np
#Defining both the matrices
a= np.array([5,72,13,100])
b=np.array([2,3,4,5])
c=np.array([1,2,3,4])
add_ans = a+b+c
print(add_ans)
```

```
[  8  77  20 109]
```

In [13]:
```
add_ans = np.add(a,b,c)
print(add_ans)
```

```
[  7  75  17 105]
```

In [ ]:

```

```

In [15]:

```
import numpy as np
#Defining both the matrices
a= np.array([5,72,13,100])
b=np.array([2,3,4,5])

#Performing addition using arithematic operator
add_ans = a+b
print(add_ans)

#Same function can be use for multiplication
c=np.array([1,2,3,4])
add_ans = a+b+c
print(add_ans)

add_ans = np.add(a,np.add(b,c))

print(add_ans)
```

```
[  7  75  17 105]
[  8  77  20 109]
[  8  77  20 109]
```

In [21]:

```
#Indexing

#Python progran to demonstrate
#the use of index arrays
import numpy as np
#Create a sequence of integers from
#10 to 1 with step of -2
a=np.arange(10,1,-2)
print("\n As sequential array with a negative step: \n",a)
```

```

 As sequential array with a negative step:
 [10  8  6  4  2]
```

In [24]:

```
import numpy as np
a=np.arange(20)
print("\nArray is \n",a)
```

```

Array is
 [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
```

In [18]:

```
print("\n[-8:17:1] =",a[-8:17:1])
```

```

[-8:17:1] = [12 13 14 15 16]
```

In [19]:

```
print("\na[10:] =",a[10:])
```

```

a[10:] = [10 11 12 13 14 15 16 17 18 19]
```

In [22]:

```
a=np.arange(55)
print("\nArray is \n",a)

print("\n[0:49:1] =",a[0:50:1])
```

```

Array is
 [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54]

[0:49:1] = [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49]
```

In [25]:

```
import pandas as pd #IMport before importing numpy
import numpy as np
#Creating a series from a simple array
data =np.array(['a','b','c','d','e'])
ser =pd.Series(data)
print(ser)
```

```
0    a
1    b
2    c
3    d
4    e
dtype: object
```

In [ ]:

```

```

In [28]:

```
import pandas as pd
#Assume CSV file is in Jupyter home directory
csv_file_path ='bank-full.csv'
#Use pandas to read the CSV file into a Dataframe
df = pd.read_csv(csv_file_path)

#Printing top 5 Rows
df.head()
```

Out[28]:

	age;"job";"marital";"education";"default";"balance";"housing";"loan";"contact";"day";"month";"duration";"campaign";"pdays";"previous";"poutcom
0	58;"management";"married";"tertiary";"no";
1	44;"technician";"single";"secondary";"no'
2	33;"entrepreneur";"married";"secondary";"
3	47;"blue-collar";"married";"unknown";"no";
4	33;"unknown";"single";"unknown";"no";1;"r
5	35;"management";"married";"tertiary";"no"

In [29]:

	age;"job";"marital";"education";"default";"balance";"housing";"loan";"contact";"day";"month";"duration";"campaign";"pdays";"previous";"poutcom
0	58;"management";"married";"tertiary";"no";
1	44;"technician";"single";"secondary";"no'
2	33;"entrepreneur";"married";"secondary";"
3	47;"blue-collar";"married";"unknown";"no";
4	33;"unknown";"single";"unknown";"no";1;"r

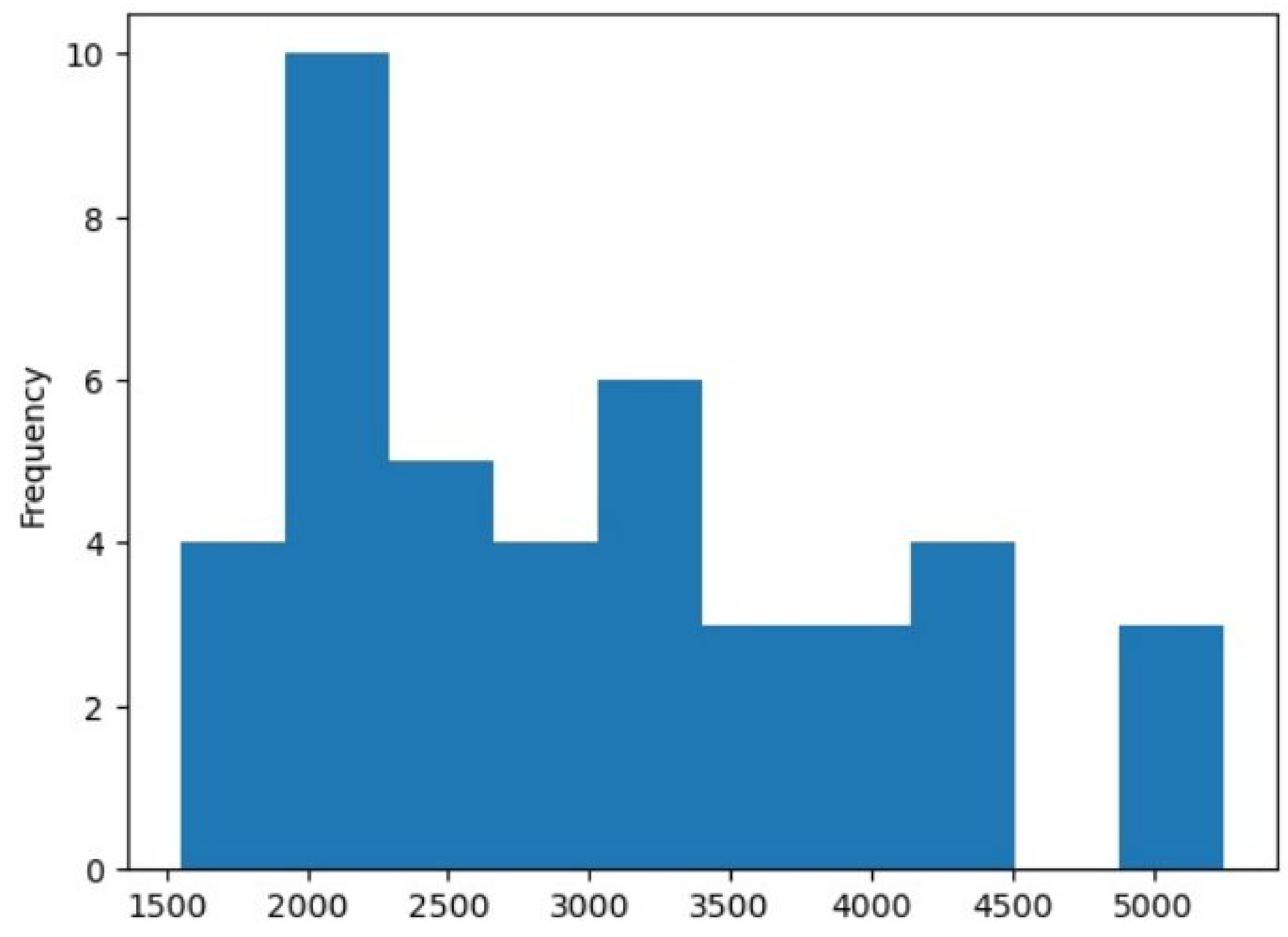
In [30]:
```
coca_cola=pd.read_excel("CocaCola_Sales_Rawdata.xlsx")
coca_cola.head()
```

Out[30]:

	Quarter	Sales
0	Q1_86	1734.827000
1	Q2_86	2244.960999
2	Q3_86	2533.804993
3	Q4_86	2154.962997
4	Q1_87	1547.818996

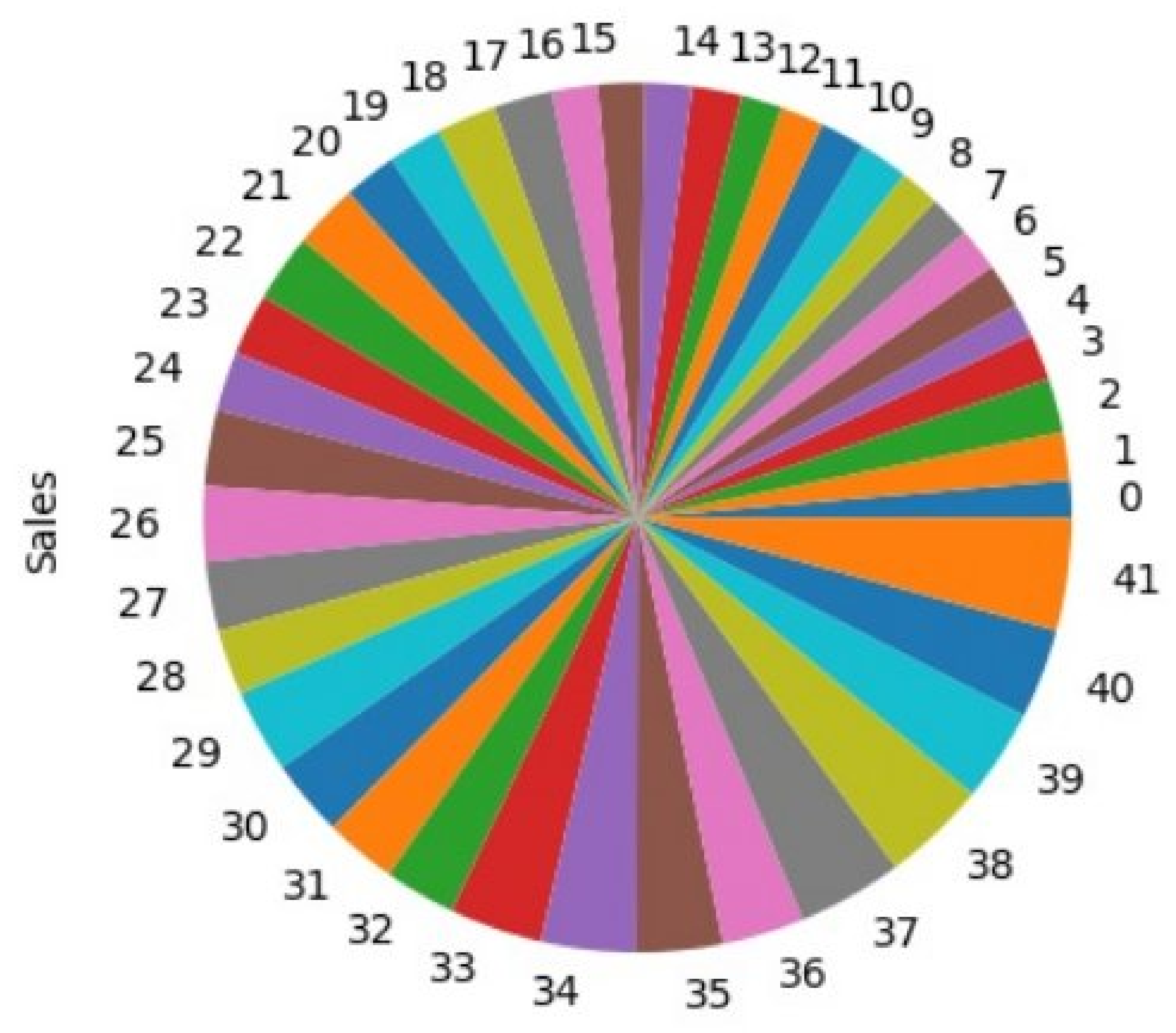
In [31]:
```
coca_cola.Sales.plot(kind ='hist')
```

Out[31]:
```
<AxesSubplot:ylabel='Frequency'>
```



In [32]:
```
coca_cola.Sales.plot(kind ='pie')
```

Out[32]:
```
<AxesSubplot:ylabel='Sales'>
```



In [33]:
```
 ataset1=coca_cola.rename({'Quarter': 'A','Sales':'B'},axis =1)
dataset1
```

Out[33]:

	A	B
0	Q1_86	1734.827000
1	Q2_86	2244.960999
2	Q3_86	2533.804993
3	Q4_86	2154.962997
4	Q1_87	1547.818996
...	...	...
37	Q2_95	4936.000000
38	Q3_95	4895.000000
39	Q4_95	4333.000000
40	Q1_96	4194.000000
41	Q2_96	5253.000000

42 rows × 2 columns

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Conclusion:

In summary, we've worked with matrices, doing math like adding, subtracting, and multiplying them. We've also created arrays in a sequence and explored important Python libraries like pandas, numpy, seaborn, and matplotlib, understanding why they're useful. Plus, we've learned how to read CSV files and show data in easy-to-understand charts like histograms and pie charts using pandas. This helps us analyze information better.