



Sinhgad Technical Education Society's

Sinhgad Institute of Management

**S. No. 44/1, Off Sinhgad Road, Vadgaon (Bk), Pune – 411
041**

Practical Journal On

Python Programming

A.Y. 2021-22

Class: MCA – I Semester- II

Index

Sr.no	Program
-------	---------

1.	<p>Write a python code to find given number is Armstrong Number or Not.</p> <p>Steps:</p> <ol style="list-style-type: none"> Accept Number from user in variable named as X. Print message whether number X is Armstrong or not. <p>Note: Armstrong number is a number that is equal to the sum of cubes of its digits. For example 153. ($1^3 + 5^3 + 3^3 = 153$)</p>
2	<p>Write a code in python to count number of vowels in given string</p> <p>Steps:</p> <ol style="list-style-type: none"> Accept string from user in variable named STR1. Count the number of vowels in STR1 and print. <p>Eg. 1.STR1 = 'COCONUT' => 3 2.STR1 = 'CONFIDence' => 4</p>
3	<p>Write a program, which will find all such numbers between 2000 and 3000 (both included) such that each digit of the number is an even number. eg. 2000, 2002...2888.</p>
4	<p>Write a program that accepts a sentence and calculate the number of letters and digits. Suppose the following input is supplied to the program: hello world! 123 Then, the output should be: ALPHABETS 10 DIGITS 3 (Note : Special symbols are not alphabets)</p>
5	<p>Write a Python function that takes a list and returns a new list with unique elements of the first list. Sample List : [1,2,3,3,3,3,4,5] Unique List : [1, 2, 3, 4, 5]</p>
6	<p>Write a Python program to make a chain of function decorators (bold, italic, underline etc.) in Python</p>

7	Write a Python program to generate a random alphabetical character, alphabetical string and alphabetical string of a fixed length. Use random.choice()
8	Write a generator function to print the table of the given number.
9	Write a python program to create University package which contains result modules. Also create sub-package College which contains exam module. Create test module and add getdata() and display() and access University and College package in it. Note: Assume suitable data.
10	Create a child class Bus that will inherit all of the variables and methods of the Vehicle class
11	Create a Bus class that inherits from the Vehicle class. Give the capacity argument of Bus.seating_capacity() a default value of 50.
12	Create a Bus child class that inherits from the Vehicle class. The default fare charge of any vehicle is seating capacity * 100. If Vehicle is Bus instance, we need to add an extra 10% on full fare as a maintenance charge. So total fare for bus instance will become the final amount = total fare + 10% of the total fare.
13	Write a Python class named Student with two attributes student_name, marks. Modify the attribute values of the said class and print the original and modified values of the said attributes.
14	Write a Python program to match a string that contains only upper and lowercase letters, numbers, and underscores.
15	Write a python program to validate the password by using regular expression.

	<p>a. Complexity requirement is that we need at least one capital letter, one number and one special character.</p> <p>b. We also need the length of the password to be between 8 and 18</p>
16	Write a python program to validate the URL by using regular expression.
17	Write a python program to validate an email address by using regular expression.
18	Write a python program which consists of - try, except, else, finally blocks.
19	Write a python program which raises the exception with a message.
20	Write a python program to accept age from user, raise user defined exception if age is below 18 years.
21	Write a Python multithreading program to print the thread name and corresponding process for each task (assume that there are four tasks).
22	Write a Python multithreading program which creates two threads, one for calculating the square of a given number and other for calculating the cube of a given number.
23	<p>Given a file called myfile.txt which contains the text: "Python is object oriented programming language".</p> <p>Write a program in Python that transforms the content of the file by writing each word in a separate line.</p>
24	Write a Python program that displays the longest word found in a text file.
25	Write a function in python that allows you to count the frequency of repetition of each word found in a given file.
26	Write a Python program which allows you to extract the content of a file from the 3rd line to the 7th line and save it in another file called extract_content.txt.

27	<p>Create the following DataFrame Sales containing year wise sales figures for five salespersons in INR. Use the years as column labels, and salesperson names as row labels.</p> <table><tr><td></td><td>2018</td><td>2019</td><td>2020</td><td>2021</td></tr><tr><td>Kapil</td><td>110</td><td>205</td><td>177</td><td>189</td></tr><tr><td>Kamini</td><td>130</td><td>165</td><td>175</td><td>190</td></tr><tr><td>Shikhar</td><td>115</td><td>206</td><td>157</td><td>179</td></tr><tr><td>Mohini</td><td>118</td><td>198</td><td>183</td><td>169</td></tr></table> <ol style="list-style-type: none">1. Create the DataFrame.2. Display the row labels of Sales.3. Display the column labels of Sales.4. Display the data types of each column of Sales.5. Display the dimensions, shape, size and values of Sales.		2018	2019	2020	2021	Kapil	110	205	177	189	Kamini	130	165	175	190	Shikhar	115	206	157	179	Mohini	118	198	183	169
	2018	2019	2020	2021																						
Kapil	110	205	177	189																						
Kamini	130	165	175	190																						
Shikhar	115	206	157	179																						
Mohini	118	198	183	169																						
28	<p>Plot the following data on a line chart and customize the chart according to the below-given instructions:</p> <table><tr><td>Month</td><td>January</td><td>February</td><td>March</td><td>April</td><td>May</td></tr><tr><td>Sales</td><td>510</td><td>350</td><td>475</td><td>580</td><td>600</td></tr></table> <p>Weekly Sales Report</p> <ol style="list-style-type: none">1. Write a title for the chart “The Monthly Sales Report”2. Write the appropriate titles of both the axes3. Write code to Display legends4. Display blue color for the line5. Use the line style – dashed6. Display diamond style markers on data points	Month	January	February	March	April	May	Sales	510	350	475	580	600													
Month	January	February	March	April	May																					
Sales	510	350	475	580	600																					
29	<p>Observe following data and plot data according to given instructions:</p> <table><tr><td>Batsman</td><td>2017</td><td>2018</td><td>2019</td><td>2020</td></tr><tr><td>Virat Kohli</td><td>2501</td><td>1855</td><td>2203</td><td>1223</td></tr></table>	Batsman	2017	2018	2019	2020	Virat Kohli	2501	1855	2203	1223															
Batsman	2017	2018	2019	2020																						
Virat Kohli	2501	1855	2203	1223																						

	<p>Steve Smith 2340 2250 2003 1153 Babar Azam 1750 2147 1896 1008 Rohit Sharma 1463 1985 1854 1638 Kane Williamson 1256 1785 1874 1974 Jos Butler 1125 1853 1769 1436</p> <ol style="list-style-type: none"> 1. Create a bar chart to display data of Virat Kohli & Rohit Sharma. 2. Customize the chart in this manner <ol style="list-style-type: none"> 2.1 . Use different widths 2.2. Use different colors to represent different years score 2.3. Display appropriate titles for axis and chart 2.4. Show legends 2.5. Create a bar chart to display data of Steve Smith, Kane Williamson & Jos Butler. Customize Chart as per your wish. 2.6. Display data of all players for the specific year.
30	Write a program to create a 3*3 numpy array with all the elements as per the user choice and print the sum of all elements of the array
31	Write a program to perform basic arithmetic operations on 1D and 2D array.
32	Write a Menu Driver Program to add, display, update, delete and exit in a student database containing Student_id, Student_name, Course through Python-MongoDB connectivity.
33	Demonstrate step by step MongoDB connection in Python
34	Write a Menu Driver Program to add, display, search, sort and exit in book database containing

	Book_id, Book_name, Book_author through Python-MongoDB connectivity
--	--

1]Write a python code to find given number is Armstrong Number or Not.

Steps:

- a. Accept Number from user in variable named as X.**
- b. Print message whether number X is Armstrong or not.**

Note: Armstrong

number is a number that is equal to the sum of cubes of its digits. For

example($1^3 + 5^3 + 3^3 = 153$)

Solution:

```
x= int(input("Enter a number: "))
```

```
sum = 0
```

```
temp = x
```

```
while temp > 0:
```

```
    rem = temp % 10
```

```
    sum += rem ** 3
```

```
    temp //= 10
```

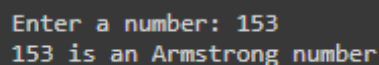
```
if x == sum:
```

```
    print(x,"is an Armstrong number")
```

```
else:
```

```
    print(x,"is not an Armstrong number")
```

Output:



```
Enter a number: 153
153 is an Armstrong number
```

2] Write a code in python to count number of vowels in given string

Steps: a. Accept string from user in variable named STR1.

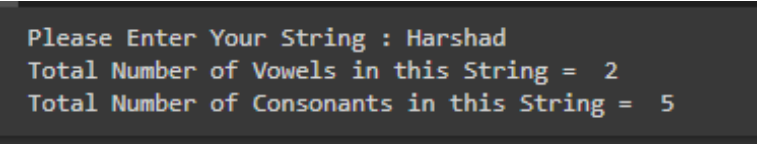
b. Count the number of vowels in STR1 and print.

Eg. 1.STR1 = 'COCONUT' => 3 2.STR1 = 'CONFIDence' => 4

Solution:

```
str1 = input("Please Enter Your String : ")
vowels = 0
consonants = 0
str1.lower()
for i in str1:
    if(i == 'a' or i == 'e' or i == 'i' or i == 'o' or i == 'u'):
        vowels = vowels + 1
    else:
        consonants = consonants + 1
print("Total Number of Vowels in this String = ", vowels)
print("Total Number of Consonants in this String = ",
consonants)
```

Output:

A screenshot of a terminal window showing the output of the program. The text is as follows:

```
Please Enter Your String : Harshad
Total Number of Vowels in this String =  2
Total Number of Consonants in this String =  5
```

Please Enter Your String : Harshad
Total Number of Vowels in this String = 2
Total Number of Consonants in this String = 5

3] Write a program, which will find all such numbers between 2000 and

3000 (both included) such that each digit of the number is an even

number. eg. 2000, 2002...2888.

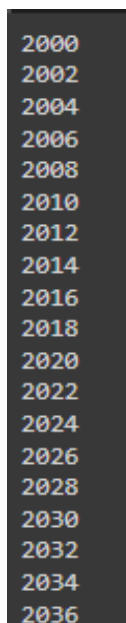
Solution:

```
for i in range(2000, 3001):
```

```
    if i%2==0:
```

```
        print(i)
```

output:



```
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
2024
2026
2028
2030
2032
2034
2036
```

4] Write a program that accepts a sentence and calculate the number of

letters and digits. Suppose the following input is supplied to the program:

hello world! 123 Then, the output should be: ALPHABETS 10 DIGITS 3

(Note : Special symbols are not alphabets)

Solution:

```
s = input("Input a string")
```

```
d=l=0
```

```
for c in s:
```

```
    if c.isdigit():
```

```
        d=d+1
```

```
    elif c.isalpha():
```

```
        l=l+1
```

```
    else:
```

```
        pass
```

```
print("Letters", l)
```

```
print("Digits", d)
```

output:

```
Input a stringhello world! 123
Letters 10
Digits 3
```

5] Write a Python function that takes a list and returns a new list with

unique elements of the first list.

Sample List : [1,2,3,3,3,3,4,5]

Unique List : [1, 2, 3, 4, 5]

solution:

```
def unique_list(l):  
    x = []  
    for a in l:  
        if a not in x:  
            x.append(a)  
    return x  
  
print(unique_list([1,2,3,3,3,3,4,5]))
```

output:

```
[1, 2, 3, 4, 5]
```

6] Write a Python program to make a chain of function decorators (bold, italic, underline etc.) in Python

```
def make_bold(fn):  
    def wrapped():  
        return "<b>" + fn() + "</b>"  
    return wrapped
```

```
def make_italic(fn):  
    def wrapped():  
        return "<i>" + fn() + "</i>"
```

```
    return wrapped
```

```
def make_underline(fn):
```

```
    def wrapped():
```

```
        return "<u>" + fn() + "</u>"
```

```
    return wrapped
```

```
@make_bold
```

```
@make_italic
```

```
@make_underline
```

```
def hello():
```

```
    return "hello world"
```

```
print(hello())
```

output:

```
<b><i><u>hello world</u></i></b>
```

**7]Write a Python program to generate a random
alphabetical character,**

**alphabetical string and alphabetical string of a fixed length.
Use**

random.choice()

Solution:

```
import random
```

```
import string
print("Generate a random alphabetical character:")
print(random.choice(string.ascii_letters))
print("\nGenerate a random alphabetical string:")
max_length = 255
str1 = ""
for i in range(random.randint(1, max_length)):
    str1 += random.choice(string.ascii_letters)
print(str1)
print("\nGenerate a random alphabetical string of a fixed
length:")
str1 = ""
for i in range(10):
    str1 += random.choice(string.ascii_letters)
print(str1)
```

Output:

```
Generate a random alphabetical character:  
M  
  
Generate a random alphabetical string:  
RYFpYlUwGFPEBkkHfIdPgHpRXXHvPjlAYXugjSYMtfODpgxSMBRHAjLNI9  
  
Generate a random alphabetical string of a fixed length:  
QawovtILhy
```

9]Create a child class Bus that will inherit all of the variables and methods

of the Vehicle class

Solution:

class Vehicle:

```
    def __init__(self, name, max_speed, mileage):  
        self.name = name  
        self.max_speed = max_speed  
        self.mileage = mileage
```

class Bus(Vehicle):

```
    pass
```

School_bus = Bus("School Volvo", 180, 12)

```
print("Vehicle Name:", School_bus.name, "Speed:",  
School_bus.max_speed, "Mileage:", School_bus.mileage)
```

Output:

```
Vehicle Name: School Volvo Speed: 180 Mileage: 12
```

**10] Create a Bus class that inherits from the Vehicle class.
Give the**

**capacity argument of Bus.seating_capacity() a default value
of 50.**

Solution:

class Vehicle:

```
    def __init__(self, name, max_speed, mileage):
```

```
        self.name = name
```

```
        self.max_speed = max_speed
```

```
        self.mileage = mileage
```

```
    def seating_capacity(self, capacity):
```

```
        return f"The seating capacity of a {self.name} is  
{capacity} passengers"
```



```
class bus(Vehicle):  
  
    def seating_capacity(self, capacity=50):  
        return f"The seating capacity of a {self.name} is  
{capacity} passengers"  
  
b= bus("school bus", 120, 2990)  
  
b.seating_capacity()
```

Output:

```
'The seating capacity of a school bus is 50 passengers'
```

11]Create a Bus child class that inherits from the Vehicle class. The default fare charge of any vehicle is seating capacity * 100. If Vehicle is Bus instance, we need to add an extra 10% on full fare as a maintenance charge. So total fare for bus instance will become the

final amount = totalfare + 10% of the total fare

Solution:

```
class Vehicle:
```

```
def __init__(self, name, mileage, capacity):
```

```
    self.name = name
```

```
    self.mileage = mileage
```

```
    self.capacity = capacity
```

```
def fare(self):
```

```
    return self.capacity * 100
```

```
class Bus(Vehicle):
```

```
    def fare(self):
```

```
        fare_bus = self.capacity * 100
```

```
        total_fare = fare_bus + (0.1 *fare_bus)
```

```
        return total_fare
```

```
School_bus = Bus("School Volvo", 12, 50)
```

```
print("Total Bus fare is:", School_bus.fare())
```

output:

```
Total Bus fare is: 5500.0
```

12]Write a Python class named Student with two attributes student_name, marks. Modify the attribute values of the said class and print the original and modified values of the said attributes.

Solution::

```
class Student:
```

```
    student_name = 'Harshad '
```

```
    marks = 93
```

```
print(f"Student Name: {getattr(Student, 'student_name')}")
```

```
print(f"Marks: {getattr(Student, 'marks')}")
```

```
setattr(Student, 'student_name', 'Siddhesh ')
```

```
setattr(Student, 'marks', 95)
```

```
print(f"Student Name: {getattr(Student, 'student_name')}")
```

```
print(f"Marks: {getattr(Student, 'marks')}")
```

Output:

```
Student Name: Harshad  
Marks: 93  
Student Name: Siddhesh  
Marks: 95
```

13]Write a Python program to match a string that contains only upper and lowercase letters, numbers, and underscores.

solution

```
import re
```

```
def text_match(text):
```

```
    patterns = '^[a-zA-Z0-9_]*$'
```

```
    if re.search(patterns, text):
```

```
        return 'Found a match in given string!'
```

```
    else:
```

```
        return ('Match not found in given string')
```

```
print(text_match("Python__1"))
```

```
print(text_match("important programs plotting with python  
class 12"))
```

Output:

```
Found a match in given string!  
Match not found in given string
```

14]Write a python program to validate the password by using regular expression.

a. Complexity requirement is that we need at least one capital letter, one number and one special character.

b. We also need the length of the password to be between 8 and 18.

Solution:

```
import re
p= input("Input your password")
x = True
while x:
    if (len(p)<8 or len(p)>18):
        break
    elif not re.search("[A-Z]",p):
        break
    elif not re.search("[a-z]",p):
        break
    elif not re.search("[$#@]",p):
        break
    elif re.search("\s",p):
        break
    else:
```

```
print("Valid Password")
```

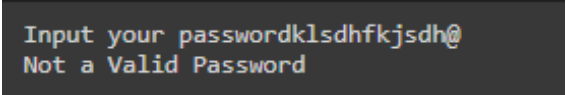
```
x=False
```

```
break
```

```
if x:
```

```
    print("Not a Valid Password")
```

output:

A terminal window with a dark background. The first line shows the prompt 'Input your password' followed by the input 'klsdhfkjsdh@'. The second line shows the output 'Not a Valid Password'.

```
Input your passwordklsdhfkjsdh@  
Not a Valid Password
```

15]Write a python program to validate the URL by using regular expression.

Solution:

```
import re
```

```
def isValidURL(str):
```

```
    regex = ("((http|https)://)(www.)?" +
```

```
            "[a-zA-Z0-9@:%._\\+~#?&//=]" +
```

```
"{2,256}\\.[a-z]" +  
"{2,6}\\b([-a-zA-Z0-9@:~#?&//=]*)"
```

```
p = re.compile(regex)
```

```
if (str == None):  
    return False
```

```
if(re.search(p, str)):  
    return True
```

```
else:  
    return False
```

```
url = "https://colab.research.google.com/"
```

```
if(isValidURL(url) == True):  
    print("Yes it is valid URL")  
else:  
    print("No it is not valid URL")
```

Output:

```
Yes it is valid URL
```

16]Write a python program to validate an email address by using regular expression

Solution:

```
import re
```

```
email_regex = re.compile(r"^[^@]+@[^@]+\.[^@]+")
```

```
if email_regex.match("foo@bar.com"):
```

```
    print("address is valid")
```

```
else:
```

```
    print("not valid")
```

Output:

```
address is valid
```

17]Write a python program which consists of - try, except, else, finally blocks

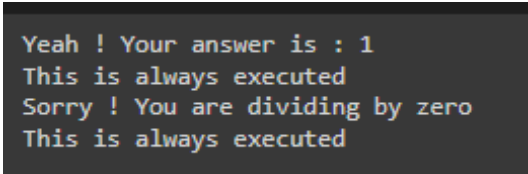
Solution:

```
def divide(x, y):
```



```
try:
    result = x // y
except ZeroDivisionError:
    print("Sorry ! You are dividing by zero ")
else:
    print("Yeah ! Your answer is :", result)
finally:
    print('This is always executed')
divide(3, 2)
divide(3, 0)
```

Output:



```
Yeah ! Your answer is : 1
This is always executed
Sorry ! You are dividing by zero
This is always executed
```

18]Write a python program which raises the exception with a message.

Solution:

```
x = -1
if x < 0:
    raise Exception("Sorry, no numbers below zero")
```

Output:

```
File "<ipython-input-43-ce7b10f402ef>", line 5
except ValueError:
    ^
```

19]Write a Python multithreading program to print the thread name and corresponding process for each task (assume that there are four tasks).

Solution:

```
import threading
```

```
import os
```

```
def task1():
```

```
    print("Task 1 assigned to thread:
    {}".format(threading.current_thread().name))
```

```
    print("ID of process running task 1: {}".format(os.getpid()))
```

```
def task2():
```

```
    print("Task 2 assigned to thread:
    {}".format(threading.current_thread().name))
```

```
    print("ID of process running task 2: {}".format(os.getpid()))
```

```
def task3():
```

```
    print("Task 3 assigned to thread:
    {}".format(threading.current_thread().name))
```

```
print("ID of process running task 3: {}".format(os.getpid()))
```

```
def task4():
```

```
    print("Task 4 assigned to thread:  
    {}".format(threading.current_thread().name))
```

```
    print("ID of process running task 4: {}".format(os.getpid()))
```

```
if __name__ == "__main__":
```

```
    print("ID of process running main program:  
    {}".format(os.getpid()))
```

```
    print("Main thread name:  
    {}".format(threading.current_thread().name))
```

```
t1 = threading.Thread(target=task1, name='t1')
```

```
t2 = threading.Thread(target=task2, name='t2')
```

```
t3 = threading.Thread(target=task3, name='t3')
```

```
t4 = threading.Thread(target=task4, name='t4')
```

```
t1.start()
```

```
t2.start()
```

t3.start()

t4.start()

t1.join()

t2.join()

t3.join()

t4.join()

Output:

```
ID of process running main program: 64
Main thread name: MainThread
Task 1 assigned to thread: t1Task 2 assigned to thread: t2
ID of process running task 2: 64

ID of process running task 1: 64
Task 3 assigned to thread: t3
ID of process running task 3: 64
Task 4 assigned to thread: t4
ID of process running task 4: 64
```

20] Write a Python multithreading program which creates two threads,

one for calculating the square of a given number and other for calculating

the cube of a given number.

Solution:

```
import threading
```

```
def print_cube(num):
```

```
    """
```

```
    function to print cube of given num
```

```
    """
```

```
    print("Cube: {}".format(num * num * num))
```

```
def print_square(num):
```

```
    """
```

```
    function to print square of given num
```

```
    """
```

```
    print("Square: {}".format(num * num))
```

```
if __name__ == "__main__":
```

```
    t1 = threading.Thread(target=print_square, args=(12,))
```

```
    t2 = threading.Thread(target=print_cube, args=(13,))
```

```
    t1.start()
```

```
    t2.start()
```

```
    t1.join()
```

```
    t2.join()
```

```
    print("Done!")
```

Output:

```
Square: 144  
Cube: 2197  
Done!
```

**21] Given a file called myfile.txt which contains the text:
“Python is object
oriented programming language”. Write a program in
Python that
transforms the content of the file by writing each word in a
separate line.**

Solution:

```
f = open('/content/drive/MyDrive/Colab Notebooks/myfile' ,  
'w')
```

```
f.write("Python\n is\n object\n oriented\n programming\n language")
```

```
f.close()
```

Output:

```
1 Python  
2 is  
3 object  
4 oriented  
5 programming  
6 language
```

**22] Write a Python program that displays the longest word
found in a text**

file

Solution:

```
def longest_word(filename):  
    with open('/content/drive/MyDrive/Colab  
Notebooks/myfile', 'r') as infile:  
        words = infile.read().split()  
        max_len = len(max(words, key=len))  
        return [word for word in words if len(word) == max_len]  
  
print(longest_word('/content/drive/MyDrive/Colab  
Notebooks/myfile'))
```

Output:

```
['programming']
```

23] Write a function in python that allows you to count the frequency of

repetition of each word found in a given file

Solution:

```
text = open("/content/drive/MyDrive/Colab  
Notebooks/myfile", "r")  
  
d = dict()  
  
for line in text:  
    line = line.strip()
```

```

line = line.lower()

words = line.split(" ")

for word in words:

    if word in d:

        d[word] = d[word] + 1

    else:

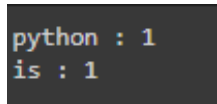
        d[word] = 1

for key in list(d.keys()):

    print(key, ":", d[key])

```

Output:



```

python : 1
is : 1

```

24 Write a Python program which allows you to extract the content of a

file from the 3rd line to the 7th line and save it in another file called

extract_content.txt.

Solution:

```

f = open("/content/drive/MyDrive/Colab Notebooks/myfile",
'w')

```

```

line="line1\nline2\nline3\nline4\nline5\nline6\nline7"

```



```
f.writelines(line)

print("writting suceed")

f.close()

f = open("/content/drive/MyDrive/Colab Notebooks/myfile",
'r')

content = f.readlines()

f2=open("/content/drive/MyDrive/Colab
Notebooks/extract_content", 'w')

f2.writelines(content[3:7])

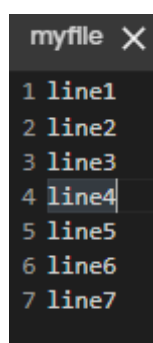
f.close()

f2.close()

print("text shifted sucessfully")
```

Output:

Content of myfile



```
myfile X
1 line1
2 line2
3 line3
4 line4
5 line5
6 line6
7 line7
```

Content after extracting text to extract_content.txt

```
extract_content X
1 line4
2 line5
3 line6
4 line7
```

25] Create the following DataFrame Sales containing year wise sales figures for five salespersons in INR. Use the years as column labels, and salesperson names as row labels.

	2018	2019	2020	2021
Kapil	110	205	177	189
Kamini	130	165	175	190
Shikhar	115	206	157	179
Mohini	118	198	183	169

- 1. Create the DataFrame.**
- 2. Display the row labels of Sales.**
- 3. Display the column labels of Sales.**
- 4. Display the data types of each column of Sales**

Solution:

```
import pandas as pd
```

```
d = {2018:[110,130,115,118],
```

```
2019:[205,165,175,190],
2020:[115,206,157,179],
2021:[118,198,183,169]}
sales=pd.DataFrame(d,index=['Kapil','Kamini','Shikhar','Mohini'])
```

```
print("Row Lables:\n",sales.index)
print("~~~~~\n")
print("Column Lables:\n",sales.columns)
print("~~~~~\n")
print("\nDisplay column data types")
print("~~~~~")
print(sales.dtypes)
print("\nDisplay the dimensions, shape, size and values of Sales")
print("~~~~~")
print("Dimensions:",sales.ndim)
print("Shape:",sales.shape)
print("Size:",sales.size)
print("Values:",sales.values)
```

```
Row Lables:  
Index(['Kapil', 'Kamini', 'Shikhar', 'Mohini'], dtype='object')  
~~~~~
```

```
Column Lables:  
Int64Index([2018, 2019, 2020, 2021], dtype='int64')  
~~~~~
```

```
Display column data types  
~~~~~
```

```
2018    int64  
2019    int64  
2020    int64  
2021    int64  
dtype: object
```

```
Display the dimensions, shape, size and values of Sales  
~~~~~
```

```
Dimensions: 2  
Shape: (4, 4)  
Size: 16  
Values: [[110 205 115 118]  
         [130 165 206 198]  
         [115 175 157 183]  
         [118 190 179 169]]
```

26]Plot the following data on a line chart and customize the chart according

to the below-given instructions:

Month January February March April May

Sales 510 350 475 580 600

Weekly Sales Report

- 1. Write a title for the chart “The Monthly Sales Report”**
- 2. Write the appropriate titles of both the axes**
- 3. Write code to Display legends**
- 4. Display blue color for the line**
- 5. Use the line style – dashed**
- 6. Display diamond style markers on data points**

Solution:

```
import matplotlib.pyplot as pp
mon=['January','February','March','April','May']
sales = [510,350,475,580,600]

pp.plot(mon,sales,label='Sales',color='b',linestyle='dashed',marker='D')

pp.title("The Monthly Sales Report")
pp.xlabel("Months")
pp.ylabel("Sales")
pp.legend()
```

pp.show()

Output:



27] Observe following data and plot data according to given instructions:

Batsman	2017	2018	2019	2020
Virat Kohli	2501	1855	2203	1223
Steve Smith	2340	2250	2003	1153
Babar Azam	1750	2147	1896	1008

Rohit Sharma 1463 1985 1854 1638

Kane Williamson 1256 1785 1874 1974

Jos Butler 1125 1853 1769 1436

1. Create a bar chart to display data of Virat Kohli & Rohit Sharma.

2. Customize the chart in this manner

1. Use different widths

2. Use different colors to represent different years score

3. Display appropriate titles for axis and chart

4. Show legends

5. Create a bar chart to display data of Steve Smith, Kane Williamson & Jos Butler.

Customize Chart as per your wish.

Solution:

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
barWidth = 0.15
```

```
fig = plt.subplots(figsize =(12, 8))
```

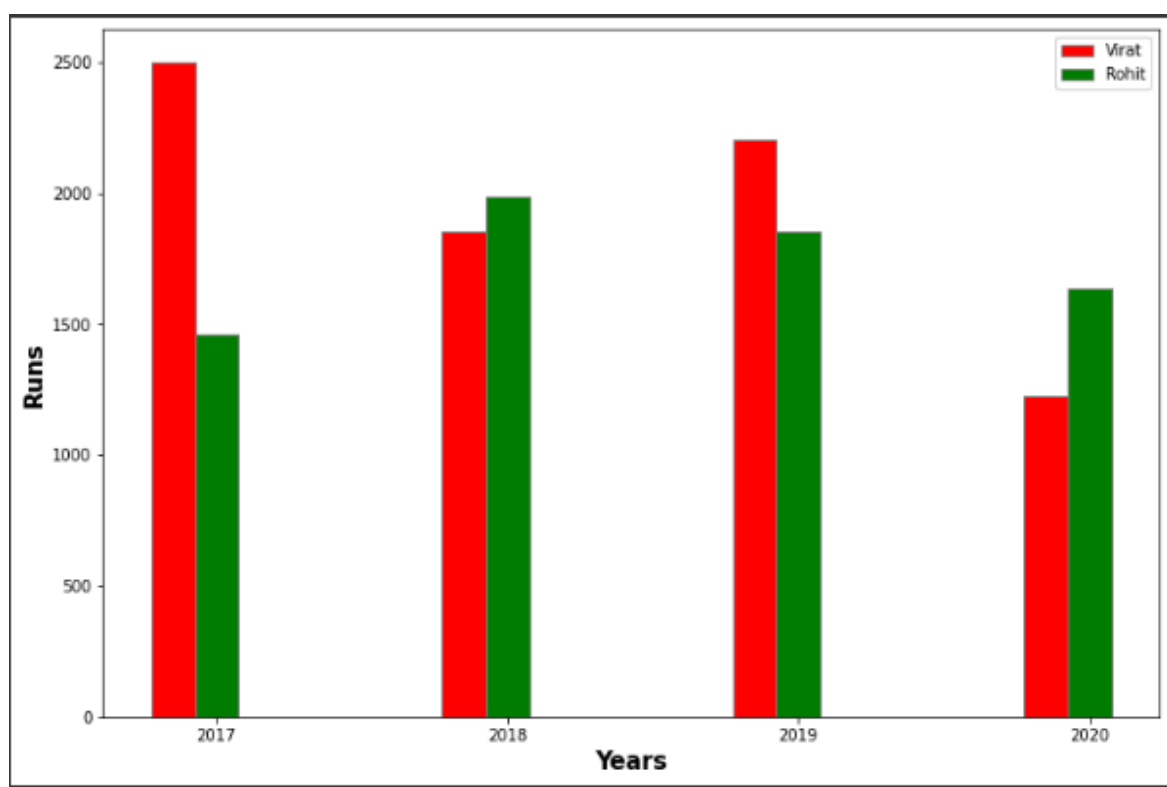
```
Virat = [2501,1855,2203,1223]
```

```
Rohit = [1463,1985,1854,1638]
```

```
br1 = np.arange(len(Rohit))
```

```
br2 = [x + barWidth for x in br1]
plt.bar(br1, Virat, color = 'r', width = barWidth,
        edgecolor = 'grey', label = 'Virat')
plt.bar(br2, Rohit, color = 'g', width = barWidth,
        edgecolor = 'grey', label = 'Rohit')
plt.xlabel('Years', fontweight = 'bold', fontsize = 15)
plt.ylabel('Runs', fontweight = 'bold', fontsize = 15)
plt.xticks([r + barWidth for r in range(len(Rohit))],
           [ '2017', '2018', '2019', '2020'])
plt.legend()
plt.show()
```

Output:



```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
barWidth = 0.2
```

```

fig = plt.subplots(figsize =(10, 6))
Kane = [1256,1785,1874,1974]
Smith = [2340,2250,2003,1153]
Babar =[1750 ,2147 ,1896 ,1008]
Jos= [1125 ,1853, 1769 ,1436]
br1 = np.arange(len(Smith))
br2 = [x + barWidth for x in br1]
br3 = [x + barWidth for x in br2]
br4 = [x + barWidth for x in br3]
plt.bar(br1, Kane, color ='b', width = barWidth,
        edgecolor ='blue', label ='kane')
plt.bar(br2,Smith , color ='g', width = barWidth,
        edgecolor ='orange', label ='Smith')
plt.bar(br3, Kane, color ='r', width = barWidth,
        edgecolor ='black', label ='Babar')
plt.bar(br4,Smith , color ='y', width = barWidth,
        edgecolor ='yellow', label ='Jos')

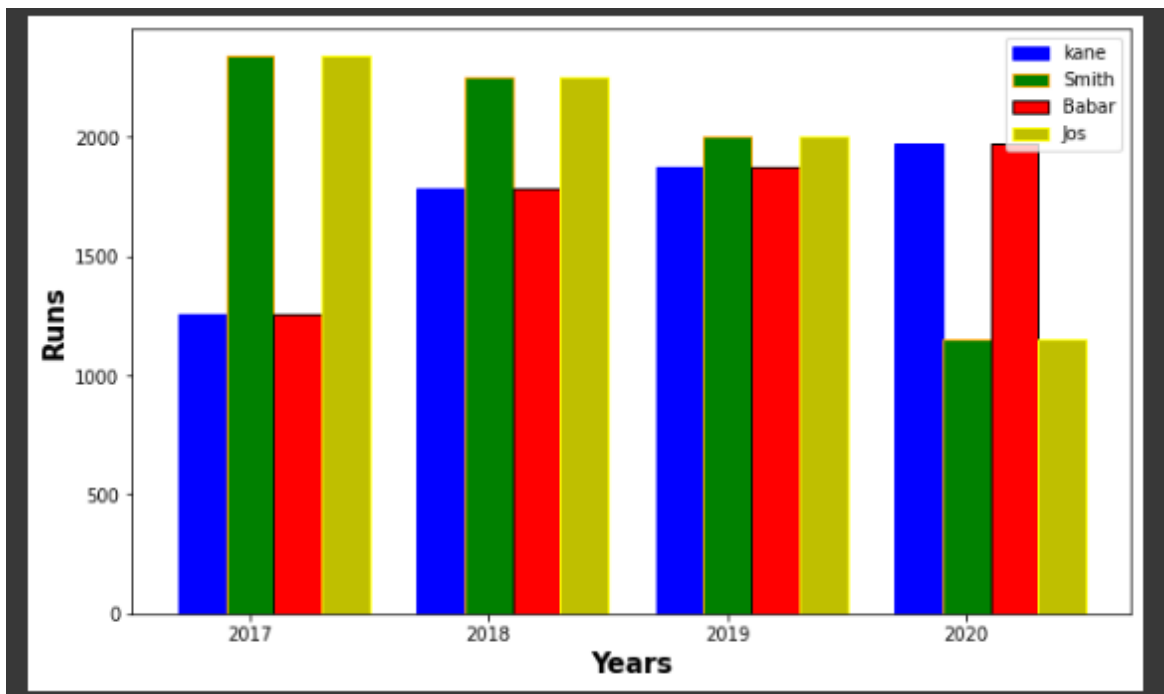
plt.xlabel('Years', fontweight ='bold', fontsize = 15)
plt.ylabel('Runs', fontweight ='bold', fontsize = 15)
plt.xticks([r + barWidth for r in range(len(Babar))],
           [ '2017', '2018', '2019','2020'])

```

```
plt.legend()
```

```
plt.show()
```

Output:



```
import numpy as np
```

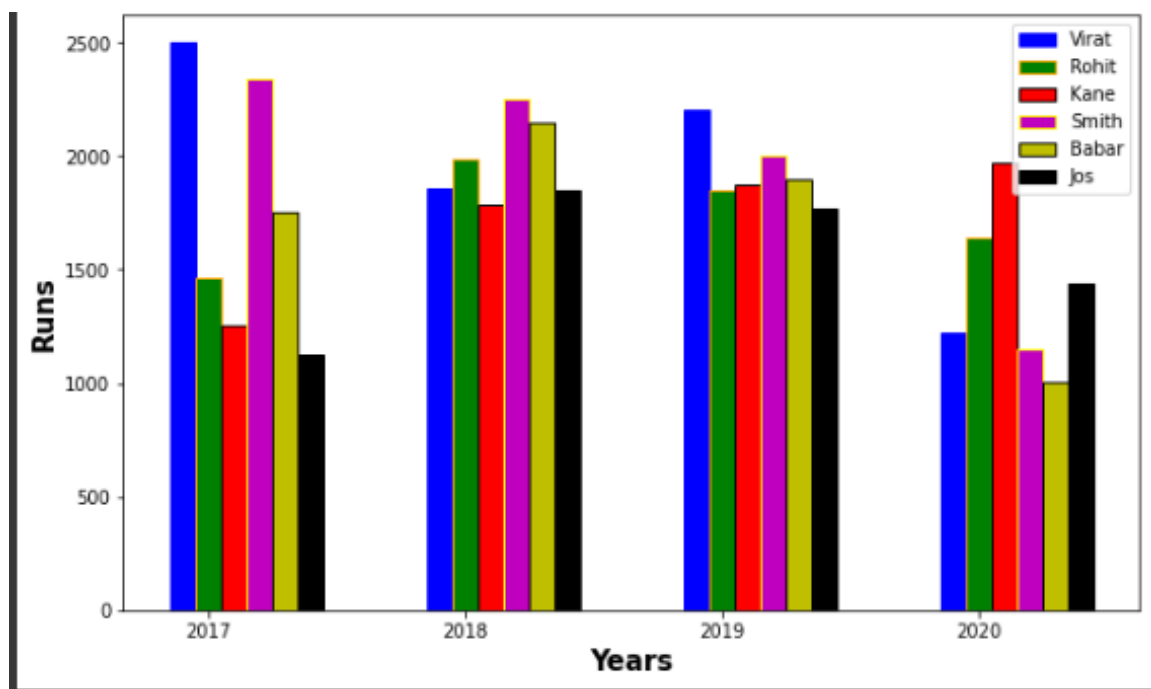
```
import matplotlib.pyplot as plt
```

```
barWidth = 0.1
fig = plt.subplots(figsize =(10, 6))
Virat = [2501,1855,2203,1223]
Rohit = [1463,1985,1854,1638]
Kane = [1256,1785,1874,1974]
Smith = [2340,2250,2003,1153]
Babar =[1750 ,2147 ,1896 ,1008]
Jos= [1125 ,1853, 1769 ,1436]
br1 = np.arange(len(Virat))
br2 = [x + barWidth for x in br1]
br3 = [x + barWidth for x in br2]
br4 = [x + barWidth for x in br3]
br5 = [x + barWidth for x in br4]
br6 = [x + barWidth for x in br5]

plt.bar(br1, Virat, color ='b', width = barWidth,
        edgecolor ='blue', label ='Virat')
plt.bar(br2,Rohit, color ='g', width = barWidth,
        edgecolor ='orange', label ='Rohit')
plt.bar(br3, Kane, color ='r', width = barWidth,
        edgecolor ='black', label ='Kane')
plt.bar(br4,Smith , color ='m', width = barWidth,
```

```
    edgecolor='yellow', label='Smith')
plt.bar(br5,Babar, color='y', width = barWidth,
        edgecolor='black', label='Babar')
plt.bar(br6,Jos, color='black', width = barWidth,
        edgecolor='black', label='Jos')

plt.xlabel('Years', fontweight='bold', fontsize = 15)
plt.ylabel('Runs', fontweight='bold', fontsize = 15)
plt.xticks([r + barWidth for r in range(len(Smith))],
           [ '2017', '2018', '2019','2020'])
plt.legend()
plt.show()
```



28] WAP to create a 3*3 numpy array with all the elements as per the

user choice and print the sum of all elements of the array

Solution:

```
import numpy as np
```

```
lst = []

for i in range(0, 9):
    if(i<9):
        ele = int(input(f"Enter:{i}, position Elements"))
        lst.append(ele)
```

```
arr = np.asarray(lst).reshape(3,3)
print(arr)
print(np.sum(arr))
```

Output:

```
Enter:0, position Elements1
Enter:1, position Elements2
Enter:2, position Elements3
Enter:3, position Elements4
Enter:4, position Elements4
Enter:5, position Elements5
Enter:6, position Elements5
Enter:7, position Elements6
Enter:8, position Elements6
[[1 2 3]
 [4 4 5]
 [5 6 6]]
36
33
```

29] WAP to perform basic arithmetic operations on 1D and 2D array .

solution:

```
import numpy as np
```

```
a = np.arange(4, dtype = np.float_).reshape(2,2)
print(a)
b = np.array([10,5])
print(b)
print('\n')
print(np.add(a,b))
print('\n')
print(np.subtract(a,b))
print('\n')
print(np.multiply(a,b))
print('\n')
print(np.divide(a,b))
```

Output:

```
[[0.  1.]
 [2.  3.]]
[10  5]

[[10.  6.]
 [12.  8.]]

[[-10.  -4.]
 [ -8.  -2.]]

[[ 0.  5.]
 [20. 15.]]

[[0.  0.2]
 [0.2 0.6]]
```


30]Write a Menu Driver Program to add, display, update, delete and exit

**in a student database containing
Student_id,Student_name,Course
through Python-MongoDB connectivity.**

```
from pymongo import MongoClient
```

```
server=MongoClient('localhost:27017')
```

```
db=server.student
```

```
def main():
```

```
    while True:
```

```
        selection = input('\nSelect\n 1.insert\n 2.read\n 3.update\n 4.delete\n 5.exit\n')
```

```
        if selection == '1':
```

```
            insert()
```

```
        elif selection == '2':
```

```
            read()
```

```
        elif selection == '3':
```

```
            update()
```

```
        elif selection == '4':
```

```
            delete()
```

```
        elif selection == '5':
```

```
            exit()
```

```
        break
```

```
else:
```

```
    print('\n INVALID SELECTION \n')
```

```
    break
```

```
def insert():
```

```
    try:
```

```
        student_id = input('Enter student id :')
```

```
        student_Name = input('Enter student Name :')
```

```
        course = input('enter name of course :')
```

```
        db.student.insert_one(
```

```
        {
```

```
        "id": student_id,
```

```
        "name": student_Name,
```

```
        "course": course
```

```
        })
```

```
        print ('\nInserted data successfully\n')
```

```
    except Exception:
```

```
        print("error while inserting")
```

```
def read():
```

```
    try:
```

```
        empCol = db.student.find()
```

```
        print ('\n All data from Database \n')
```

```

    for data in empCol:
        print(data)
except Exception:
    print("error while reading")
def update():
    try:
        student_id = input('\nEnter id to update\n')
        student_name = input('\nEnter name to update\n')
        course = input('\nEnter course to update\n')
        db.student.update_one(
            {"id": student_id},
            {
                "$set": {
                    "name": student_name,
                    "course": course
                }
            }
        )
        print ("\nRecords updated successfully\n")
    except Exception:
        print("error while updating")
def delete():

```

try:

```
student_id = input('\nEnter student id to delete\n')
```

```
db.student.delete_many({'id':student_id})
```

```
print ('\nDeletion successful\n')
```

except Exception:

```
print("error while deleting")
```

def exit():

```
print("Bye")
```

main()

Output:

Enter Your choice:

2

```
{'_id': ObjectId('6150874245a900d9f2bf23c9'), 'Rollno': 138,  
'Student_name': 'Harshad', 'Course': 'MCAI'}
```

Please Enter the valid choice::

Press 1 to insert a record:

Press 2 to Display a Record:

Press 3 to Delete a Record:

Press 4 to Update a Record:

Enter student id :103

Enter student Name :pratik

Enter name of course :mca

Inserted data successfully

`_id: ObjectId("6150874245a900d9f2bf23c9")`
`id: "138"`
`name: "Harshad"`
`course: "MCAI"`

`_id: ObjectId("61535a2fb5f5f99590220915")`
`id: "104"`
`name: "Abhishek"`
`course: "MCA"`

`_id: ObjectId("61535a3db5f5f99590220916")`
`id: "303"`
`name: "Saurabh"`
`course: "MCA"`

`_id: ObjectId("61535a83b5f5f99590220917")`
`id: "103"`
`name: "Prathamesh"`
`course: "MCAI"`

31] Demonstrate step by step MongoDB connection in Python

Installing and Running MongoDB

Now that you're familiar with MongoDB, it's time to get your hands dirty and start using it. But first, you need to install it on your machine. MongoDB's official site provides two editions of the database server:

- [The community edition](#) offers the flexible document model along with ad hoc queries, indexing, and real-time aggregation to provide powerful ways to access and analyze your data. This edition is freely available.

[The enterprise edition](#) offers the same features as the community edition, plus other advanced features related to security and monitoring. This is the commercial edition, but you can use it free of charge for an unlimited time for evaluation and development purposes.

select the Windows platform in the Available Downloads box, choose the .msi installer that fits your current system, and click *Download*.

Run the installer and follow the on-screen instructions on the installation wizard. This page also provides information on how to [run HYPERLINK](#)

["https://docs.mongodb.com/manual/tutorial/install-mongodb-enterprise-on-windows/"](https://docs.mongodb.com/manual/tutorial/install-mongodb-enterprise-on-windows/) as a Windows service.

["https://docs.mongodb.com/manual/tutorial/install-mongodb-enterprise-on-windows/"](https://docs.mongodb.com/manual/tutorial/install-mongodb-enterprise-on-windows/) as a Windows service.

If you're on macOS, then you can use [Homebrew](#) to install MongoDB on your system. Also, make sure to follow the instruction to [run HYPERLINK](#)
["https://docs.mongodb.com/manual/tutorial/install-mongodb-enterprise-on-ubuntu/"](https://docs.mongodb.com/manual/tutorial/install-mongodb-enterprise-on-ubuntu/)[MongoDBHYPERLINK](#)
["https://docs.mongodb.com/manual/tutorial/install-mongodb-enterprise-on-ubuntu/"](https://docs.mongodb.com/manual/tutorial/install-mongodb-enterprise-on-ubuntu/) as a HYPERLINK
["https://docs.mongodb.com/manual/tutorial/install-mongodb-enterprise-on-ubuntu/"](https://docs.mongodb.com/manual/tutorial/install-mongodb-enterprise-on-ubuntu/)macOSHYP
["https://docs.mongodb.com/manual/tutorial/install-mongodb-enterprise-on-ubuntu/"](https://docs.mongodb.com/manual/tutorial/install-mongodb-enterprise-on-ubuntu/) service.

If you're on Linux, then the installation process will depend on your specific distribution. Make sure you run the MongoDB daemon, mongod, at the end of the installation.

Creating MongoDB Databases With the mongo Shell

If you've followed the installation and running instructions, then you should already have an instance of MongoDB running on your system. Now you can start creating and testing your own databases. In this section, you'll learn how to use the [mongo](#) shell to create, read, update, and delete documents on a database.

Running the mongo Shell

The mongo shell is an interactive [JavaScript](#) interface to MongoDB. You can use this tool to query and manipulate your data as well as to perform administrative operations. Since it's a JavaScript interface, you won't use the familiar SQL language to query the database. Instead, you'll use JavaScript code.

To launch the mongo shell, open your terminal or command line and run the following command:

```
$ mongo
```

This command takes you to the mongo shell. At this point, you'll probably see a bunch of messages with information on the shell's version and on the sever address and port. Finally, you'll be presented with the shell prompt (>) to enter queries and commands.

You can pass the database address as an argument to the mongo command. You can also use several options, such as specifying the host and port to access a remote database, and so on. For more details on how to use the mongo command, you can run `mongo --help`.

Establishing a Connection

When you run the mongo command without arguments, it launches the shell and connects to the default local server provided by the mongod process at `mongod://127.0.0.1:27017`. This means you're connected to the local host through port 27017.

By default, the mongo shell starts the session by establishing a connection to the test database. You can access the current database through the db object:

```
> db
```

```
test
```

```
>
```

In this case, db holds a reference to test, which is the default database. To switch databases, issue the command `use`, providing a database name as an argument.

For example, say you want to create a website to publish Python content, and you're planning to use MongoDB to store your tutorials and articles. In that case, you can switch to the site's database with the following command:

```
> use student
```

```
switched to db student
```

This command switches your connection to the student database. MongoDB doesn't create the physical database file on the file system until you insert real data into the database. So in this case, student won't show up in your current database list:

```
> show dbs
```

```
admin      0.000GB
```

```
config     0.000GB
```

```
local      0.000GB
```

```
>
```

The mongo shell provides a lot of features and options. It allows you to query and manipulate your data and also to manage the database server itself.

Instead of using a standardized query language such as SQL, the mongo shell uses the JavaScript programming language and a user-friendly [API](#). This API allows you to play around with your data, which is the topic for the next section.

Creating Collections and Documents

A MongoDB [database](#) is a physical container for [collections](#) of [documents](#). Each database gets its own set

of files on the file system. These files are managed by the MongoDB server, which can handle several databases.

In MongoDB, a collection is a group of documents.

Collections are somewhat analogous to tables in a traditional RDBMS, but without imposing a rigid schema. In theory, each document in a collection can have a completely different structure or set of fields.

In practice, documents in a collection commonly share a similar structure to allow uniform retrieval, insertion, and update processes. You can enforce a uniform document structure by using [document validation rules](#) during updates and insertions.

Allowing different document structures is a key feature of MongoDB collections. This feature provides flexibility and allows adding new fields to documents without the need for modifying a formal table schema.

To create a collection using the mongo shell, you need to point db to your target database and then create the collections using the dot notation:

30]Write a Menu Driver Program to add, display, update, delete and exit in a student database containing Student_id,Student_name,Course through Python-MongoDB connectivity.

```
import pymongo
myclient =
pymongo.MongoClient("mongodb://localhost:27017/")
```

```
mydb = myclient["Book"]
mycol = mydb["Book_detail"]
```

```
def addRecord():
    print("\n Please the Enter Book Details:\n")
    Book_id=int(input("\n Enter a Book id:\n"))
    Book_name=input("\nEnter a Book Name:\n")
    Author=input("\nEnter a Author name:\n")
    mydict = { "Book_id": Book_id, "Book_name":
Book_name,"Author":Author }
    x = mycol.insert_one(mydict)
    print("\n Your Record is Inserted successfully!!\n",x)
```

```
def displayRecord():
    for x in mycol.find():
        print(x)
```

```
def searchRecord():
    ch=int(input("\nEnter book id to search book:\n"))
    myquery = { "Book_id":ch }

    if(mycol.find(myquery)):
        print(f"\n{mycol.find(myquery)}\n")
    else:
        print("\n value not found!!!\n")
```

```
def sortRecord():

    mydoc = mycol.find().sort("Book_name")
    for x in mydoc:
```

```
print(x)
```

```
while(True):
```

```
    print("\n Press 1 to insert a record:\n")
```

```
    print("\n Press 2 to Display a Record:\n")
```

```
    print("\n Press 3 to Search a Record by Book id:\n")
```

```
    print("\n Press 4 to Sort a Record by Book Name:\n ")
```

```
    print("\n Press 5 to Exit\n")
```

```
    ch=int(input("\nEnter Your choice:\n"))
```

```
    if(ch==1):
```

```
        addRecord()
```

```
    if(ch==2):
```

```
        displayRecord()
```

```
    if(ch==3):
```

```
        searchRecord()
```

```
    if(ch==4):
```

```
        sortRecord()
```

```
    if (ch==5):
```

```
        break
```

```
    else:
```

```
        print("\n Please Enter the valid choice::\n")
```

Enter Your choice:

3

Enter book id to search book:

1

<pymongo.cursor.Cursor object at 0x000001DE2DF9202DF0>

Please Enter the valid choice::

Press 1 to insert a record:

Press 2 to Display a Record:

Press 3 to Search a Record by Book id:

Press 4 to Sort a Record by Book Name:

Press 5 to Exit

_id: ObjectId("615351abff662e5db9f1e904")
Book_id: 101
Book_name: "C language"
Author: "Balguruswamy"

_id: ObjectId("615351daff662e5db9f1e905")
Book_id: 102
Book_name: "Complete Reference"
Author: "Tata Publications"

_id: ObjectId("6153530aff662e5db9f1e906")
Book_id: 104
Book_name: "Durga soft"
Author: "Durga sir"

_id: ObjectId("61535355ff662e5db9f1e907")
Book_id: 103
Book_name: "7 secrets of life"
Author: "Rhonda Burns"

