INDEX

INDEX

S.No.	Date	Name of the Experiment / Programme / Practical	Page No.	Remarks
13	10/3/25	CRUD operations in mySQL database	39-43	

Page No.: 1

Date: 3/1/25

Name of the Experiment : Armstrong or not

Practical No.: 1

Aim:-

writing a program to check whether given number is Armstrong or not.

Program/source code :-

```
n= int(input(" Enter a number :"))
temp=n
tot=0
num_digits= len(str(n))
while n!=0:
    r=n%10
    tot+=r** num_digits
    n//=10
if temp==tot:
    print(f" {temp} is an Armstrong number")
else:
    print(f" {temp} is not an Armstrong number")
```

Output:-

Enter a number : 9474
9474 is an Armstrong number

Page No.: 4

Date: 9/1/25

Name of the Experiment : Given num is perfect or not

Practical No.: 2

Aim :-

Writing a program to check whether the given number is perfect or not.

Name of the Experiment :

Page No.: 5
Date:
Practical No.:

Source code:-

```
def is_Perfect(number):
    sum_of_divisors = sum(i for i in range(1,number)
                          if number % i == 0)
    return sum_of_divisors == number.

num = int(input("Enter a number : "))
if is_Perfect(num):
    print(f"{num} is a perfect number")
else:
    print(f"{num} is not a perfect number")
```

Output :-

```
Enter a number : 10
10 is not a perfect number.
```

Page No.: 7

Date: 17/1/25

Name of the Experiment : factorial of a given num using recursive

Practical No.: 3

Aim :

writing a program to find factorial of given number using recursive function.

Source code:

```
def recursive_factorial (n):
    if n == 1:
        return n
    else:
        return n * recursive_factorial (n-1)

num = 6
if num < 0:
    print ("Invalid input! please enter a positive number:")
elif num == 0:
    print ("factorial of number 0 is 1")
else:
    print ("Factorial of number", num, "=", recursive_
                                    factorial (num))
```

Page No.: 9

Date:

Name of the Experiment :

Practical No.:

Output:

Factorial of number 6 = 720.

Page No.: 10

Date: 20/1/25

Name of the Experiment : implement inheritance & polymorphism

Practical No.: 4

Aim :-

writing a program to implement inheritance and polymorphism.

Page No.: 11

Date:

Name of the Experiment :

Practical No.:

Source code:

```
class operations:
    def __init__(self):
        self.a=10
        self.b=20
        self.c="vinam"
        self.d="bratha"
    def execute(self):
        pass

class Addition(operations):
    def execute(self):
        print("Addition:", self.a+self.b)

class concat(operations):
    def execute(self):
        print("concatenation:", self.c+self.d)


add_obj=Addition()
concat_obj=concat()

add_obj.execute()
concat_obj.execute()
```

Output :-

Addition : 30

concatenation : vinambratha

Page No.: 13

Date: 23/1/25

Name of the Experiment: Try, except & finally block statements

Practical No.: 5

Aim:

Demonstrating a python code to print try, except and finally block statements.

Source code:-

```
try:
    print ("Try Block: Attempting to divide...")
    result = 10/0
except ZeroDivisionError:
    print("Except Block: Cannot divide by zero!")
finally:
    print ("finally Block: This will always execute.")
```

Output:-

```
Try Block: Attempting to divide
Except Block: Cannot divide by zero!
Finally Blak: This will always execute.
```

Page No.: 16

Date: 27/1/25

Name of the Experiment: string handling functions

Practical No.: 6

Aim:

Writing a program to demonstrate string handling functions.

Source code :-

```
def menu():
    while True:
        print("\n string Handling Functions")
        print("1. string Length")
        print("2. slicing string")
        print("3. UPPER CASE")
        print("4. lower case")
        print("5. string concatenation")
        print("0. Exit")
        choice = input("Enter your choice : ")
        if choice == "1":
            print("\n")
            string = input("Enter a string : ")
            print("string Length : ", len(string))
        elif choice == "2":
            print("\n")
            string = input("Enter a string :")
            start = int(input("Enter starting
                index : "))
```

```
    end = int(input("Enter ending index: "))
    print ("slicing string: ", string[(start):(end)])
elif choice == '3':
    print ("\n")
    string = input("Enter a string: ")
    print ("UPPER CASE: ", string.upper())
elif choice == '4':
    print("\n")
    string= input("Enter a string: ")
    print("lower case: ", string.lower())
elif choice == '5':
    print("\n")
    string1 = input ("Enter first string: ")
    string2 = input ("Enter second string: ")
    print ("string Concatenation: ", string1+" "+
                                    string2)

elif choice == '0':
    print ("Exiting... ")
    break
else:
    print ("Invalid! try again...")
menu()
```

output :-

output for choice:1

Enter a string : Lightning McQueen

string length : 17

output for choice: 2

Enter a string : Lightning McQueen

Enter starting index : 0

Enter Ending index : 10

slicing string : Lightning

output for choice : 3

Enter a string : LiGhtNiNG MCQUEEN

UPPER CASE : LIGHTNING MCQUEEN

Output for choice : 4

Enter a string : LIGHTNING MCQUEEN

lower case : lightning mcqueen

output for choice : 5

Enter first string : Lightning McQueen

Enter second string : ka-chow!

string concatenation : Lightning McQueen ka-chow!

Output for choice : 0

Exiting...

Page No.: 20

Date: 04/2/25

Name of the Experiment: max & min from Tuple

Practical No.: 7

Aim:

Writing a program to input n numbers from the user. store these numbers in a tuple. print the maximum and minimum number from this tuple.

Source code :-

```
n= int(input("Enter the number of elements: "))
num_list=[int(input(f"Enter number {i+1}: ")) for
                                    i in range(n)]
numbers = tuple(num_list)
print("In Tuple of numbers:", numbers)
print("maximum number: ", max(numbers))
print("minimum number: ", min(numbers))
```

output:

```
Enter the number of elements : 3
Enter number 1 : 45
Enter number 2 : 79
Enter number 3 : 2
Tuple of numbers : (45, 79, 2)
maximum number: 79
Minimum number : 2
```

Page No.: 23
Date: 15/2/25
Name of the Experiment : Dictionary
Practical No.: 8

Aim:

Write a program to enter names of employees and their salaries as input and store them in a dictionary.

Source code :

```
employees = {}
n = int(input("Enter the number of employees: "))
for i in range(n):
    name = input(f"Enter the name of employee {i+1}:")
    salary = int(input(f"Enter the salary of {name}: "))
    employees[name] = salary
print("\n Printing Dictionary: ")
print(employees)
print("\n printing Dictionary in a structured Format:")
print("\n Employees Salary Data: \n")
print("Emp Name \t Emp Salary")
print("..........")
for name, salary in employees.items():
    print(f"{name}\t\t{salary}")
```

Output:-

```
Enter the number of employees : 3
Enter the name of employee 1 : sethu
Enter the salary of sethu : 35999
Enter the name of employee 2 : vardhan
Enter the salary of vardhan : 99999
Enter the name of employee 3 : ram
Enter the salary of ram : 150000

Printing Dictionary:
{'sethu': 35999, 'vardhan': 99999, 'ram': 150000}

Printing Dictionary in a structured format:
Employee salary Data

Emp Name        Emp Salary
.....................
sethu           35999
vardhan         99999
ram             150000
```

Page No.: 26

Date: 17/2/25

Name of the Experiment: statistical operation using Numpy array

Practical No.: 9

Aim:

Writing a program to implement statistical operations on arrays using Numpy.

Source code :

```
import numpy as np
arr1 = np.array([2,10,6,8])
arr2 = np.array([2,2,3,2])
newarr = np.add(arr1,arr2)
print(newarr)
newarr = np.subtract(arr1,arr2)
print(newarr)
newarr = np.multiply(arr1,arr2)
print(newarr)
newarr = np.divide(arr1,arr2)
print(newarr)
newarr = np.power(arr1,arr2)
print(newarr)
print(min(arr1))
print(max(arr1))
```

output:

[4 12 9 10]

[0 8 3 6]

[4 20 18 16]

[1. 5. 2. 4.]

[4 100 216 64]

2

10

Page No.: 29

Date: 28/2/25

Name of the Experiment: Impor & Export CSV file to dataframes

Practical No.: 10

Aim:

Writing a program to import and export CSV file to DataFrame.

Page No.: 30

Date:

Name of the Experiment :

Practical No.:

Source code:

```
import pandas as pd
data = {
"character": ["Lightning McQueen", "Mater", "Doc Hudson"],
"car_model": ["stock car", "tow Truck", "Hudson Hornet"],
"role": ["Racer", "Tow Truck Driver", "Retired Racer"]}
df = pd.DataFrame(data)
csv_filename = "cars.csv"
df.to_csv(csv_filename, index=False)
print(f"Data exported to {csv_filename}")
df_imported = pd.read_csv(csv_filename)
print("Imported DataFrame:")
print(df_imported)
```

Page No.: 31
Date:
Name of the Experiment :
Practical No.:

output:-

Data exported to cars.csv

Imported Dataframe :

	character	car-model	role
0	Lightning McQueen	stock car	Racer
1	mater	Tow Truck	Tow Truck Driver
2	Doc Hudson	Hudson Hornet	Retired Racer

Page No.: 32
Date: 3/3/25
Practical No.: 11

Name of the Experiment : Dataframes basic operations

AIM:

Create the Data Frame sales containing year wise sales and perform basic operation on it.

SOL

source code:

```
import pandas as pd
data = {
'year': [2015, 2016, 2017, 2018, 2019],
'sales': [45000, 52000, 61000, 58000, 67000] }
df = pd.DataFrame(data)
print("sales DataFrame:\n", df)
print("\nBasic statistics:\n", df.describe())
total_sales = df['sales'].sum()
print("Total sales: ", total_sales)
average_sales = df['sales'].mean()
print("Average sales: ", average_sales)
max_sales = df['sales'].max()
print("\nMaximum sales: ", max_sales)
min_sales = df['sales'].min()
print("minimum sales: ", min_sales)
df['sales_after_increase'] = df['sales'] * 1.10
print("\nDataFrame after 10% increase in sales:\n"
      , df)
```

Name of the Experiment :

output:

Sales DataFrame:

	year	sales
0	2015	45000
1	2016	52000
2	2017	61000
3	2018	58000
4	2019	67000

Basic Statistics:

	year	sales
count	5	5
mean	2017	56600
std	1.58	8443.93
min	2015	45000
25%	2016	52000
50%	2017	58000
75%	2018	61000
max	2019	67000

Total sales : 283000
Average sales : 56600
maximum sales : 67000
minimum sales : 45000

Name of the Experiment :

Dataframe after 10% increase in sales :

	year	sales	sales_after_increase
0	2015	45000	49500
1	2016	52000	57200
2	2017	61000	67100.
3	2018	58000	63800
4	2019	67000	73700

Page No.: 36

Date: 4/3/25

Name of the Experiment: visualizing ten plot using matplotlib

Practical No.: 12

Aim:

visualize ten plots using matplotlib.

Name of the Experiment :

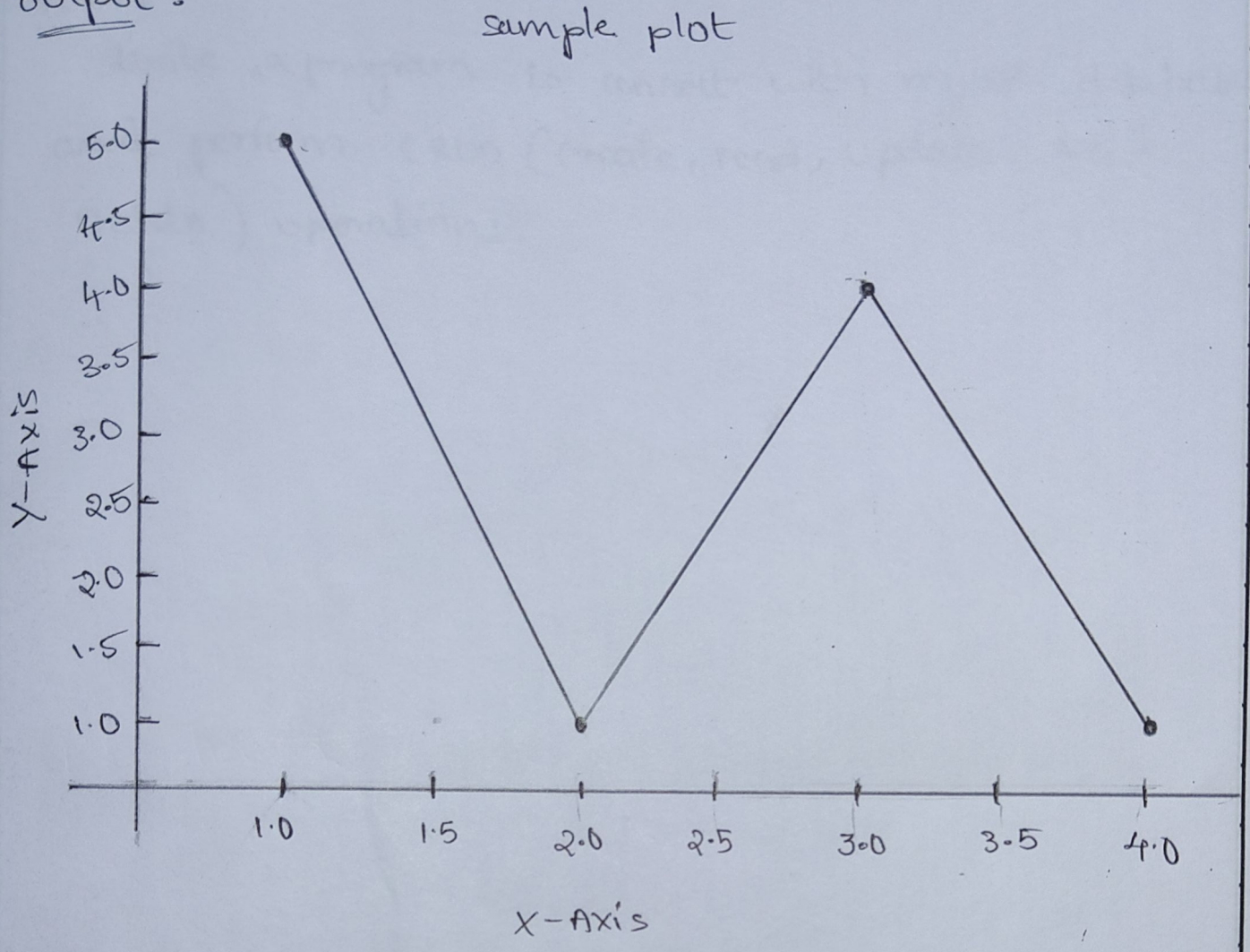
Page No.: 37
Date:
Practical No.:

Source code :-

```
import matplotlib.pyplot as plt
x = [1,2,3,4]
y = [5,1,4,1]
plt.plot(x,y, marker='o', linestyle="-", color='black',
         linewidth=2, markersize=8)
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')
plt.title('sample plot')
plt.show().
```

Name of the Experiment :

output :-



Page No.: 39

Date: 10/3/25

Name of the Experiment: CRUD operations in mysql database

Practical No.: 13

Aim:

Write a program to connect with mysql database and perform CRUD (create, read, update, and Delete) operations.

source code:

```
import mysql.connector
conn = mysql.connector.connect(
    host = "localhost",
    user = "your username",
    password = "your mysql password",
    database = "your database name")
cursor = conn.cursor()
cursor.execute("""
  CREATE TABLE IF NOT EXISTS shows(
  id INT AUTO_INCREMENT PRIMARY KEY,
  show_name VARCHAR(255) NOT NULL,
  genre VARCHAR(100) NOT NULL,
  episodes INT NOT NULL);""")
print("table 'shows' created successfully!")
shows_data=[
("man vs wild", "Adventure", 120),
("Food factory", "Food", 80),
("snakes in the city", "wildlife", 50),
("Deadliest catch", "Fishing", 200),
```

```
("MythBusters", "Science", 300) ]
cursor.executemany("INSERT INTO shows (show_name,
    genre, episodes) VALUES (%s, %s, %s)", shows_data)
conn.commit()
print("Inserted Discovery channel shows successfully!.")
cursor.execute("SELECT * FROM shows")
print("In Discovery channel shows:")
for row in cursor.fetchall():
    print(row)

cursor.execute("UPDATE shows SET episodes = 130 WHERE
        shows_name = 'man vs wild'")
conn.commit()
print("In Update 'man vs wild' episode count.")
cursor.execute("DELETE FROM shows WHERE show_
    name = 'snakes in the city'")
conn.commit()
print("In Deleted 'snakes in the city' from database.")
cursor.execute("SELECT * FROM shows")
print("In Final Show List:")
for row in cursor.fetchall():
    print(row)
```

```
cursor.close()
conn.close()
print("In Database connection closed.")
```

Name of the Experiment :

output :-

Table 'shows' created successfully!
Inserted Discovery channel shows successfully!
Discovery channel shows:
(1, 'man vs wild', 'Adventure', 120)
(2, 'Food Factory', 'Food', 80)
(3, 'snakes in the city', 'wildlife', 50)
(4, 'Deadliest Catch', 'Fishing', 200)
(5, 'mythBusters', 'science', 300)
Updated 'man vs wild' episode count.
Deleted 'snakes in the city' From database.
Final shows list:
(1, 'man vs wild', 'Adventure', 130)
(2, 'Food Factory', 'Food', 80)
(4, 'Deadliest Catch', 'Fishing', 200)
(5, 'mythBusters', 'science', 300)
Database connection closed.