Practical File
Computer Science(083)
Class XII (2025–26)

Submitted by - Pratyush Ranjan Maharana

Submitted to - Mrs.Kavita

Signature
Roll Number:-

Index of Practicals

Sl. No.	Name Of Practical	Date
1	Write a program in python to check a number whether it is prime or not (using functions).	
2	Write a program to check a number whether it is palindrome or not (using functions)	
3	Write a program to display ASCII code of a character and vice versa.	
4	Write a program using functions that reads a line and prints its statistics like:- Number of uppercase letters: Number of lowercase letters: Number of digits:	
5	Write a program to compute GCD and LCM of two numbers using functions.	
6	Write a program using a function FACT( ) to calculate the factorial of an integer.	
7	Write a program to generate random numbers between 1 to 6 and check whether a user won a lottery or not.	
8	Write a program to count the number of vowels present in a text file.	
9	Write a program to write those lines which have the character 'p' from one text file to another text file.	
10	Write a program to count number of words in a text file.	
11	Write a python program to append one more record in the existing text file name as student.txt.	
12	Write a python program to write student data in a binary file.	
13	Write a python program to read student data from a binary file.	

14	Write a python program to modify/update student data in a binary file.	
15	Write a python program to delete student data from a binary file.	
16	Write a python program to search a student record in a binary file.	
17	Create a binary file with name and roll number. Search for a given roll number and display the name, if not found display appropriate message.	
18	Create a binary file with roll number, name and marks. Input a roll number and update the marks.	
19	Write a program to perform read and write operation with .csv file.	
20	Create a CSV file by entering user-id and password, read and search the password for given userid.	
21	Write a program to pass an integer list as stack to a function and push only those elements in the stack which are divisible by 7.	
22	Write a menu-based program to perform the operation on stack in python.	
23	Queries using Create database, Show databases, USE, Create table, drop table, Show Tables, Describe, Rename, Alter, Select, From, Where, Insert, Update commands.	
24	Queries using DISTINCT, BETWEEN, IN, LIKE, IS NULL, ORDER BY, GROUP BY, HAVING.	
25	Queries for Aggregate functions: SUM( ), AVG( ), MIN( ), MAX( ), COUNT( ).	
26	Write a program to connect Python with MySQL using database connectivity and perform the following operation on data in database: Create a table in database.	
27	Write a program to connect Python with MySQL using database connectivity and perform the following operation on data in database:	

	Insert record in the table.	
28	Write a program to connect Python with MySQL using database connectivity and perform the following operation on data in database: Fetch records from the table using fetchone( ), fetchall() and fetchmany( ).	
29	Write a program to connect Python with MySQL using database connectivity and perform the following operation on data in database: Update record in the table.	
30	Write a program to connect Python with MySQL using database connectivity and perform the following operation on data in database: Delete record from the table.	

PROGRAM 1

1. Write a program to check a number whether it is prime or not.

```
def is_prime(n):
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True

num = int(input("Enter a number: "))
print(f"{num} is {'prime' if is_prime(num) else '
  ↪ not prime'}")
```

Output:

```
Enter a number: 29
29 is prime
```

PROGRAM 2

2.Write a program to check whether it is plaindrome or not

```
def is_palindrome(s):
    return str(s) == str(s)[::-1]

val = input("Enter a number/string: ")
print(f"'{val}' is {'a palindrome' if
  ↪ is_palindrome(val) else 'not a palindrome'}"
  ↪ )
```

Output:

```
Enter a number/string: madam
'madam' is a palindrome
```

PROGRAM 3

3. Write a program to display ASCII code of a character and vice versa.

```
def ascii():
    c = input("Enter character: ")
    if len(c) == 1:
        print("ASCII ->", ord(c))
    else:
        print("Only 1 character allowed.")
def char():
    n = int(input("Enter ASCII code: "))
    if 0 <= n <= 127:
        print("Character",chr(n))
    else:
        print("Invalid ASCII code.")
ascii()
char()
```

Output:

```
Enter a character: Z
ASCII value: 90
Enter ASCII code (0-127): 97
Character: a
```

PROGRAM 4

4. Write a program using function that reads a line and prints its statistics like

number of uppercase: number of lowercase: number of digits :

```
def string_stats(text):
    return {
        'Uppercase': sum(1 for c in text if c.
          ↪ isupper()),
        'Lowercase': sum(1 for c in text if c.
          ↪ islower()),
        'Digits': sum(1 for c in text if c.
          ↪ isdigit())
    }
text = input("Enter a line of text: ")
stats = string_stats(text)
print("\nStatistics:")
for k, v in stats.items():
    print(f"{k}: {v}")
```

Output:

```
Enter character: p
ASCII -> 112
Enter ASCII code: 113
Character     q
```

PROGRAM 5

5. Write a program to compute GCD and LCM of two numbers.

```
def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a
def lcm(a, b):
    return abs(a * b) // gcd(a, b)

num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))

g = gcd(num1, num2)
l = lcm(num1, num2)

print(f"GCD of {num1} and {num2} is: {g}")
print(f"LCM of {num1} and {num2} is: {l}")
```

Output:

```
Enter first number: 2
Enter second number: 3
GCD of 2 and 3 is: 1
LCM of 2 and 3 is: 6
```

PROGRAM 6

6. Write a program using the FACT() function to calculate the factorial of an integer.

```
def FACT(n):
    if n < 0:
        return "Factorial not defined for
          ↪ negative numbers."
    elif n == 0 or n == 1:
        return 1
    else:
        fact = 1
        for i in range(2, n + 1):
            fact *= i
        return fact
num = int(input("Enter an integer: "))
result = FACT(num)
print(f"The factorial of {num} is: {result}")
```

Output:

```
Enter a number: 5
Factorial of 5: 120
```

PROGRAM 7

7. Write a program to generate random numbers between 1 to 6 and check whether a user won a Lottery or not.

```
import random
def roll_dice():
    return random.randint(1, 6)
user_guess = int(input("Guess a number between 1 and 6: "))
lottery_number = roll_dice()
if user_guess == lottery_number:
    print("you won")
else:
    print(f"you loser. The winning number was {lottery_number}. Try again!")
```

Output:

```
Guess a number between 1 and 6: 2
Sorry, you lost. The winning number was 4. Try again!
```

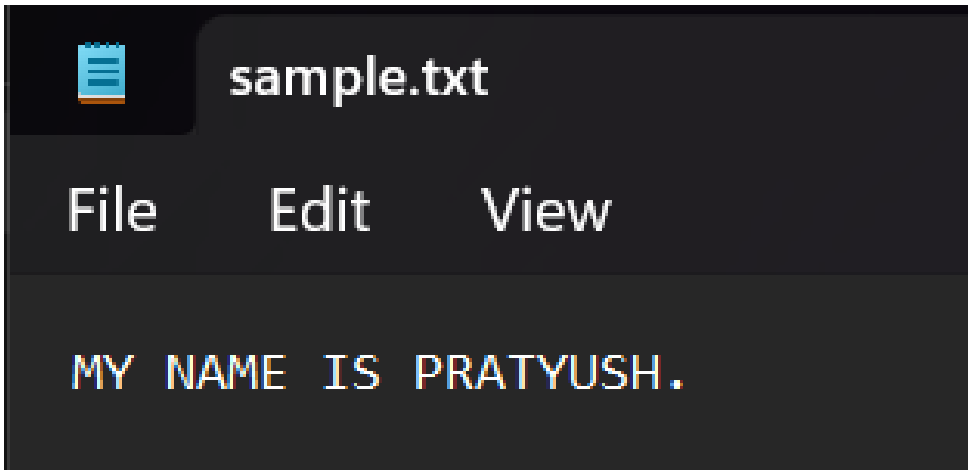
PROGRAM 8

8. Vowel Counter in Text File

```
def count_vowels(filename):
    vowels = 'aeiouAEIOU'
    with open(filename, 'r') as f:
        return sum(1 for char in f.read() if char
          ↪  in vowels)
with open('sample.txt', 'w') as f:
    f.write("MY NAME IS
    PRATYUSH.\n")
print("Vowel count:", count_vowels('sample.txt'))
```

Output:

```
Vowel count: 5
```



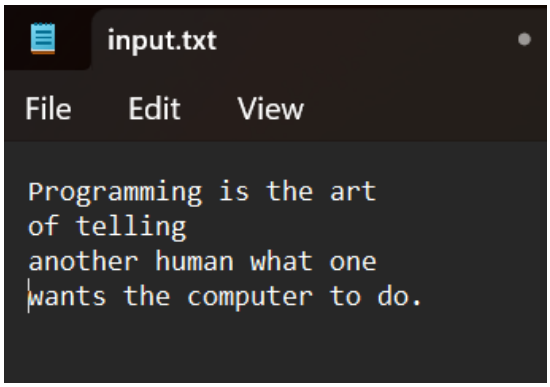
PROGRAM 9

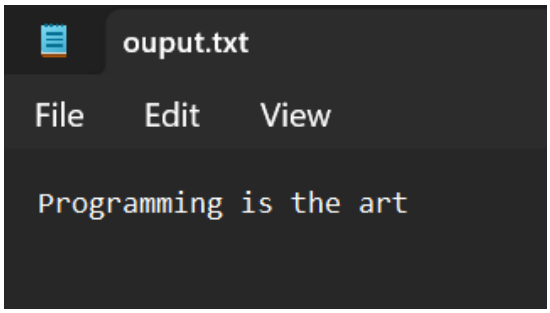
9. Filter lines containing 'p'

```
with open('input.txt', 'w') as f:
    f.write("This is a text with some p's.\n")
    f.write("Python is powerful.\n")
    f.write("This line does not contain the
      ↪ letter p.\n")
    f.write("Pineapple is a fruit.\n")
with open('input.txt', 'r') as infile, open('
  ↪ output.txt', 'w') as outfile:
    for line in infile:
        if 'p' in line.lower():
            outfile.write(line)

print("Lines with 'p' copied to output.txt")
```

Output:

```
Lines with 'p' copied to output.txt
```





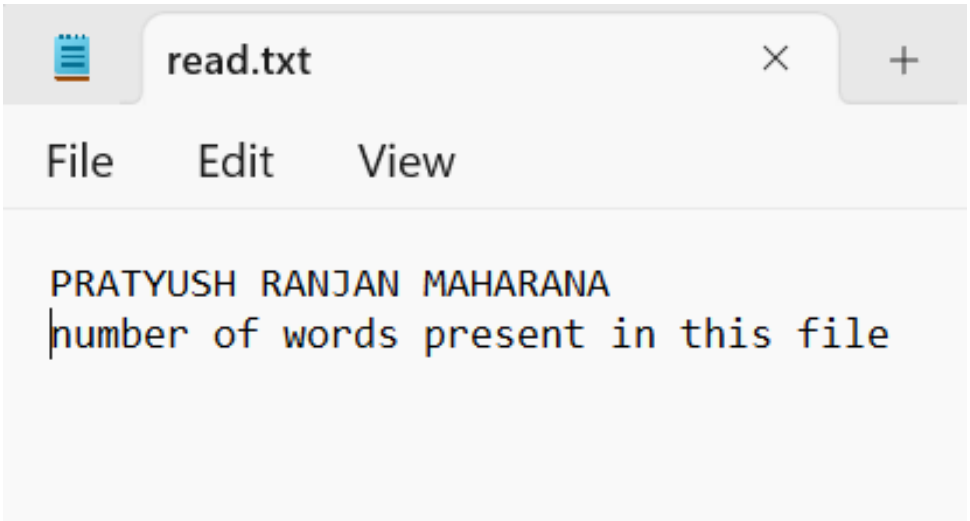
PROGRAM 10

10. Write a program to count number of words in text file.

```
with open('read.txt', 'w') as f:
    f.write("PRATYUSH RANJAN MAHARANA.\n")
    f.write("number of words present in this file
      ↪ \n")
def count_words(filename):
    with open(filename, 'r') as f:
        return len(f.read().split())
print("Total words:", count_words('read.txt'))
```

Output:

```
Total words: 10
```



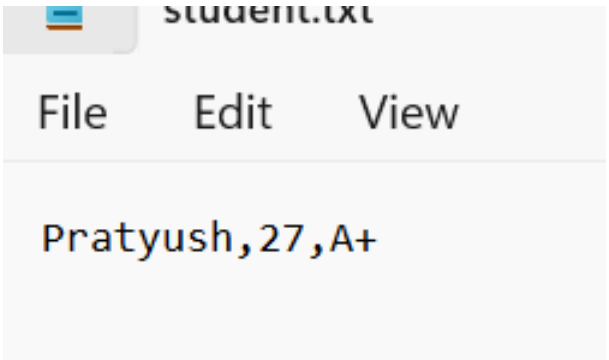
PROGRAM 11

11. Write a program to append one more record in the existing text file name as student.txt

```
def append_student_record():
    filename = "student.txt"
    print("Enter student details to append:")
    name = input("Name: ")
    roll_no = input("Roll Number: ")
    grade = input("Grade: ")
    with open(filename, 'a') as file:
        file.write(f"{name}, {roll_no}, {grade}\n
          ↪ ")
append_student_record()
```

Output:

```
Enter student details to append:
Name: pratyush
Roll Number: 27
Grade: A+
```



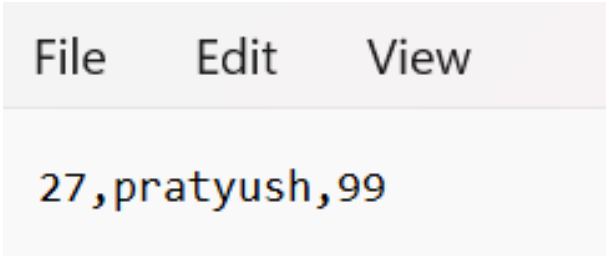
PROGRAM 12

12. Write a python program to write student data in binary file

```
def get_student_data():
  roll_no=int(input("enter roll number:"))
  name=input("enter name:")
  marks=int(input("enter marks:"))
  return f"{roll_no},{name},{marks}\n"
def write_to_binary_file(filename,student):
  with open(filename,"ab") as f:
    f.write(student.encode('utf-8'))
def main():
  filename="students.dat"
  while True:
    data=get_student_data()
    write_to_binary_file(filename,data)
    cont=input("add another record?(y/n):").lower
      ↪ ()
    if cont !='y':
      break
  print(f"data written to {filename}")
main()
```

Output:

```
enter roll number:27
enter name:pratyush
enter marks:99
add another record?(y/n):n
data written to students.dat
```


File Edit View
27,pratyush,99

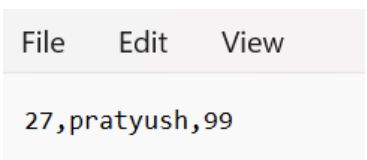
PROGRAM 13

13. Write a python program to read student data from a binary file.

```
def read_from_binary_file(filename):
  try:
      with open(filename, "rb") as f:
          data = f.read().decode('utf-8')
          if not data:
              print("No data found in the file.")
              return
          print("Roll No | Name | Marks")
          for line in data.strip().split("\n"):
              roll_no, name, marks = line.split(
                ↪ ",")
              print(f"{roll_no} , {name} , {marks
                ↪ }")
  except FileNotFoundError:
      print(f"The file {filename} does not exist.
        ↪ ")
def main():
    filename = "students.dat"
    read_from_binary_file(filename)
main()
```

Output:

```
27 , pratyush , 99
```



PROGRAM 14

14. Write a python program to modify/update student data in a binary file.

```
import pickle
try:
    with open('students.dat', 'rb') as f:
        data = pickle.load(f)
except:
    data = []
print("\nBefore:", *data, sep="\n")
roll = int(input("\nRoll to update: "))
for s in data:
    if s['roll'] == roll:
        s['name'] = input("New name: ")
        s['marks'] = int(input("New marks: "))
        break
else:
    print("Not found."); exit()
with open('students.dat', 'wb') as f:
    pickle.dump(data, f)
print("\nAfter:", *data, sep="\n")
```

Output:

```
 Student Data Before Update
{'roll': 1, 'name': 'pratyush', 'marks': 99}
{'roll': 2, 'name': 'paaji', 'marks': 0}
{'roll': 3, 'name': 'rahul', 'marks': 79}
Enter roll number to update: 2
Enter new name: rohan
Enter new marks: 99
Student Data After Update
```

```
{'roll': 1, 'name': 'pratyush', 'marks': 99}
{'roll': 2, 'name': 'rohan', 'marks': 99}
{'roll': 3, 'name': 'rahul', 'marks': 79}
```

File Edit View

```
'roll': 1, 'name': 'Pratyush', 'marks': 99
'roll': 2, 'name': 'Rohan', 'marks': 99
'roll': 3, 'name': 'Rahul', 'marks': 79
```

PROGRAM 15

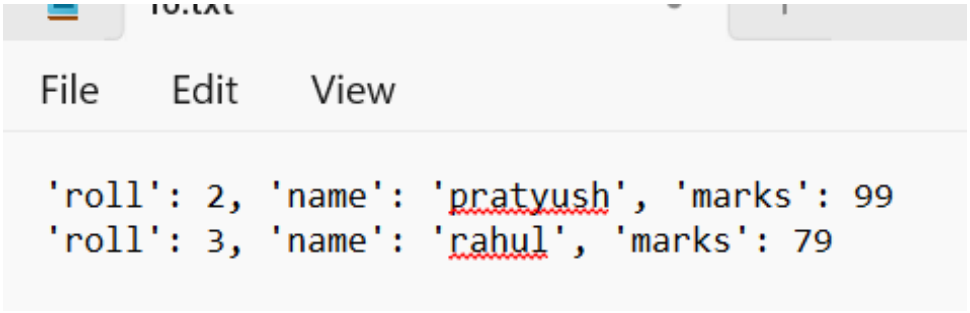
15. Write a python program to delete student data from a binary file.

```
import pickle
try:
    with open('students.dat', 'rb') as f:
        data = pickle.load(f)
except:
    data = []
print("\nBefore:", *data, sep="\n")
roll = int(input("\nRoll to delete: "))
deleted = [s for s in data if s['roll'] == roll]
if deleted:
    data = [s for s in data if s['roll'] != roll]
    with open('students.dat', 'wb') as f:
        pickle.dump(data, f)
    print(f"\nDeleted: {deleted[0]}")
else:
    print("\nNot found.")
print("\nAfter:", *data, sep="\n")
```

Output:

```

Before:
{'roll': 1, 'name': 'ok', 'marks': 2}
{'roll': 2, 'name': 'pratyush', 'marks': 99}
{'roll': 3, 'name': 'rahul', 'marks': 79}
Roll to delete: 1
Deleted: {'roll': 1, 'name': 'ok', 'marks': 2}

After:
```

```
{'roll': 2, 'name': 'pratyush', 'marks': 99}
{'roll': 3, 'name': 'rahul', 'marks': 79}
```



PROGRAM 16

16. Write a python program to search a student record in a binary file.

```
import pickle
try:
    with open('students.dat', 'rb') as f:
        students = pickle.load(f)
except:
    students = []
roll = int(input("Enter Roll Number to Search: ")
  ↪ )
record = next((s for s in students if s['roll']
  ↪ == roll), None)

if record:
    print("\nStudent Found:")
    print(f"Roll: {record['roll']}\nName: {record
      ↪ ['name'].title()}\nMarks: {record['marks
      ↪ ']}")
else:
    print("\nNo student found with that roll
      ↪ number.")
```

Output:

```
Enter Roll Number to Search: 2
Student Found:
Roll: 2
Name: Pratyush
Marks: 99
```

PROGRAM 17

17. Create a binary file with name and roll number.Search for a given roll number and display the name,if not found display appropriate message.

```
import pickle
def file(filename):
    with open(filename, 'wb') as f:
        n = int(input("Enter number of students: "))
        for _ in range(n):
            name = input("Enter name: ")
            roll = int(input("Enter roll number: "))
            data = {'name': name, 'roll': roll}
            pickle.dump(data, f)
    print("File created successfully.\n")
def search_roll(filename, roll_to_search):
    found = False
    try:
        with open(filename, 'rb') as f:
            while True:
                try:
                    data = pickle.load(f)
                    if data['roll'] == roll_to_search:
                        print(f"Roll Number: {roll_to_search}, Name: {data['name']} ")
                        found = True
```

```
                            break
                    except EOFError:
                        break
            if not found:
                print("Roll number not found.")
        except FileNotFoundError:
            print("File does not exist.")

filename = "students.dat"
file(filename)
roll = int(input("Enter roll number to search: ")
  ↪ )
search_roll(filename, roll)
```

Output:

```
Enter number of students: 1
Enter name: pratyush
Enter roll number: 27
File created successfully.
Enter roll number to search: 27
Roll Number: 27, Name: pratyush
#if rollno doesn't exist in file then:-
Enter roll number to search: 1
Roll number not found.
```

PROGRAM 18

18. Create a binary file with roll number,name,and marks. Input a roll number and update the marks

```
import pickle

def create_file(fname):
    with open(fname, 'wb') as f:
        for _ in range(int(input("Number of students: "))):
            data = {'roll': int(input("Roll: ")), 'name': input("Name: "), 'marks': float(input("Marks: "))}
            pickle.dump(data, f)
def update_marks(fname, roll):
    students, found = [], False
    with open(fname, 'rb') as f:
        try:
            while True: students.append(pickle.load(f))
        except EOFError: pass
    for s in students:
        if s['roll'] == roll:
            s['marks'] = float(input(f"Current marks: {s['marks']}. Enter new marks: "))
            found = True
            break
    if found:
        with open(fname, 'wb') as f:
            for s in students: pickle.dump(s, f)
        print("Marks updated successfully.")
```

```
    else:
        print("Roll number not found.")
file = 'students.dat'
create_file(file)
update_marks(file, int(input("Enter roll to
  ↪ update marks: ")))
```

Output:

```
Number of students: 1
Roll: 27
Name: pratyush
Marks: 99
Enter roll to update marks: 2
Roll number not found.
```

PROGRAM 19

19. Write a program to perform read and write operation with csv.file

```
import csv
def write():
  file_name="meow.csv"
  data=[
      ["roll","name","marks"],  #data is written
        ↪ to meow.csv
      [1 ,'pratyush', 99],
      [2 ,'rohan', 98],
      [3 ,'piyush', 97],
  ]
  with open(file_name,mode='w',newline='') as f:
    writer=csv.writer(f)
    writer.writerows(data)
  print("data is written to",file_name)

def read():
    file_name='meow.csv'
    print("\nReading data from",file_name)
    with open(file_name,mode='r') as f:
      reader=csv.reader(f)
      for row in reader:
        print(row)
write()
read()
```

Output:

```
data is written to meow.csv

```

```
Reading data from meow.csv
['roll', 'name', 'marks']
['1', 'pratyush', '99']
['2', 'rohan', '98']
['3', 'piyush', '97']
```

PROGRAM 20

20. Create a CSV file by entering user-id and password, read and search the password for given userid.

```
def create_csv():
    if not os.path.exists(CSV):
        with open(CSV,"w",newline="") as f:
            writer = csv.DictWriter(f,fieldnames
              ↪ =["userid","password"])
            writer.writeheader()
def add_users():
    while True:
        uid = input("Enter numeric userid (leave
          ↪ empty to stop): ").strip()
        if uid == "": break
        pwd = input("Enter password: ").strip()
        with open(CSV,"a",newline="") as f:
            writer = csv.DictWriter(f,fieldnames
              ↪ =["userid","password"])
              ↪                writer.writerow({"
              ↪ userid":uid,"password":pwd})
def show_all():
    if os.path.exists(CSV):
        print("\nCurrent users:")
        with open(CSV,newline="") as f:
            for r in csv.DictReader(f):
                print(r["userid"], r["password"])
def find_password(uid):
    uid = str(uid).strip()
    if os.path.exists(CSV):
        with open(CSV,newline="") as f:
            for r in csv.DictReader(f):
```

```
                if r["userid"].strip() == uid:
                    return r["password"]
    return None
create_csv()
add_users()
show_all()
search = input("\nEnter numeric userid to search.
").strip()
print("Password:", pw if pw else "Not found")
```

output

```
Enter numeric userid (leave empty to stop): 101
Enter password: pass101
Enter numeric userid (leave empty to stop):
Current users:
prm123
prm456
rohl789
101 pass101
```

PROGRAM 21

21.Write a program to pass an integer list as stack to a function and push only those elements in the stack which are divisible by 7.

```
def gamma(numbers):
  stack=[]
  for num in numbers:
    if num%7==0:
      stack.append(num)
  return stack
nums=[1,2,3,4,5,6,7,8,9]
result=gamma(nums)
print("stack with elements divisible by 7 are",
  ↪ result)
```

output

```
stack with elements divisible by 7 are [7]
```

PROGRAM 22

22.Write a menu-based program to perform the operation on stack in python.

```
stack = []
while True:
    print("\n1.Push  2.Pop  3.Display 4.Peek 5.
      ↪ Exit")
    ch = input("Enter choice: ")
    if ch == '1':
        stack.append(input("Enter element: "))
    elif ch == '2':
        print("Popped:", stack.pop() if stack
          ↪ else "Stack Empty")
    elif ch == '3':
        print("Stack:", stack if stack else "
          ↪ Empty")
    elif ch == '4':
        if not stack:
          print("stack is empty")
        else:
          print(f"top element:{stack[-1]}")
    elif ch ==5:
      break
    else:
        print("Invalid choice!")
```

output

```
1.Push  2.Pop  3.Display 4.Peek 5.Exit
Enter choice: 1
Enter element: 10 12

```

```
1.Push  2.Pop  3.Display 4.Peek 5.Exit
Enter choice: 2
Popped: 10 12

1.Push  2.Pop  3.Display 4.Peek 5.Exit
Enter choice: 3
Stack: Empty

1.Push  2.Pop  3.Display 4.Peek 5.Exit
Enter choice: 1
Enter element: 11

1.Push  2.Pop  3.Display 4.Peek 5.Exit
Enter choice: 4
top element:11

1.Push  2.Pop  3.Display 4.Peek 5.Exit
Enter choice: 5
```

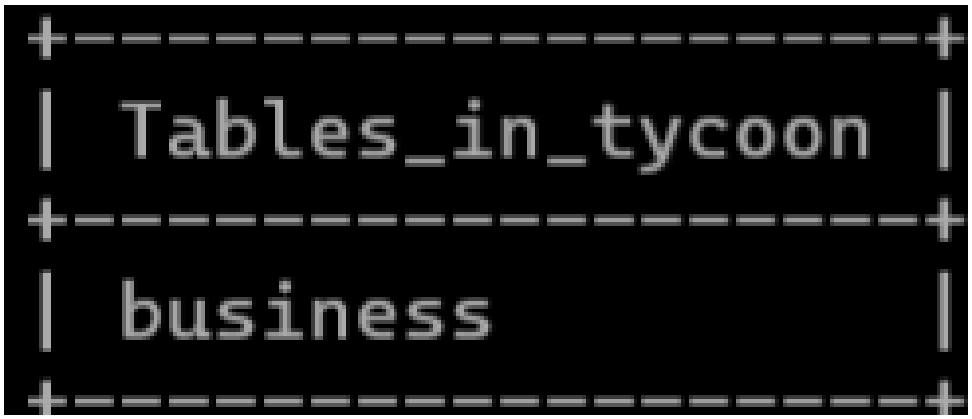
PROGRAM 23

23. Queries using Create database, Show databases, USE, Create table, drop table, Show table, Describe, Rename, Alter, Select, From, Where, Insert, Update commands:-

```
>> Create database: CREATE DATABASE TYCOON;
>> Show databases: SHOW DATABASES;
```

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| community          |
| company            |
| gpt                |
| information_schema |
| llm                |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| tycoon             |
| world              |
+--------------------+
11 rows in set (0.00 sec)
```

```
>>Use: USE TYCOON
>> CREATE TABLE:
 CREATE TABLE BUSINESS(
 ID INT PRIMARY KEY,
 NAME VARCHAR(30),
 MONEY DECIMAL(15,2),
 FIELD VARCHAR(50));

>>Show Tables: SHOW TABLES;
```



```
Describe: DESC BUSINESS;
```

Field	Type	Null	Key	Default	Extra
ID	int	NO	PRI	NULL	
name	varchar(30)	YES		NULL	
money	decimal(15,2)	YES		NULL	
field	varchar(50)	YES		NULL	

```
>>Insert: INSERT INTO BUSINESS VALUES(002,'PIYUSH
  ↪ ',
100000000 ,'ML');

>>Update: UPDATE BUSINESS SET MONEY=0 WHERE ID
  ↪ =1;

>>Rename: RENAME TABLE BUSINESS TO ELONMUSK;

>>Alter: ALTER TABLE ELONMUSK ADD COUNTRY VARCHAR
  ↪ (30);

>>Select from: SELECT*FROM ELONMUSK;
```

```
mysql> INSERT INTO business VALUES(002,'piyush',100000000,'ML');
Query OK, 1 row affected (0.01 sec)

mysql> select*from business;
+----+----------+--------------+-------+
| ID | name     | money        | field |
+----+----------+--------------+-------+
|  1 | pratyush |  10000000.00 | AI    |
|  2 | piyush   | 100000000.00 | ML    |
+----+----------+--------------+-------+
2 rows in set (0.00 sec)
```

```
Where: SELECT* FROM BUSINESS WHERE FIELD='AI';
```

```
mysql> select * from business where field='AI';
+----+----------+-------------+-------+
| ID | name     | money       | field |
+----+----------+-------------+-------+
|  1 | pratyush | 10000000.00 | AI    |
+----+----------+-------------+-------+
1 row in set (0.00 sec)
```

```
>>Drop column: DROP COLUMN COUNTRY;
```

```
mysql> alter table elonmusk drop column country ;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select*from elonmusk;
+----+----------+--------------+-------+
| ID | name     | money        | field |
+----+----------+--------------+-------+
|  1 | pratyush |         0.00 | AI    |
|  2 | piyush   | 100000000.00 | ML    |
+----+----------+--------------+-------+
2 rows in set (0.00 sec)
```

```
>>Drop table: DROP TABLE ELONMUSK;
```

```
mysql> drop table elonmusk;
Query OK, 0 rows affected (0.01 sec)

mysql> select*from elonmusk;
ERROR 1146 (42S02): Table 'tycoon.elonmusk' doesn't exist
```

PROGRAM 24

24. Queries using DISTINCT, BETWEEN, IN, LIKE, IS NULL, ORDER BY, HAVING:-

```
>>Distinct: SELECT DISTINCT FIELD FROM ELONMUSK;
```

```
mysql> select distinct field from elonmusk;
+-------+
| field |
+-------+
| ML    |
| AI    |
+-------+
2 rows in set (0.00 sec)
```

```
>>Between: SELECT NAME, MONEY FROM ELONMUSK WHERE
  ↪  MONEY BETWEEN 90000 AND 200000000;
```

```
mysql> select name, money
    -> from elonmusk
    -> where money between 900000 and 200000000;
+----------+--------------+
| name     | money        |
+----------+--------------+
| pratyush |    999999.00 |
| piyush   | 100000000.00 |
+----------+--------------+
2 rows in set (0.00 sec)
```

```
>>IN: SELECT NAME,FIELD,FROM ELONMUSK WHERE FIELD
  ↪  IN('AI','ML');
```

```
mysql> select name,field
    -> from elonmusk
    -> where field in('AI','ML');
+----------+-------+
| name     | field |
+----------+-------+
| pratyush | ML    |
| piyush   | AI    |
+----------+-------+
2 rows in set (0.00 sec)
```

```
>>Like: SELECT*FROM ELONMSUK WHERE NAME LIKE'p%';
```

```
mysql> select*from elonmusk where name like 'p%';
+----+----------+-------------+-------+
| ID | name     | money       | field |
+----+----------+-------------+-------+
|  1 | pratyush |    999999.00 | ML    |
|  2 | piyush   | 100000000.00 | AI    |
+----+----------+-------------+-------+
2 rows in set (0.00 sec)
```

```
>>IS NULL: SELECT*FROM ELONMUSK WHERE FIELD IS
  ↪ NULL;
```

```
mysql> select*from elonmusk where field is null;
Empty set (0.00 sec)
```

```
>>Order by:SELECT*FROM ELONMUSK ORDER BY MONEY
  ↪ DESC;
```

```
mysql> select*from elonmusk
    -> order by money desc;
+----+----------+--------------+-------+
| ID | name     | money        | field |
+----+----------+--------------+-------+
|  2 | piyush   | 100000000.00 | AI    |
|  1 | pratyush |    999999.00 | ML    |
+----+----------+--------------+-------+
2 rows in set (0.00 sec)
```

```
>>Group by: SELECT FIELD, COUNT(x) AS
  ↪ TOTAL_PEOPLE FROM ELONMUSK GROUP BY FIELD;
```

```
mysql> select field, count(*) as total_people
    -> from elonmusk
    -> group by field;
+-------+--------------+
| field | total_people |
+-------+--------------+
| ML    |            1 |
| AI    |            1 |
+-------+--------------+
2 rows in set (0.00 sec)
```

```
>>HAVING: SELECT FIELD,AVG(MONEY) AS AVG_MONEY
  ↪ FROM ELONMUSK GROUP BY FIELD HAVING AVG(
  ↪ MONEY)>1000000;
```

```
mysql> select field ,avg(money) as avg_money
    -> from elonmusk
    -> group by field
    -> having avg(money)>1000000;
+-------+------------------+
| field | avg_money        |
+-------+------------------+
| AI    | 100000000.000000 |
+-------+------------------+
1 row in set (0.00 sec)
```

PROGRAM 25

25. Queries for aggregate functions:-SUM(),AVG(),MIN(), MAX(),COUNT()

```
>>Sum(): SELECT SUM(MONEY) AS TOTAL_MONEY FROM
  ↪ ELONMUSK;
```

```
mysql> select sum(money) as total_money
    -> from elonmusk;
+--------------+
| total_money  |
+--------------+
| 100999999.00 |
+--------------+
1 row in set (0.00 sec)
```

```
>>AVG(): SELECT AVG(MONEY) AS AVERAGE_MONEY FROM
  ↪ ELONMUSK;
```

```
mysql> select avg(money) as average_money
    -> from elonmusk;
+-----------------+
| average_money   |
+-----------------+
| 50499999.500000 |
+-----------------+
1 row in set (0.00 sec)
```

```
>>MIN(): SELECT MIN(MONEY) AS MIN_MONEY FROM
  ↪ ELONMUSK;
```

```
mysql> select min(money) as min_money
    -> from elonmusk;
+-----------+
| min_money |
+-----------+
| 999999.00 |
+-----------+
1 row in set (0.00 sec)
```

```
>>MAX(): SELECT MAX(MONEY) AS MAX_MONEY FROM
  ↪ ELONMUSK;
```

```
mysql> select max(money) as max_money
    -> from elonmusk;
+--------------+
| max_money    |
+--------------+
| 100000000.00 |
+--------------+
1 row in set (0.00 sec)
```

```
>>COUNT(): SELECT COUNT(X) AS TOTAL_RECORDS FROM
  ↪ ELONMUSK;
```

```
mysql> select count(*) as total_records
    -> from elonmusk;
+---------------+
| total_records |
+---------------+
|             2 |
+---------------+
1 row in set (0.00 sec)
```

PROGRAM 26

26. Write a program to connect python with MySQL using database connectivity and perform the following operation on data in database: Create a table in database:-

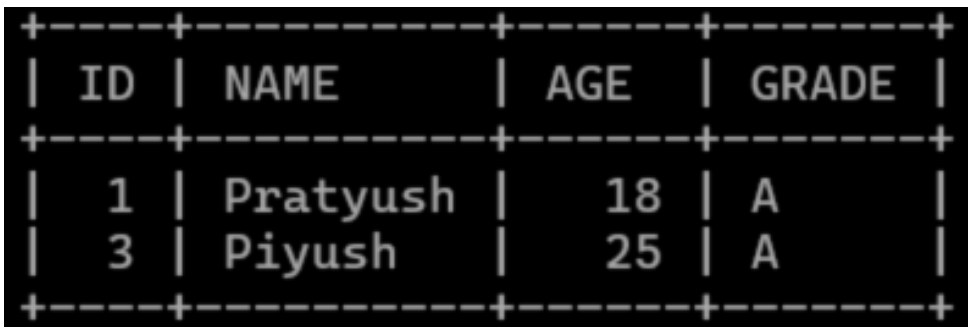
```
import mysql.connector as sql
con=sql.connect(host="localhost",user="root",
  ↪ password="root",database="APPLE1")
cursor=con.cursor()
cursor.execute('''CREATE TABLE IF NOT EXISTS
  ↪ students(
  ID INT AUTO_INCREMENT PRIMARY KEY,
  NAME VARCHAR(50),
  AGE INT,
  GRADE VARCHAR(60)
)
''')
con.commit()
```

```
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| ID    | int         | NO   | PRI | NULL    | auto_increment |
| NAME  | varchar(50) | YES  |     | NULL    |                |
| AGE   | int         | YES  |     | NULL    |                |
| GRADE | varchar(60) | YES  |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
```

PROGRAM 27

27. Write a program to connect python with MySQL using database connectivity and perform the following operation on data in database: Insert record in the table:-

```
import mysql.connector as sql
con=sql.connect(host="localhost",user="root",
  ↪ password="root",database="APPLE1")
cursor=con.cursor()
cursor.execute('''CREATE TABLE IF NOT EXISTS
  ↪ students(
  ID INT AUTO_INCREMENT PRIMARY KEY,
  NAME VARCHAR(50),
  AGE INT,
  GRADE VARCHAR(60)
)
''')
cursor.execute("INSERT INTO students (name, age,
  ↪ grade) VALUES ('Pratyush', 17, 'A')")
cursor.execute("INSERT INTO students (name, age,
  ↪ grade) VALUES ('Piyush', 25, 'A')")
con.commit()
```



ID	NAME	AGE	GRADE
1	Pratyush	18	A
3	Piyush	25	A

PROGRAM 28

28. Write a program to connect python with MySQL using database connectivity and perform the following operation on data in database: Fetch records from the table using fetchone(), fetchmany() and fetchall():-

```
1  cursor.execute("SELECT*FROM students")
2  rows=cursor.fetchone()
3  for row in rows:
4  |  print(row)
```

```
1
Pratyush
17
A
```

Figure 1: Fetchone()

```
1  cursor.execute("SELECT*FROM students")
2  rows=cursor.fetchmany(1)
3  for row in rows:
4  |  print(row)
```

```
(1, 'Pratyush', 17, 'A')
```

Figure 2: Fetchmany(size)

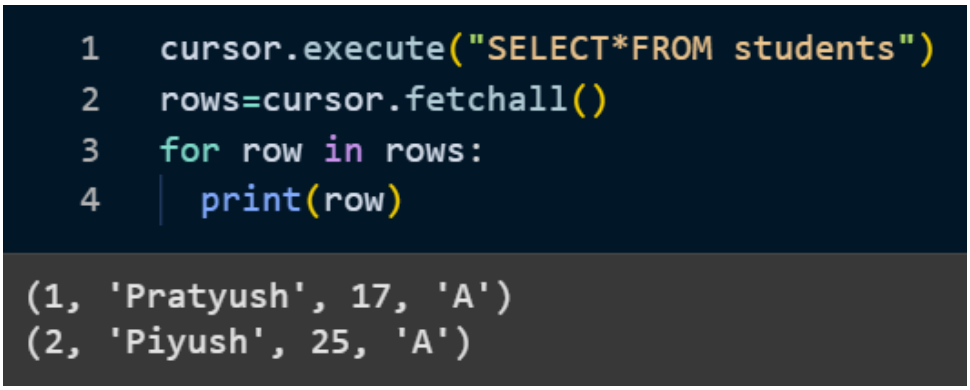


Figure 3: Fetchall()

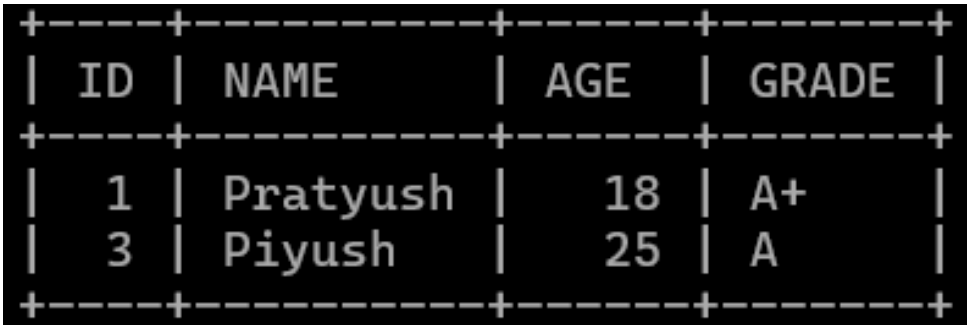
PROGRAM 29

29. Write a program to connect python with MySQL using database connectivity and perform the following operations on data in database: Update record in table :-

```
import mysql.connector as sql

con = sql.connect(
    host="localhost",
    user="root",
    password="root",
    database="APPLE1"
)
cursor = con.cursor()
name_to_update = 'Pratyush'
new_age = 18
new_grade = 'A+'

cursor.execute("UPDATE students SET AGE = %s,
  ↪ GRADE = %s WHERE NAME = %s", (new_age,
  ↪ new_grade, name_to_update))
con.commit()

print(cursor.rowcount, "The record was updated
  ↪ successfully .")
```

Output:

```
The record was updated successfully.
```



ID	NAME	AGE	GRADE
1	Pratyush	18	A+
3	Piyush	25	A

Figure 4: updating record in the table

PROGRAM 30

30. Write a program to connect Python with MySQL using database connectivity and perform the following operation on data in database:- Delete record from the table

```
import mysql.connector as sql
con = sql.connect(
    host="localhost",
    user="root",
    password="root",
    database="APPLE1"
)
cursor = con.cursor()
name_to_delete = 'Piyush'
cursor.execute("DELETE FROM students WHERE NAME =
  ↪  %s", (name_to_delete,))
con.commit()

print(cursor.rowcount, "record(s) deleted
  ↪ successfully.")
```

Output:

```
1 record(s) deleted successfully.
```

ID	NAME	AGE	GRADE
1	Pratyush	18	A+