

1. LOGIC BASED PROGRAM ON STRING/LIST/TUPLE/Dictionary

Code:

```
x = input("Enter a string: ")
s = ''
for i in range(len(x)-1,-1,-1):
    s+=x[i]
print(f"{s} is the reverse of {x}")
```

Output:

Enter a string: Hello everyone

enoyreve olleH is the reverse of Hello everyone

2. FLOOR DIVISION

Code:

```
x = float(input("Enter a number: "))
y = int(input("Enter the number for division: "))
print("Enter your choice")
print("1.Floor division")
print("2.Regular division")
choice = input("Enter choice: ")
if choice == "1":
    print(x//y)
elif choice == "2":
    print(x/y)
else:
    print("Enter a valid choice")
```

Output:

Enter a number: 78

Enter the number for division: -4

Enter your choice

1.Floor division

2.Regular division

Enter choice: 1

-20.0

3. OPERATORS IN PYTHON

Code

```
x = int(input("Enter a number: "))
y = int(input("Enter another number: "))
print("Choose the option")
print("1.Arithmetic operator")
print("2.Logical operator")
print("3.Comparison operator")
choice = input("Your choice: ")

if choice == "1":
    print(f"{x+y} if their sum")
    print(f"{x-y} if their diff")
    print(f"{x*y} if their prod")
    print(f"{x/y} if their quotient")
elif choice == "2":
    if x+y == 10 or x-y==5:
        print("the sum is 10 or the diff is 5")
    else:
        print("neither the sum is 10 nor the diff is
5")
elif choice == "3":
    if x>y:
        print(f"{x} if greater than {y}")
    elif y>x:
        print(f"{y} if greater than {x}")
    else:
        print("Both are equal")
else:
    print("Enter a valid option")
```

Output:

Enter a number: 3

Enter another number: 7

Choose the option

1.Arithmetic operator

2.Logical operator

3.Comparison operator

Your choice: 1

10 if their sum

-4 if their diff

21 if their prod

0.42857142857142855 if their quotient

4. NUMBER SYSTEM

CODE

```
def D_t_B():
    a=int( input('Enter Decimal Number: ') )
    q=a
    l=[]
    s=''
    while q>0:
        (q,r)=(q//2,q%2)
        l.append(r)
    l.reverse()
    for i in range(len(l)):
        s=s+str(l[i])
    return f'\t\t\t ({a}){chr(0x2081)}{chr(0x2080)} = ({s}){chr(0x2082)}\n'

def D_t_O():
    a=int( input('Enter Decimal Number: ') )
    q=a
    l=[]
    s=''
    while q>0:
        (q,r)=(q//8,q%8)
        l.append(r)
    l.reverse()
    for i in range(len(l)):
        s += str(l[i])
    return f'\t\t\t ({a}){chr(0x2081)}{chr(0x2080)} = ({s}){chr(0x2088)}\n'

def D_t_H():
    a=int( input('Enter Decimal Number: ') )
    q=a
    l=[]
    s=''
    while q>0:
        (q,r)=(q//16,q%16)
        l.append(r)
    l.reverse()
```

```

d={10: 'A',11: 'B',12: 'C',13: 'D',14: 'E',15: 'F'}
for i in range(len(l)):
    if l[i]>9:
        s += str(d[l[i]])
    else:
        s += str(l[i])
return f'\t\t\t ({a}){chr(0x2081)}{chr(0x2080)} = ({s}){chr(0x2081)}{chr(0x2086)}\n'

def Menu():
    while True:
        print('\t\t\t -----')
        print('\t\t\t DATA REPRESENTATION')
        print('\t\t\t -----')
        print('\t\t\t SELECT A OPTION:')
        print('\t\t\t -----')
        print('\t\t\t 1.Convert Decimal to Binary')
        print('\t\t\t 2.Convert Decimal to Octal')
        print('\t\t\t 3.Convert Decimal to Hexal')
        print('\t\t\t 4.Exit')
        z=int(input('Enter choice: '))
        print()
        if z == 1:
            print( D_t_B() )
        elif z == 2:
            print( D_t_O() )
        elif z == 3:
            print( D_t_H() )
        else:
            exit()

Menu()

```

OUTPUT

```
-----
DATA REPRESENTATION
-----
SELECT A OPTION:
-----
1.Convert Decimal to Binary
2.Convert Decimal to Octal
3.Convert Decimal to Hexal
4.Exit
Enter choice: 1
Enter Decimal Number: 344
(344)10 = (101011000)2

-----
DATA REPRESENTATION
-----
SELECT A OPTION:
-----
1.Convert Decimal to Binary
2.Convert Decimal to Octal
3.Convert Decimal to Hexal
4.Exit
Enter choice: 2
Enter Decimal Number: 55
(55)10 = (67)8

-----
DATA REPRESENTATION
-----
SELECT A OPTION:
-----
1.Convert Decimal to Binary
2.Convert Decimal to Octal
3.Convert Decimal to Hexal
4.Exit
Enter choice: 3
Enter Decimal Number: 29
(29)10 = (1D)16

-----
DATA REPRESENTATION
-----
SELECT A OPTION:
-----
1.Convert Decimal to Binary
2.Convert Decimal to Octal
3.Convert Decimal to Hexal
4.Exit
Enter choice: 4
>>>
```

5. SEARCHING

CODE

```
def Bin_asc():
    x=input('Enter Sorted List: ').split()
    #to input elements with spaces in between

    s=input('Enter Element to be Searched: ')
    print()
    l=0
    u=len(x)-1
    while l <= u:
        m = (l+u)//2
        if s == x[m]:
            print(f'Element {s} is present at index {m} of list')
            break
        elif s > x[m]:
            l = m+1
        else:
            u = m-1
    else:
        print('Element not in list')
    print()
def Bin_desc():
    | x=input('Enter Sorted List: ').split()
    | #to input elements with spaces in between

    s=input('Enter Element to be Searched: ')
    print()
    l=0
    u=len(x)-1
    while l <= u:
        m = (l+u)//2
        if s == x[m]:
            print(f'Element {s} is present at index {m} of list')
            break
        elif s > x[m]:
            u = m-1
```



```

        else:
            l = m+1
    else:
        print('Element not in list')
    print()

def Binary_Search():
    print('\t\t\t BINARY SEARCH')
    print('\t\t\t 1.Data in Ascending Order')
    print('\t\t\t 2.Data in Descending Order')
    a=int( input('Enter Choice: ') )
    print()
    if a==1:
        Bin_asc()
        print()
    elif a==2:
        Bin_desc()
        print()

def Linear_Search():
    x=input('Enter Sorted List: ').split()
    #to input elements with spaces in between

    s=input('Enter Element to be Searched: ')
    print()
    l=len(x)
    c=0
    y=[]
    if s in x:
        for i in range(l):
            if x[i] == s:
                c += 1
                y.append( str(i) )
        z=', '.join(y)
        print(f'Element {s} is present {c} time(s) at index {z} of list')

    else:
        print('Element not in list')
    print()

def Menu():
    while True:

        print('\t\t\t SEARCHING')
        print('\t\t\t 1.Binary Search')
        print('\t\t\t 2.Linear Search')
        print('\t\t\t 3.Exit')
        z=int( input('Enter Choice: ') )

```

```

print()
if z==1:
    Binary_Search()
    print()
elif z==2:
    Linear_Search()
    print()
else:
    exit()
Menu()

```

OUTPUT

```

SEARCHING
1.Binary Search
2.Linear Search
3.Exit
Enter Choice: 1

BINARY SEARCH
1.Data in Ascending Order
2.Data in Descending Order
Enter Choice: 1

Enter Sorted List: 1 1 2 2 2 3 3 4 5 6 6 7
Enter Element to be Searched: 2

Element 2 is present at index 2 of list

SEARCHING
1.Binary Search
2.Linear Search
3.Exit
Enter Choice: 1

BINARY SEARCH
1.Data in Ascending Order
2.Data in Descending Order
Enter Choice: 2

Enter Sorted List: 5 5 5 4 4 3 3 3 2 2 1 1 1
Enter Element to be Searched: 1

```

Element 1 is present at index 11 of list

SEARCHING

1.Binary Search

2.Linear Search

3.Exit

Enter Choice: 2

Enter Sorted List: 1 3 4 5 2 3 6 7 9 5 3 7

Enter Element to be Searched: 3

Element 3 is present 3 time(s) at index 1,5,10 of list

SEARCHING

1.Binary Search

2.Linear Search

3.Exit

Enter Choice: 3

>>>

6. SORTING

CODE

```
def Bubble():
    print('\t\t\t BUBBLE SORT')
    print('\t\t\t 1.Ascending Order')
    print('\t\t\t 2.Descending Order')

    a=int( input('Enter Choice: ') )
    print()

    x=[int(el) for el in input('Enter List: ').split()]
    print()
    l=len(x)
    for j in range(l-1):
        for i in range(l-1-j):
            if a == 1:
                if x[i] > x[i+1]:
                    x[i], x[i+1] = x[i+1], x[i]
            else:
                if x[i] < x[i+1]:
                    x[i], x[i+1] = x[i+1], x[i]
    print('Sorted List:', x)

def Insertion():
    print('\t\t\t INSERTION SORT')
    print('\t\t\t 1.Ascending Order')
    print('\t\t\t 2.Descending Order')

    a=int( input('Enter Choice: ') )
    print()
    x=[int(el) for el in input('Enter List: ').split()]
    print()
    l=len(x)
    for i in range(1, l):
        t = x[i]
        j = i-1
        if a == 1:
```

```

        while j >= 0 and t < x[j]:
            x[j+1] = x[j]
            j=j-1
    else:
        while j >= 0 and t > x[j]:
            x[j+1] = x[j]
            j=j-1
    x[j+1]=t
    print('Sorted List:', x)

def Selection():
    print('\t\t\t SELECTION SORT')
    print('\t\t\t 1.Ascending Order')
    print('\t\t\t 2.Descending Order')
    a=int( input('Enter Choice: ') )
    print()

    x=[int(el) for el in input('Enter List: ').split()]

    print()
    l=len(x)
    for i in range(l-1):
        for j in range(i+1, l):
            if a == 1:
                if x[i] > x[j]:
                    x[i], x[j] = x[j], x[i]
            else:
                if x[i] < x[j]:
                    x[i], x[j] = x[j], x[i]

    print('Sorted List:', x)

def Merge():
    print('\t\t\t MERGE SORT')
    print()

    x=eval(input('Enter List 1: '))
    y=eval(input('Enter List 2: '))
    z=[]
    j=0
    for i in range(len(x)):
        while y[j]<x[i]:
            z.append(y[j])
            j=j+1
        z.append(x[i])
    z.extend(y[j:])
    print('Sorted List in Descending Order:', z)

```

```

def Menu():
    while True:
        print('\t\t\t SORTING')
        print('\t\t\t 1.Bubble Sort')
        print('\t\t\t 2.Insertion Sort')
        print('\t\t\t 3.Selection Sort')
        print('\t\t\t 4.Merge Sort')
        print('\t\t\t 5.Exit')
        z=int( input('Enter Choice: ') )
        print()

        if z == 1:
            Bubble()
            print()
        elif z == 2:
            Insertion()
            print()
        elif z == 3:
            Selection()
            print()
        elif z == 4:
            Merge()
            print()
        else:
            exit()

Menu()

```

OUTPUT

```

SORTING
1.Bubble Sort
2.Insertion Sort
3.Selection Sort
4.Merge Sort
5.Exit
Enter Choice: 1

BUBBLE SORT
1.Ascending Order
2.Descending Order
Enter Choice: 1
Enter List: 1 2 6 4 3 6 2
Sorted List: [1, 2, 2, 3, 4, 6, 6]

SORTING
1.Bubble Sort
2.Insertion Sort
3.Selection Sort
4.Merge Sort
5.Exit
Enter Choice: 1

BUBBLE SORT
1.Ascending Order
2.Descending Order
Enter Choice: 2
Enter List: 2 8 4 2 1 5 3
Sorted List: [8, 5, 4, 3, 2, 2, 1]

SORTING
1.Bubble Sort
2.Insertion Sort
3.Selection Sort
4.Merge Sort
5.Exit
Enter Choice: 2

INSERTION SORT
1.Ascending Order
2.Descending Order
Enter Choice: 1

```

Enter List: 1 3 6 2 4 7 4

Sorted List: [1, 2, 3, 4, 4, 6, 7]

SORTING

- 1.Bubble Sort
- 2.Insertion Sort
- 3.Selection Sort
- 4.Merge Sort
- 5.Exit

Enter Choice: 2

INSERTION SORT

- 1.Ascending Order
- 2.Descending Order

Enter Choice: 2

Enter List: 2 3 9 1 5 7 8

Sorted List: [9, 8, 7, 5, 3, 2, 1]

SORTING

- 1.Bubble Sort
- 2.Insertion Sort
- 3.Selection Sort
- 4.Merge Sort
- 5.Exit

Enter Choice: 3

SELECTION SORT

- 1.Ascending Order
- 2.Descending Order

Enter Choice: 1

Enter List: 5 6 8 3 9 10 22

Sorted List: [3, 5, 6, 8, 9, 10, 22]

SORTING

- 1.Bubble Sort
- 2.Insertion Sort
- 3.Selection Sort
- 4.Merge Sort
- 5.Exit

Enter Choice: 4

MERGE SORT

Enter List 1: [1,2,3,4]

Enter List 2: [3,4,5,6,7]

Sorted List in Descending Order: [1, 2, 3, 3, 4, 4, 5, 6, 7]

SORTING

1.Bubble Sort

2.Insertion Sort

3.Selection Sort

4.Merge Sort

5.Exit

Enter Choice: 5

>>>

7. STRING HANDLING

CODE

```
s=str(input('enter a string:'))
l=len(s)

print('original string is',s)

#capitalizing alternate elements in a string
s2=''
for i in range(0,l,2):
    s2=s2+s[i]
    if i<(l-1):
        s2=s2+s[i+1].upper()
print('alternatively capitalized string is',s2)

#split a string on a specific characters
print(s.split(' '))

#reverse a string
print('').join(reversed(s))

#string slicing
string='python is a great language.'
print(string)
print(string[:6])
print(string[7:13])
print(string[0:-1:2])
|
```

OUTPUT

```
enter a string: 'python is a computer language'
original string is 'python is a computer language'
alternatively capitalized string is 'PyThOn iS A CoMpUtEr lAnGuAgE'
['python', 'is', 'a', 'computer', 'language']
'egaugnal retupmoc a si nohtyp'
python is a great language.
python
is a g
pto sagetlnug
>>>
```

8. LIST HANDLING

CODE

```
l = [1,2,3,4,5,6,7,7,87,7,7,7,7,7,7,73,3,3,3]

while 7 in l:
    l.remove(7)

for i in range(len(l)):
    l.append(i)

l.extend([321,234,432,345,543,456])

del(l[0])

l.insert(4,5)

print(l)
```

OUTPUT

```
[2, 3, 4, 5, 5, 6, 87, 73, 3, 3, 3, 0, 1, 2, 3, 4, 5, 6, 7, 8
, 9, 10, 321, 234, 432, 345, 543, 456]
>>>
```

9. TUPLE HANDLING

CODE

```
l=eval(input('Enter a list to convert into tuple: '))
t=tuple(l)

tl=len(t)
print('Length of tuple', tl)

tmax=max(t)
tmin=min(t)
print('Maximum value of tuple: ',tmax)
print('Minimum value of tuple: ',tmin)

n=int(input('Enter number to be counted from tuple: '))
tc=t.count(n)
print('The number ',n , 'has occurred ',tc , 'no. of times')
```

OUTPUT

```
Enter a list to convert into tuple: [1,3,5,7,6,6,9,33]
Length of tuple 8
Maximum value of tuple:  33
Minimum value of tuple:  1
Enter number to be counted from tuple: 6
The number  6 has occurred  2 no. of times
>>>
```

10. DICTIONARY HANDLING

CODE

```
n=int(input('Enter number of entries: '))
d={}
for i in range (n):
    key=int(input('Enter key: '))
    value=input('Enter value: ')
    d[key]=value
print('The dictionary is: ',d)

print('Length of dictionary is: ',len(d))

print(d.keys())
print(d.values())

d[4]='d'
print('New dictionary is: ',d)
```

OUTPUT

```
Enter number of entries: 3
Enter key: 1
Enter value: a
Enter key: 2
Enter value: b
Enter key: 3
Enter value: c
The dictionary is:  {1: 'a', 2: 'b', 3: 'c'}
Length of dictionary is:  3
dict_keys([1, 2, 3])
dict_values(['a', 'b', 'c'])
New dictionary is:  {1: 'a', 2: 'b', 3: 'c', 4: 'd'}
>>>
```

11. RANDOM

CODE

```
import random

numbers = range(11)
string = 'testString'
alpha_numeric = ''
for i in range(random.randint(0,20)):
    alpha_numeric += str(random.choice(numbers))
    if random.choice([True,False]):
        alpha_numeric += random.choice(string)

print(alpha_numeric)
```

OUTPUT

```
6S3g5n10g
>>>
```

12. FILE HANDLING (TEXT MODE)

--> removing all empty lines

CODE

```
with open('abc.txt','r') as f:
    s = f.read().split('\n')
    while '' in s:
        s.remove('')
    for i in range(len(s)):
        s[i]+='\n'

with open('abc.txt','w') as f:
    f.writelines(s)
```

OUTPUT

Initially:
file

handling

text

Finally:

file
handling
text

13. FILE HANDLING (BINARY MODE)

CODE

```
import pickle

with open('abc.dat','rb') as f:
    try:
        while True:
            s = pickle.load(f)
            print(s)
    except EOFError:
        print("End of file reached")
```

OUTPUT

```
bye
hello
how are you?
where are you?
bye
End of file reached
>>> |
```

14. FILE HANDLING (CSV)

CODE

```
import csv
with open('abc.csv','r') as f:
    reader = csv.reader(f,delimiter = ',')
    for row in reader:
        for item in row:
            print(item,end = ' ')
        print()
```

OUTPUT

```
Name Class
Someone 12
Nobody 11
>>>
```

15. USER DEFINED FUNCTION

code:

```
def avg_number(numList):  
    x=0  
    for i in numList:  
        x+=i  
    mean=x/len(numList)  
    print("The average is ",mean)  
numListStr = input("Enter numbers separated by space:").split()  
numList = [int(i) for i in numListStr]  
avg_number(numList)
```

output:

Enter numbers separated by space:90 80 75 94 21 47 82 56

The average is 68.125

16. MYSQL QUERIES

1: Creating a table with suitable fields

Code:

```
CREATE TABLE jobs (  
JOB_ID integer NOT NULL UNIQUE PRIMARY KEY,  
JOB_TITLE varchar(35) NOT NULL DEFAULT ' ',  
MIN_SALARY decimal(6,0) DEFAULT 8000,  
MAX_SALARY decimal(6,0) DEFAULT NULL  
);
```

Output:

```
mysql> CREATE TABLE jobs (  
-> JOB_ID integer NOT NULL UNIQUE PRIMARY KEY,  
-> JOB_TITLE char(10) NOT NULL DEFAULT ' ',  
-> MIN_SALARY decimal(6,0) DEFAULT 8000,  
-> MAX_SALARY decimal(6,0) DEFAULT NULL  
-> );  
Query OK, 0 rows affected (1.92 sec)
```

2: Entering 3 records at once

Code:

```
insert into jobs  
values(281,'Engineer',24435,291198),(991,'Engineer',246645,  
,291198),(721,'Engineer',287645,291348);
```

Output:

```
mysql> insert into jobs values(281,'Engineer',24435,291198),(991,'Engineer',246645,291198),(721,'Engineer',287645,291348);  
Query OK, 3 rows affected (0.14 sec)  
Records: 3 Duplicates: 0 Warnings: 0
```

empno	ename	job	mgr	hiredate	sal	comm	deptno
8369	SMITH	CLERK	8902	1991-12-18	800	NULL	20
8499	ANYA	SALESMAN	8698	1991-02-20	1600	300	30
8521	SETH	SALESMAN	8698	1991-02-22	1250	500	30
8566	MAHADEVAN	MANAGER	8839	1991-04-02	2985	NULL	20
8654	MOMIN	SALESMAN	8698	1991-09-28	1250	1400	30
8882	SHIAUNSH	MANAGER	8839	1991-06-09	2450	NULL	10
8698	BINA	MANAGER	8339	1991-05-10	2850	NULL	30
8888	SCOTT	ANALYST	8566	1992-12-09	3000	NULL	20
8839	AMIR	PRESIDENT	NULL	1991-11-18	5000	NULL	10
8844	KULDEEP	SALESMAN	8698	1991-09-08	1500	0	30
8886	ANOOP	CLERK	8888	1993-01-12	110000	NULL	20
8900	JATIN	CLERK	8698	1991-12-03	950	NULL	30
8902	FAKIR	ANALYST	8566	1991-12-03	3000	NULL	20
8934	MITA	CLERK	8882	1992-01-23	1300	NULL	10

1. Details of those employees whose annual salary is between 25000 and 40000.

SELECT * FROM empl where sal BETWEEN 2500 AND 4000;

empno	ename	job	mgr	hiredate	sal	comm	deptno
8566	MAHADEVAN	MANAGER	8839	1991-04-02	2985	NULL	20
8698	BINA	MANAGER	8339	1991-05-10	2850	NULL	30
8888	SCOTT	ANALYST	8566	1992-12-09	3000	NULL	20
8902	FAKIR	ANALYST	8566	1991-12-03	3000	NULL	20

2. Names of employees whose name contains 'A' as the 4th alphabet.

SELECT ename FROM empl WHERE ename like '___a%';

ename
ANYA
MAHADEVAN
SHIAUNSH
BINA
MITA

3. Name, Job and Salary of employees who do not have a manager.

SELECT ename, job, sal FROM empl WHERE mgr IS NULL;

ename	job	sal
AMIR	PRESIDENT	5000

4. Name, Salary and salary added with commision.

SELECT ename, sal, sal+comm FROM empl;

ename	sal	sal+comm
SMITH	800	NULL
ANYA	1600	1900
SETH	1250	1750
MAHADEVAN	2985	NULL
MOMIN	1250	2650
SHIAUNSH	2450	NULL
BINA	2850	NULL
SCOTT	3000	NULL
AMIR	5000	NULL
KULDEEP	1500	1500
ANOOP	110000	NULL
JATIN	950	NULL
FAKIR	3000	NULL
MITA	1300	NULL

5. Details of employees who earn more commission than their salaries.

SELECT * FROM empl WHERE comm>sal;

empno	ename	job	mgr	hiredate	sal	comm	deptno
8654	MOMIN	SALESMAN	8698	1991-09-28	1250	1400	30

ID	First_Name	Last_Name	User_ID	Salary
1	DIM	JOSEPH	JDIM	5000
2	jagganath	MISHRA	JNMISHRA	4000
3	SIDDHARTH	MISHRA	SMISHRA	8000
4	SHANKAR	GIRI	SGIRI	7000
5	GAUTAM	BUDDHA	BGAUTAM	2000

1. For record with ID=4 update record with last Name, User ID and Salary.

UPDATE employe SET

Last_Name='SAHUKAR',User_ID='skar',Salary=9000

WHERE ID=4;

ID	First_Name	Last_Name	User_ID	Salary
1	DIM	JOSEPH	JDIM	5000
2	jagganath	MISHRA	JNMISHRA	4000
3	SIDDHARTH	MISHRA	SMISHRA	8000
4	SHANKAR	SAHUKAR	skar	9000
5	GAUTAM	BUDDHA	BGAUTAM	2000

2. Modify the last name of employees to Gautam where salary<5000.

UPDATE employe SET Last_Name='Gautam' WHERE Salary;

ID	First_Name	Last_Name	User_ID	Salary
1	DIM	JOSEPH	JDIM	5000
2	jagganath	Gautam	JNMISHRA	4000
3	SIDDHARTH	MISHRA	SMISHRA	8000
4	SHANKAR	SAHUKAR	skar	9000
5	GAUTAM	Gautam	BGAUTAM	2000

3. Add column Email of data type VARCHAR to the table.

ALTER TABLE employe ADD(Email VARCHAR(30));

ID	First_Name	Last_Name	User_ID	Salary	Email
1	DIM	JOSEPH	JDIM	5000	NULL
2	jagganath	Gautam	JNMISHRA	4000	NULL
3	SIDDHARTH	MISHRA	SMISHRA	8000	NULL
4	SHANKAR	SAHUKAR	skar	9000	NULL
5	GAUTAM	Gautam	BGAUTAM	2000	NULL

4. Delete the employee record having first name as SIDDHARTH.

DELETE FROM employe WHERE First_Name='SIDDHARTH';

ID	First_Name	Last_Name	User_ID	Salary	Email
1	DIM	JOSEPH	JDIM	5000	NULL
2	jagganath	Gautam	JNMISHRA	4000	NULL
4	SHANKAR	SAHUKAR	skar	9000	NULL
5	GAUTAM	Gautam	BGAUTAM	2000	NULL

5. Modify the salary and increases it by 1000, for all who get salary less than 5000.

UPDATE employe SET Salary =Salary+1000 WHERE Salary<5000;

ID	First_Name	Last_Name	User_ID	Salary	Email
1	DIM	JOSEPH	JDIM	5000	NULL
2	jagganath	Gautam	JNMISHRA	5000	NULL
4	SHANKAR	SAHUKAR	skar	9000	NULL
5	GAUTAM	Gautam	BGAUTAM	3000	NULL

17. RECURSION

Code:

```
def pow(a,n):  
    if n==0:  
        return 1  
    else:  
        return a*pow(a,n-1)
```

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

```
def Menu():  
    print("Choose your option: ")  
    print("1.Power")  
    print("2.Factorial")  
    choice = int(input ("Enter choice: "))  
    if choice == 1:  
        base = int(input ("Enter base: "))  
        exponent = int(input ("Enter exponent: "))  
        print(pow(base,exponent))  
    elif choice == 2:  
        n = int(input ("Enter number: "))
```

```
        print(factorial(n))
    else:
        print("Enter valid input ")
        return Menu()

if __name__ == "__main__":
    Menu()
```

Output:

Choose your option:

1.Power

2.Factorial

Enter choice: 2

Enter number: 5

120

Choose your option:

1.Power

2.Factorial

Enter choice: 1

Enter base: 2

Enter exponent: 10

1024

18. STACKS

```
stack=[]
size=0
s=0
def push():
    global s,size
    if s==size:
        print('Over Flow')
    else:
        s=s+1
        e=int(input("ENTER DATA:"))
        stack.append(e)

def pop():
    global s
    top=-1
    if stack==[]:
        print('Under Flow')
    else:
        print("Element popped is:",stack[-1])
        del(stack[top])
        s=s-1

def display():
    if stack==[]:
        print('Under Flow')
    else:
        q=len(stack)
        for top in range (q-1,-1,-1):
            print(stack[top])

def menu():
    global size
    size=int(input('Enter the size of the stack:'))
    while True:
```

```

print("\t\t\t1. Push")
print("\t\t\t2. Pop")
print("\t\t\t3. Display")
print("\t\t\t4. Exit")
c=int(input("ENTER CHOICE:"))
if c==1:
    push()
elif c==2:
    pop()
elif c==3:
    display()
else:
    break
menu()

```

OUTPUT

```

Enter the size of the stack:2
1. Push
2. Pop
3. Display
4. Exit

ENTER CHOICE:1
ENTER DATA:1

1. Push
2. Pop
3. Display
4. Exit

ENTER CHOICE:1
ENTER DATA:2

1. Push
2. Pop
3. Display
4. Exit

```

ENTER CHOICE:1

Over Flow

1. Push
2. Pop
3. Display
4. Exit

ENTER CHOICE:2

Element popped is: 2

1. Push
2. Pop
3. Display
4. Exit

ENTER CHOICE:2

Element popped is: 1

1. Push
2. Pop
3. Display
4. Exit

ENTER CHOICE:2

Under Flow

1. Push
2. Pop
3. Display
4. Exit

ENTER CHOICE:1

ENTER DATA:1

1. Push
2. Pop
3. Display
4. Exit

ENTER CHOICE:3

1

1. Push
2. Pop
3. Display
4. Exit

ENTER CHOICE:2

Element popped is: 1

1. Push
2. Pop
3. Display
4. Exit

ENTER CHOICE:3

Under Flow

1. Push
2. Pop
3. Display
4. Exit

ENTER CHOICE:4

>>>

19. QUEUES

CODE

```
queue=[]
r=f=None

def enqueue():
    global queue,r,f
    if queue==[]:
        r=0
    else:
        r=r+1
    e=int(input("Enter Element:"))
    queue.insert(r,e)

def dequeue():
    global queue,r,f
    if queue==[]:
        return "Empty Queue"
    else:
        f=0
        print('Element to be removed is:', queue[f])
        del(queue[f])

def display():
    global queue,r,f
    if queue==[]:
        print("Empty Queue")
    else:
        q=len(queue)
        for i in range (q):
            print(queue[i],end=' - ')
```



```

def menu() :
    global queue,r,f
    queue=[]
    f=None
    while True:
        print("\t\t\tQUEUE OPERATIONS")
        print("\t\t\t1. Enqueue")
        print("\t\t\t2. Dequeue")
        print("\t\t\t3. Display")
        print("\t\t\t4. Exit")
        c=int(input("ENTER CHOICE:"))
        if c==1:
            enqueue()
        elif c==2:
            dequeue()
        elif c==3:
            display()

        else:
            break
menu()

```

OUTPUT

```

- - -
                                QUEUE OPERATIONS
                                1. Enqueue
                                2. Dequeue
                                3. Display
                                4. Exit

ENTER CHOICE:1
Enter Element:1

                                QUEUE OPERATIONS
                                1. Enqueue
                                2. Dequeue
                                3. Display
                                4. Exit

ENTER CHOICE:1
Enter Element:2

```

QUEUE OPERATIONS

1. Enqueue
2. Dequeue
3. Display
4. Exit

ENTER CHOICE:3
1-2-

QUEUE OPERATIONS

1. Enqueue
2. Dequeue
3. Display
4. Exit

ENTER CHOICE:2
Element to be removed is: 1

QUEUE OPERATIONS

1. Enqueue
2. Dequeue
3. Display
4. Exit

ENTER CHOICE:2
Element to be removed is: 2

QUEUE OPERATIONS

1. Enqueue
2. Dequeue
3. Display
4. Exit

ENTER CHOICE:3
Empty Queue

QUEUE OPERATIONS

1. Enqueue
2. Dequeue
3. Display
4. Exit

ENTER CHOICE:4

20. ART INTEGRATION PROJECT

CODE

```
import pymysql

import matplotlib.pyplot as plt

from tkinter import *

def create_table():

    db=pymysql.connect(host="localhost",user="root",passwd=rootpwd,db=database)

    cur=db.cursor()

    cur.execute("create table stu(Roll int,Name char(20),Class char(5),English int,
    Physics int, Chemistry int, Maths int, Computers int, Percentage char(6),Grade ch
    ar(2), Remark char(4));")

    db.commit()

    cur.close()

    db.close()

def add_record_screen():

    global mainframe

    mainframe.destroy()

    mainframe = Frame(root,width=1100,height=600,bg="#111")

    mainframe.grid_propagate(0)

    mainframe.pack()

def add_record():

    roll=int(rolle.get())

    name=namee.get()
```

```

clas=clase.get()

eng=int(enge.get())

mat=int(mate.get())

cs=int(cse.get())

chem=int(cheme.get())

phy=int(phye.get())

total = eng+mat+phy+chem+cs

perc = round(total/500 * 100,2)

if perc > 33 : rem = 'PASS'

else: rem = 'FAIL'

if perc > 90: grade = 'A1'

elif perc > 80: grade = 'A2'

elif perc > 70: grade = 'B1'

elif perc > 60: grade = 'B2'

elif perc > 50: grade = 'C1'

elif perc > 40: grade = 'C2'

elif perc > 33: grade = 'D'

else: grade = 'F'

perc = str(perc)+"%"


db=pymysql.connect(host="localhost",user="root",passwd=rootpwd,db=database)

cur=db.cursor()

cur.execute(f"insert into stu values({roll},\"{name}\",\"{clas}\",{eng},{phy}
,{chem},{mat},{cs},\"{perc}\",\"{grade}\",\"{rem}\");")

db.commit()

cur.close()

db.close()

```

```

rolle.delete(0, 'end')

namee.delete(0, 'end')

clase.delete(0, 'end')

enge.delete(0, 'end')

phye.delete(0, 'end')

cheme.delete(0, 'end')

mate.delete(0, 'end')

cse.delete(0, 'end')

```

```

Label(mainframe, bg="#111", fg="#fff", text='Roll No.').grid(row=1, column=1)

Label(mainframe, bg="#111", fg="#fff", text='Name').grid(row=2, column=1)

Label(mainframe, bg="#111", fg="#fff", text='Class').grid(row=3, column=1)

Label(mainframe, bg="#111", fg="#fff", text='English').grid(row=4, column=1)

Label(mainframe, bg="#111", fg="#fff", text='Physics').grid(row=5, column=1)

Label(mainframe, bg="#111", fg="#fff", text='Chemistry').grid(row=6, column=1)

Label(mainframe, bg="#111", fg="#fff", text='Mathematics').grid(row=7, column=1)

Label(mainframe, bg="#111", fg="#fff", text='Comuper Science').grid(row=8, column=1
)

```

```

rolle = Entry(mainframe)

rolle.grid(row=1, column=2)

namee = Entry(mainframe)

namee.grid(row=2, column=2)

clase = Entry(mainframe)

clase.grid(row=3, column=2)

enge = Entry(mainframe)

```

```
enge.grid(row=4,column=2)
```

```
phye = Entry(mainframe)
```

```
phye.grid(row=5,column=2)
```

```
cheme = Entry(mainframe)
```

```
cheme.grid(row=6,column=2)
```

```
mate = Entry(mainframe)
```

```
mate.grid(row=7,column=2)
```

```
cse = Entry(mainframe)
```

```
cse.grid(row=8,column=2)
```

```
Button(mainframe,text="Back",command=Menu).grid(row=9,column=1)
```

```
Button(mainframe,text="Submit",command=add_record).grid(row=9,column=2)
```

```
def display():
```

```
    global box
```

```
    db=pymysql.connect(host="localhost",user="root",passwd=rootpwd,db=database)
```

```
    cur=db.cursor()
```

```
    rows=cur.execute("select * from stu;")
```

```
    rec=cur.fetchall()
```

```
    records = """
```

```
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```

```
|Roll No. |Name                |Class   |English |Physics |Chemistry |
Maths     |CS                  |Percentage|Grade   |Remarks |
```

```
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```

```
"""
```

```
    for i in rec:
```

```
for j in range(len(i)):

    if j==0: records+="|"

    if j==1: records += "{0:<25}|".format(i[j])

    else:   records += "{0:<10}|".format(i[j])

records+="\n+-----+-----+-----+-----+-----+-----+-----+-----+\n"

cur.close()

db.close()

box.configure(state='normal')

box.insert('end', records)

box.configure(state='disabled')
```

```
def display_graph():

    db=pymysql.connect(host="localhost",user="root",passwd=rootpwd,db=database)

    cur=db.cursor()

    rows=cur.execute("select * from stu;")

    all_records=cur.fetchall()

    avg_eng=avg_phy=avg_chem=avg_maths=avg_cs=0

    for i in all_records:

        avg_eng+=i[3]

        avg_phy+=i[4]

        avg_chem+=i[5]

        avg_maths+=i[6]

        avg_cs+=i[7]
```

```
avg_eng/=rows
```

```
avg_phy/=rows
```

```
avg_chem/=rows
```

```
avg_maths/=rows
```

```
avg_cs/=rows
```

```
bg2=[avg_eng,avg_phy,avg_chem,avg_maths,avg_cs]
```

```
try: rr=int(roll_no.get())
```

```
except:
```

```
    print('Please enter roll no.')
```

```
    return
```

```
aa=f"select * from stu where roll='{rr}';"
```

```
rows=cur.execute(aa)
```

```
rec=cur.fetchall()
```

```
bg1=[]
```

```
x=["English","Physics","Chemistry","Mathematics","Computer Science"]
```

```
barWidth = 0.1
```

```
for i in rec:
```

```
    bg1.append(i[3])
```

```
    bg1.append(i[4])
```

```
    bg1.append(i[5])
```

```
    bg1.append(i[6])
```

```
    bg1.append(i[7])
```



```

r1 = [0,1,2,3,4]

r2 = [i + barWidth for i in r1]

plt.bar(r1,bg2,width=0.1,label="Class Average")

plt.bar(r2,bg1,width=0.1,label="Student")

plt.xlabel('group', fontweight='bold')

plt.xticks([r + barWidth for r in range(len(bg1))], ['English', 'Physics', 'Chemistry', 'Maths', 'CS'])


plt.legend()

plt.show()

cur.close()

db.close()


def Menu():

    global mainframe,box,roll_no

    mainframe.destroy()

    mainframe = Frame(root,width=1100,height=600,bg="#111")

    mainframe.grid_propagate(0)

    mainframe.pack()

    Label(mainframe,text="Menu",bg="#111",fg="#fff",font=('serif',25)).grid(row=1,column=1)

    Button(mainframe,text="Create Table",command=create_table).grid(row=2,column=1)

    Button(mainframe,text="Add Record",command=add_record_screen).grid(row=3,column=1)

    Button(mainframe,text="Display All Records",command=display).grid(row=4,column=1)

    Button(mainframe,text="Display Bar Graph Student Wise",command=display_graph).grid(row=5,column=1)

```

```

Button(mainframe,text="Exit",command=quit).grid(row=6,column=1)

roll_no = Entry(mainframe)

roll_no.grid(row=7,column=1)

box=Text(mainframe,width=137,height=25,bg='#333',fg='#fff',state='disabled')

box.grid(row=8,column=1)


def connect(a1,a2):

    global database,rootpwd

    rootpwd = a1

    database = a2

    Menu()


def connect_screen():

    e1 = Entry(mainframe,show="*")

    e2 = Entry(mainframe)

    Label(mainframe,text="Enter root@localhost Password",bg="#111",fg="#fff").grid(
row=1,column=1)

    e1.grid(row=1,column=2)

    Label(mainframe,text="Enter name of database to be used",bg="#111",fg="#fff").g
rid(row=2,column=1)

    e2.grid(row=2,column=2)

    Button(mainframe,text="Submit",command=lambda: connect(e1.get(),e2.get()) ).gri
d(row=3,column=1,columnspan=2)


root = Tk()

root.geometry('1100x600')

mainframe = Frame(root,width=1100,height=600,bg="#111")

mainframe.grid_propagate(0)

```

```
mainframe.pack()
```

```
connect_screen()
```

```
root.mainloop()
```

OUTPUT:



Roll No.	Name	Class	English	Physics	Chemistry	Maths	CS	Percentage	Grade	Remarks
11	Divyam	XII-D	99	99	99	99	99	99.0%	A1	PASS
12	Aditya	XII-D	99	99	99	99	99	99.0%	A1	PASS
13	Harsh Vardhan	XII-D	99	99	99	99	99	99.0%	A1	PASS
13	Anirudhha	XII-D	90	90	90	90	90	90.0%	A2	PASS
15	Padmalochan	XII-D	95	95	95	95	95	95.0%	A1	PASS
16	Aditya Singh	XII-D	95	95	99	92	100	96.2%	A1	PASS

tk

Roll No.

7

Name

Person X

Class

XII-D

English

30

Physics

40

Chemistry

30

Mathematics

22

Compuer Science

25

Back

Submit

tk

Menu

Create Table

Add Record

Display All Records

Display Bar Graph Student Wise

Exit

Roll No.	Name	Class	English	Physics	Chemistry	Maths	CS	Percentage	Grade	Remarks
1	Divyam	XII-D	99	99	99	99	99	99.0%	A1	PASS
2	Aditya	XII-D	99	99	99	99	99	99.0%	A1	PASS
3	Harsh Vardhan	XII-D	99	99	99	99	99	99.0%	A1	PASS
3	Anirudhha	XII-D	90	90	90	90	90	90.0%	A2	PASS
5	Padmalochan	XII-D	95	95	95	95	95	95.0%	A1	PASS
6	Aditya Singh	XII-D	95	95	99	92	100	96.2%	A1	PASS
7	Person X	XII-D	30	40	30	22	25	29.4%	F	FAIL

Figure 1

