HOSPITAL DATABASE

Date:

Q a) Create a patient class to store basic info of patient like name, room no , registration no and amount payed

b) Create a method in class which prints Patients data

c) Sort name of patients alphabetically.

First making a class patient

```
Class Patient ( ):
    def __init__(self, name, room, regist, amount):
            self.name = name
            self.room = room
            self.regist = regist
            self.amount = amount
```

=> constructor is used to give arguments in a class, Here we used constructor to store data and also used to store data in object/instance variables.

of patient in patient class. The attributes of patient class is name, room, regist, amount.

(b)

```
def patients_info (self):
    print ("NAME" \t\t"ROOM NO" \t\t"REGISTRATION NO"
            \t\t "AMOUNT PAYED")

    Print (f"{self.name} \t\t {self.room} \t\t
            {self.regist} \t\t {self.amount} ")
```

→ The above method will print the info of any patient.

(c)

Now making objects of patient class

```
patient1 = patient("Umer", "E-9", "ABCD123", "15000")

patient2 = patient ("Sara", "E-10", "ABCD124", "17000")

patient3 = patient("Ahmed", "E-11", "ABCD125", 18000)

N = [patient1, patient2, patient3]
```

Now sorting names alphabetically

```
def bubble-Sort (a):
    n=len(a)
    for i in range (n):
        for j in range (n-i-1):
                            .name           .name
            If a [j] > a [j+1]:
                a[j], a[j+1] = a [j+1], a[j]
    Return a

N = bubble-Sort (N)
```

Date:

BANK DATABASE:-

=> Develop a system which can perform following basic banking related tasks

a) customer account could be created with name, NIC, account number and initial balance. All such attributes should be placed in a class

b) Balance of any customer could be updated and also a function of transaction

c) customer data could be sorted namewise and balance wise.

a)

```
# First making a class bank
class abcdbank( ):
    def --init-- (self, name, CNIC, account, balance):
            self.name = name
            self.CNIC = CNIC
            self.account = account
            self.balance = balance
```

=> constructor is used to give arguments in a class and used to store data in object / instance variables. Here we used constructor to store data in customer's account. The attributes of customer class are name, CNIC, account, balance.

b)

```
    def updatebalance (self, update):
        self.balance = self.balance +update
```

=> The above method will update the balance of customer

```
def transaction(self, amount):
    self.balance = self.balance - amount
```

(C)

```
# Now making objects of class bank

m1 = abcd bank("Umer", "4210148133883", "ABCD112", 15000)

m2 = abcd bank("Sara", "4210148144993", "ABCD115", 17000)

m3 = abcd bank("Wania", "42151-48133997", "ABCD118", 10000)

n = [m1, m2, m3]
b = [m1, m2, m3]
```

Now sorting name of customers

```
def bubble-sort(a):
    n=len(a)
    for i in range(n):
        for j in range(n-i-1):
            If a[j].name>a[j+1].name:
                a[j], a[j+1]=a[j+1], a[j]

    return a

n= bubble-sort(n)
```

Now presenting data of customers name wise

```
def list-name-wise( ):
    print("LIST NAME WISE")
    print("NAME\t\t CNIC\t\t ACCOUNT NO\t\t AMOUNT")
    print("------------------------------------------")

    for i in range(3):
        ~~print(f"{self.name}\t\t{self.CNIC}\t\t~~
              ~~{self.account}\t\t{self.amount~~
        print(f"{n[i].name}\t\t {n[i].CNIC}
              \t\t {n[i].account }\t\t{n[i].amount}")
    print("------------------------------------")
```

⇒ The above method will display the data of customers name wise.

Now Sorting balance of customers.

```
def bubble-sort1(a):
    n=len(a)
    for i in range(n):
        for j in range(n-i-1):
            If a[j].balance < a[j+1].balance:
                a[j],a[j+1] = a[j+1], a[j]

    return a

b = bubble-sort1(b)
```

Now presenting data of customers balance wise.

```
def list-balance-wise ( ):
    print ("LIST   BALANCE WISE")
    print ("NAME \t\t CNIC \t\t ACCOUNTNO \t\t
            BALANCE ")
    print (" - - - - - - - - - - - - - - - - - - - - ")
    for i in range (3):
        print (f"{b[i].name} \t\t {b[i].CNIC} \t\t
              {b[i].account}\t\t{b[i].balance}")
        print (" - - - - - - - - - - - - - - - - - - - "
```

(a)

Date:

Create an employee class to store basic information of an Employee like name, pay and job.

(b) Write a method in Employee Class to increase the salary of a person with desired percentage

(c) Create a sub-class of Employee named Manager which replaces the inherited method to increase the to increase the salary of a person by additional 10%.

(a)

```
Class Employee( ):
    def __init__ (self, name, pay, job):
        self.name = name
        self.pay = pay
        self.job = job
```

Constructor is used to give arguments in a ~~list~~ object. and used to store data in object/instance variables. Here we used constructor to store the data of employee. The attributes of class employee are name, pay and job.

(b)

```
def increase_salary (self, percentage):

    increment = self.pay / 100 x percentage
    self.pay = self.pay + increment
```

=> The above method will increase the salary of a person to desired percentage.

TASK: 01

Date:

Make a database of students and search for a specific roll no.

```
# First Creating class of Student
Class student( ):
    def __init__ (self, name, dept, sec, CGPA):
        self.name=name
        self.dept=dept
        self.sec=sec
        self.CGPA=CGPA

    def info (self):
        Print ("NAME\t\t DEPT\t\t SEC\t\tCGPA")
        Print (f"{self.name}\t\t{self.dept}\t\t{self.sec}
             \t\t {self.account} ")

#  Now making objects
rollno1 = student ("Umer", "Electrical", "D", 3.83 )

rollno2 = student ("Ahmed", "software", "D", 3.95 )

#  Now searching for a specific roll no
rollno1.info ( )
```

Date:

TASK: 02

Q. Implement loop based approach to create 10 entries of objects in a list.

```
# first making class of student
class student( ):
    def --init-- (self, name, rollno, dept, CGPA):
            self.name = name
            self.rollno = rollno
            self.dept = dept
            self.CGPA = CGPA

    def info (self):
        Print ("NAME \t\t ROLLNO \t\t DEPT \t\t CGPA")
        Print (f"{self.name} \t\t {self.rollno} \t\t
                {self.deft} \t\t {self.CGPA} ")

a = list(range (5))
b = list (range (5))
list = a+b

# Now implementing loop based approach to
  create 10 entries in a list
for i  in range(10):
    Print(f" Data of {i+1} student ")
    List [i] = student(input ("Name: "), input ("rollno: "),
                 input ("Dept" ), input("CGPA"))
```

Scanned with CamScanner