Homework of python

Name: Mohammad Shoaib «Mohammadi»

Class Department: IT

1) Creat a class called Person with attribute name and age. Creat an object of the class and Print its attributes.

```
class Person:
    def __init__(self,age,name)
        self.name=name
        self.age=age
P=Person("shoaib",12)
Print(P.name)
Print(P.age)
```

2) add a method called greet to the Person class that Print a greeting message including the Person's name? class Person:

```
    def __init__(self,name):
        self.name=name
    def greeting(self):
        Print(f"Hello how are you {self.name}")
info=Person("mohammad shoaib")
info.greeting()
```

تاريخ: / /

موضوع: _____

3) Create a class called car with attributes make, model, and year. include a method to print out the car's details.

```
class car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year

    def display(self):
        Print(self.make, self.model, self.year)

info_car = ("Amirecan", "Benz", 2016)
info_car.display()
```

4) Create class circle with a method that to compute the area. initilize the class with the radius.

```
class circle:
    def compute(self, r):
        Pi = 3.14
        A = Pi*r**2
        Print(A)
P = circle()
Print(P.compute(12))
```

تاريخ: / /

موضوع:

5) : create a class Rectangle with methods to compute the area perimeter.initilize the class with the length and width.

```
class Rectangle:
    def compute(self, hight, Widght):
        h = height
        W = Wight
        A = 2*(h*W)
        Print(F"the rectangle are of {A}")
result = Rectangle()
result.compute(12,43)
```

تاريخ: / /

موضوع:

. inheritance and Polymorphism Exercise.

6) Create a base class Animal with a method speak. Creat tow derived classes Dog and Cat that override the speak method.

```
class Animal:
    def speak(self):
        print("Dog is bark")
class Dog(Animal):
    def speak(self):

class Animal:
    def speak(self):
        print("Animal have different sound")

class Dog(Animal):
    def speak(self):
        print("Dog is bark")
class Cat(Animal):
    def cat(self):
        print("cat is meow")

obj = Cat()
obj.speak()
x = Dog()
x.speak()
```

7- Creat a base class With a method area. Creat a derived classes square an Triangle that implement the area method.

```
class Shape:
    def area(self):
        pass
class Square(Shape):
    def __init__(self, side):
        self.side = side
    def area(self):
        return self.side ** 2
class Triangle(Shape):
    def __init__(self, base, height):
        self.base = base
        self.height = height
```

```
def area(self):
    return self.base*
square = Square(side
triangle = Triangle(base,
def area(self):
    return self.
```

8- Creat a class Employee with attribute name and salary, Creat a derived class manger With an additional attribute department.

```
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
    def show_info(self):
        print(self)
class Manger(Employee):
    def __init__(self, overtime, name, salary):
        super().__init__(name, salary)
        self.overtime = overtime
    def __str__(self):
        return str(self.overtime) + " " + self.name
p = Manger(12, "shoaib", 500)
p.show_info()
```

9- Creat a base class Method vehicle with a method drives. Creat a derived classes Bike and Truck that override the drive method.

```
class vehicle:
    def drive(self):
        Print("car have different drives")

class Bike(vehicle):
    def drive(self):
        Print("Bicycle ride with Pedal and tire")

class Truck(vehicle):
    def drive(self):
        Print("Truck drive by ashtrinc an Energy")

dr=Bike()
dr.drive()
```

10- Creat a base class Birds with a method fly. Creat a drive classes with Eagle and Penguin override the fly method in Penguin to indicate that Penguin cannot fly.

```
class Birds:
    def method(self):
        Print("Birds have different flys")
class Eagle(Birds):
    def method(self):
        Print("Eagle can fly")
class Penguin(Birds):
    def method(self):
dsj=Penguin()Print("Penguin can not fly")
```

تاريخ: / /

موضوع: _____

Encapsulation and Abstraction:

1: creat a class account with private attribut Balances. Provide public method to dispositand withdraw money.

```
class Account:
    def __init__(self, initize_balance=0):
        if initiz_balance<0:
            raise ValueError("Initize balance cannot negat
        self._balance=initial_balance
    def desposit(self, amount)
        self._blance+=amount
        print(f"Deposited {amount}. New balance {self._bal
    def withdraw(self, amount):
        self.balance-=amount
        Print(f"withdraw {amount}, new balace{ self.bala
    de
    if
    def get_balance(self):
        return self._balance
if __name__=="__main__":
    account=Account(100)
    Print("Initize balance{amount.get_balance()})
    account.deposite(50)
    print"money is after depost{account.get_balanc
    account.withdraw(30)
    Print("Balance after withdraw {amount._get_bal
```

تاريخ: / /

موضوع:

12- Creat a class Book With Private attribute title, author, Page, Provide a Public method to get and set these attribute.

```
class Book:
    def __init__(self, title, author, Page):
        self.__title = title          # Private
        self.__author = author        # Private
        self._Page = Page             # Protected

    def display(self):
        Print(self.__title, self.__author, self._Page)

P = Book("Encupsulation", Amir, 12)
P.display()
```

تاریخ: / /

موضوع: _____

13: create a class laptop with private attribute brand, modual and price. Provide a method to apply dicount and amethod to display laptop method.

```
class laptop:
    def __init__(self, brand, model, Price):
        self._brand = brand
        self.__model = model
        self.__Price = Price
    def set_Price(self, dicount):
        if discount == 1233      #dicount
            self.__Price = self.__Price(self._Price*[illegible]
f = laptop("Dell", "2018", 2000)
f.set_Price(1233)
f.display
```

تاريخ: / /

موضوع:

M: Create a class Bank account with private attribute account_number and balace. Provide a method to dePosite withdraw and check the balace.

```
class BankAccount:
    def __init__(self, balace):
        self.balance=balance
    def Save_money(self):
        amount1= float(input("Enter the amount to be stored : "))
        self.balance+=amount
        Print("the amount store it: ", amount))
    def withdraw(self):
        amount= Float(input(" Enter the amount to be withdrawn"))
        if self.balance>=amount:
            self.balance-=amount
            Print("your withdraw", amount)
        else:
            Print("the amount is not enough")
    def display(self):
        Print("Nat avilible balance is ", self.balace )
S=BankAccount(13)
S.save_money()
S.withdraw()
S.display
```

تاریخ: / /

موضوع: _____

15: Create a class student with private attribute name, grade and age. Provide a method to get and set these attribute and a method to display the student's detail.

```
class Student:
    def __init__(self, name, grade, age):
        self.__name = name  # private
        self.__grade = grade
        self.__age = age
    def display(self):
        print(self.__name, self.__grade, self.__age)
info = Student("Mustafa", 12, 24)
info.display()
```

تاريخ: / /

موضوع: _____

Q6:- Create a class library with attribute name and books (a list of book) to add and remove book.

```
class Book
    def __init__(self, title, author):
        self.title = title
        self.author = author

    def __str__(self):
        return f"{self.title} by {self.author}"

class library:
    def __init__(self, name):
        self.name = name
        self.books = []

    def add_book(self, book):
            self.books.append(book)
            Print(f"Added: {book}")

    def remove_book(self, title):
            for book in self.books:
                if book.title == title:
                    self.books.remove(book)
                    Print(f"Removed: {book}")

    def list_books(self):
        for self
        for book in self.books:
            Print(book)
if __name__ == "__main__":
        library = library("city library")
        book1 = Book("Python", ahamd")
        library.list_book()
```

تاريخ: / /

موضوع: _____

17- create a class school with attribute name and student (list of student). create a method to add and remove student.

```
class Student:
    def __init__(self, name, grade):
        self.name = name
        self.grade = grade
    def __str__(self):
        return f"{self.name}, {self.grade}"

class School:
    def __init__(self, name):
        self.name = name
        self.students = []
    def add_student(self, student):
        self.students.append(student)
        Print(f"Added {student}")
    def remove_student(self, name):
        for student in self.students:
            if students.name == name
                self.students.remove(student)
                Print(f"Removed : {student}")
    def list_of_Student(self):
        for student in self.students:
if __name__ == "__main__": print(student)
    school = School("Naman")
    stud = Student("Ali", "10th")
    school.add_student(stud)
```

تاريخ: / / موضوع: _____

18- create a class called employee with attribute name and employee (a list of Employee). Provide a method to add and remove employee.

```
class Employee:
    def __init__(self, name, Position):
        self.name = name
        self.Position = Position

    def __str__(self):
        return f"{self.name}, {self.Position}"

class company:
    def __init__(self, name):
        self.name = name
        self.employees = []

    def add_employee(self, EPloyee):
        self.employee.append(employee)
        Print(f"Added: {employee}")
    def remove_employee(self, name):
        for employee in self.employees:
            if employee.name = name
            self.employees.remove(employee)
            Print(f"Removed: {employee}")
compa = company("cisco")
empl = Employee("Shoaib", "iT Manger")
compa.add_employee(empl)
```

تاريخ: / /

موضوع: _____

20- create a class Zoo with attribute name and animal (arist of animal). Create an object to add and remove animal.

```
class Animal:
    def __init__(self, name, species)
        self.name=name
        self.species=species
    def __str__(self):
        return f"{self.name}, {self.species}"
class Zoo:
    def __init__(self,name):
        self.name=name
        self.animals=[]
    def add_animal(self,animal):
        self.animals.append(animal)
        print(f"Added:{animal}")
    def remove_animal(self):
        for animal in self.animals:
            if animal.name=name:
                self.animals.remove(animal)
                print(f"Removed {animal}")
Z=Zoo("Zabul")
ani= Animal("cow", "cow")
Zoo.add_animal(animal)
```

تاريخ: / /

موضوع: _____

9.1- Create a class Ticket for a movie theater with attribute movie_name, set_number, Price Provide a method to display ticket detial and apply discount.

```
class Ticket:
    def __init__(self, movie_name, set_number, Price):
        self.movie_name = movie_name
        self.set_number = set_number
        self.Price = Price

    def apply_discount(self):
        discount_amount = (dicount * 0.01)
        self.Price = discount_amount

t = Ticket("jawan")
t = Ticket("jawan", 123, 2000)
t.apply_dicount(30)
t.display()
```

تاریخ: / /

موضوع: _____

22- create a class shopping card with method to add, remove and display item. Each item should be an object of a class item with attribute name and price.

```
class Item:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def __str__(self):
        return {self.name}, {self.price}

class ShoppingCart:
    def __init__(self):
        self.items = []
    def add_item(self, item):
        self.item.append(item)
        Print(f"Added {item}")
    def remove_item(self):
        for item in self.item:
            if items.name == name:
                self.items.remove(item)
    def display_item(self):
        for item in self.items:
            Print(item)
```

تاريخ: / /

موضوع:

23:- Create a class Resturant with attribute name and menu (a list of item object). Provide a method to add and remove item from the Menu.

```
class Item
class Resturant:
    def __init__(self, name, Price):
        self.name=name
        Self. Price=Price

    def __Str__(self):
        return({self.name}{sef.Price?})

class Resturant:
    def __init__(self, name):
        self.name=name
        Self.menu=[]
    def add_item(Self, item):
        self.item = item
        self.menu.append(item)
    def remove_item(self,name):
        for item in self.menu:
            if item.name==name
                set' self.menu.remove(item)
X=Resutrant("RandoZ")
item1=Item("Burger", 20)
Resuturant.add_item(item1)
```

Q- Creat a class flight with the attribute flight number, destination and Passenger(a list of passenger) Provid a method to add and remove Passenger

```
class Passenger:
    def __init__(self, flight_number,
```

```
class Passenger:
    def __init__(self,name, age):
        self.name=name
        self.age=age
    def __str__(self):
        return {self.name}, {self.age}
class flight(Passenger):
    def __init__(self, flight_number, distination):
        self.flight_number=flight_number
        self.distination=distination
        self.Passengers=[]
    def add_Passenger(self,name):
        self.nam
        self.Passenger.append(name)
    def remove_Passenger(self,name):
        if name in self.Passenger:
            self.Passenger.remove(nam)
obj= flight(123,"Canada")
obj.add:Passenge("Shoaib")
```

تاريخ: / / موضوع: Gui Application

36- create a class counter APP that use tkinter to create a simple Gui with the increament an decreament.

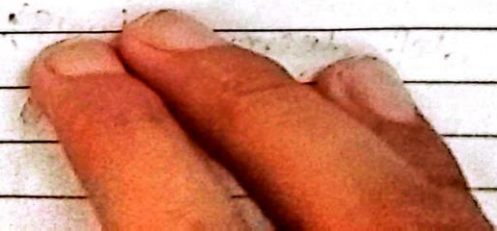
```
import tkinter as tk
class CounterAPP:
    def __init__(self,root):
        self.root=root
        self.root.title("Counter Application")
        self.counter=0
        self.label = tk.Label(root, text=str(self.counter),
         font=("Arial",48))
        self.label.pack(pady=10)
        self.incremen_Button=tk.Button (root,
          text=str(
          text="increment", command= self.increment )
        self.increment.pack(side=tk.Left,padx=10)
        self.decrement_Button=Button(root,
         text="decrement", command =self.decrement)
         self.decrement_Button.pack(side=tk.Right,padx=10
    def increment(self):
        self.counter+=1
        self.update_label()
    def decrement(self):
        self.counter-=1
        self.update_label()
```

تاريخ: / /

موضوع:

```
def update_label(self):
    self.label.config(text=str(self.counter))

root
if __name__ == "__main__":
    root = tk.Tk()
    app = CounterApp(root)
    root.mainloop()
```



تاريخ: / /

موضوع:

37- creat a class ToP APP that use Tkinter to do list GUI where user can add and remove task.

```
import tkinter as tk
from tkinter import *
class ToPAPP:
    def __init__(self, root)
        self.root = root
        self.root.title("To Do list APPlication")
        self.task_entry = tk.Entry(root, width=40)
        self.task_entry.pack(pady=10)
        self.add_button = tk.Button(root, text="Add task",
        command = self.add_task.
```