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

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

    def __str__(self):
        return f"{self.nam}, {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(empYee)
            Print(f"Removed: {employee}")

compa = Company("Cisco")
empl = Employee("shaaib", "IT Manger")
compa.add_employee(empl)
```

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:
            print(student)

if __name__ == "__main__":
    school = School("Name")
    stud1 = Student("Ali", "10th")
    school.add_student(stud1)
```

16:- 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 book in self.books:
            print(book)

if __name__ == "__main__":
    library = library("city library")
    book1 = Book("python", anand)
    library.list_book()
```

15: Create a class student with private attributes 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("Nischal", 12, 24)
info.display()
```

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):
        amount=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 withdraw"))
        if self.balance>=amount:
            self.balance-=amount
            Print("your withdraw", amount)
        else:
            Print("the amount is not enough")

    def display(self):
        Print("Not avilible balance is", self.balace)

S=BankAccount(13)
S.save_money()
S.withdraw()
S.display
```

13: create a class laptop with private attributes brand, model and Price. Provide a method to apply dicount and a method 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, discount):
        if discount == 1233   #dicount
            self.__Price = self.__Price(self.__Price*
f = laptop("Dell", "2018", 2,000)
f.set_Price(1233)
f.display
```

12- creat a class Book with Private attribute title, autho, 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           # Private

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

p = Book("Encapsulation", Amir, 12)
p.display()
```

## Encapsulation and Abstraction

1. Create a class account with private attributes Balances. Provide public methods to display, withdraw money.

```
class Account:
    def __init__(self, initial_balance=0):
        if initial_balance < 0:
            raise ValueError("Initial balance cannot be negative")
        self.__balance = initial_balance
    def deposit(self, amount):
        self.__balance += amount
        print(f"Deposited {amount}, new balance is {self.__balance}")
    def withdraw(self, amount):
        self.__balance -= amount
        print(f"Withdrew {amount}, new balance is {self.__balance}")
    def get_balance(self):
        return self.__balance
if __name__ == "__main__":
    account = Account(100)
    print("Initial balance:", account.get_balance())
    account.deposit(50)
    print("Balance after deposit:", account.get_balance())
    account.withdraw(30)
    print("Balance after withdraw:", account.get_balance())
```

9- creat a base class 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 ashtrins an Energy")

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

10- creat a base class Birds with a method fly. Creat a drived 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):
        Print("Penguin can not fly")

obj=Penguin()
obj.method()
```

7- Creat a base class With a method area. creat a derived classes square and 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
```

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 + " " + str(self.salary)
p = Manger(12, "shoaib", 500)
p.show_info()
```

8. inheritance and Polymorphism Exercise.

8) 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("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()
```

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.comPut(12,43)
```

Fig 3

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", 2010)
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))
```

# Homework of python

Name: Mohammad Shoaib «Mohammadi»

Department: IT

class

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()
```