السؤال الأول

Create a class called persone with attributes name and age. Create an object of the class and print its attributes.

```
class person:

    def __init__(self, name, age):
        self.name = name
        self.age = age

Mohammad = person("Mohammad", 30)

print(Mohammad.name)
print(Mohammad.age)
```

سوال ۳ / / تاریخ: موضوع:

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 print_details(self):
        return f"the car's details {self.make}
                {self.model}, {self.year}"

corola = car("corola-2000", "Amerecan-model", 2015)
print(corola.print_details())

#  نمایش به شکل دیکشنری
print(corola.__dict__)
```

Create a class circle with a method to compute the area. Initialize the class with the radious.

```
class circle:
    def __init__(self, radius):
        self.radius = radius

    def compute(self):
        return 3.1415 * (self.radius**2)

circle = circle(5)
print(circle.comput())
```

add a methed called great to the persone class that prints a greating message including the person's name.

```
class person:

    def __init__(self, name):
        self.name = name

    def Great(self):
        return F" Hello, {self.name} How are you "

Ahmad = person("Ahmad")
print(Ahmad.Great())
```

creat a base class Animal with a method s
creat two derived classes Dog and c
that override the speak method.

```
class Animal:
    def speak(self):
        pass

class Dog(Animal):
    def speak(self):
        return "Woof"

class Cat(Animal):
    def speak(self):
        return "Meue"

dog = Dog()
cat = Cat()

print("the dog is speak", dog.speak())
print("the cat is speak", cat.speak())
```

7 dir

create a base class shape with a method area. create derive classes square and Triangle that implemented the area method.

```
class shape:
    def area(self):
        pass
class Square(shape):
    def __init__(self, length):
        self.length = length
    def area(self):
        return self.length ** 2
class Triangle(shape):
    def __init__(self, base, heigth):
        self.base = base
        self.heigth = heigth
    def area(self):
        return 0.5 * self.base * self.heigth
squar = Square(4)
triangle = Triangle(4,4)
print("the area Square is {self.area()}")
print("the area Triangle is {self.area()}")
```

سؤال رقم 5

تاريخ: / /

موضوع:

creat a class Rictangle with method to compute the area ~~dkd~~ and perimeter. Initialize the class with the length and width.

```
class Rectangle:

    def __init__(self, Length, width):
        self.Lngth = Length
        self.width =width

    def comput_area (self):
        return self.Length * Self.width

    def comput-perimeter (self):
        return 2 * self.width + 2 * self.Lenght
```

```
rectangle=Rectangle (6, 8)
print (rectangle. comput_area)
print (rectangle. Comput_primeter)
```

create a base class vehicle with a method drive. create a drived classes Bike and Truck that override the drive method.

```
class vehicle:
    def drive(self):
        pass

class Bike(vehicle):
    def drive(self):
        print("Driving a Bike")

class Truck(vehicle):
    def drive(self):
        print("Driving a Truck")

truck = Truck()
bike = Bike()

bike.drive()
truck.drive()
```

Q.N.

create a base class Bird with a method fly. create derived classes Eagle and penguin, override the fly method in penguin to indicate that penguin cannot fly

```
class Bird:
    def fly(self):
        pass

class Eagle(Bird):
    def fly(self):
        print("Eagle is flying in the skey")

class penguin(Bird):
    def fly(Bird):
        print("penguin is ~~not~~ cannot flying")

eagle = Eagle()
penguin = penguin()

eagle.fly()
penguin.fly()
```

سوال ۴

create a class Employee with attribute name and salary. create a drive class manager with with and adidtional attribute department.

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

class Manager (Employee):
    def __init__(self, name, Lname, salary):
        Supper().__init__(name, salary)
        self.Lname = Lname

Hamid = Manager("Hamid", "Arifi", 30000)

print(Hamid.name)
print(Hamid.Lname)
print(Hamid.salary)
```

```
def withdraw(self, amount):
    if amount > 0 and amount <= self.balance:
        self.balance -= amount
        print(f"withdraw {amount}.
              New balance: {self.balance}")
    elif amount > self.balance:
        print("Insufficient funds.")
    else:
        print("Invalid withdrawal amount")

my_account = Account(50000)

my_account.deposite(40000)
my_account.withdraw(20000)
```

تاريخ: / / موضوع:

create a class book with private attributes title, author, and pages. private public method to get and set these attributes.

```
class Book:
    def __init__(self, title, author, pages):
        self._title = title
        self._author = author
        self._pages = pages

    def get_title(self):
        return self._title

    def set_title(self, title):
        self._title = title

    def get_author(self):
        return self._author

    def set_author(self, author):
        self._author = author
```

create a class Account with private attributes balance. private public methods to depsite and withdraw money.

```
class Account:
    def __init__(self, balance):
        self.balance = balance

    def deposite(self, amount)
        if amount > 0:
            self.bulance += amount
            print(f"Deposited {amount} new balance
                                {self.balance}")
        else:
            print(f"Invalid deposite amount.")
```

create a class laptop with private attributes brand, model, and price. private a method to apply a discount and a method to display laptop details.

```
class Laptop:
    def __init__(self, barand, model, price):
        self.__brand = barand
        self.__price = price
        self.__model = model
        self.__price = price

    def applay_descount(self, discount_persent):
        if 0 <= disscount_persent <= 100:
            self.__price *= (1 - discount_persent/100)
            print(f"Discount of {discount_persent}
                  % applayed.
                  New price: {self.price}")
        else:
            print("Invalid discount persent")
```

```
def display_detailes(self):
    print(f"barands: {self._banrand}\n
        model: {self._model}\n price: {self._price}")


laptop = Laptop("Apple", "MSpro", 30000)

laptop.display_details()

laptop.applay_discount(10)

laptop.display_details()
```

```
    def get_page(self):
        return self._pages

    def set_page(self, pages):
        self._pages = pages

book = Book("OOP-python", "Ali", 180)
print("Title", book.get_title())
print("author", book.get_author())
print("pages", book.get_pages())

book.set_title("list in python")
book.set_author("Ahmad")
book.set_page(277)
print("updated pages: ", book.get_pages())
```

سؤال

create a class student with private attribuit name, grade, and age. private method to get and set these attribuite and a method to display the student's details.

```
class Student:
    def __init__(self, name, grade, age):
        self.name = name
        self.grade = grade
        self.age = age

    def get_name(self):
        return self.name

    def get_grade(self):
        return self.grade

    def get_age(self):
        self.
        return self.age

    def set_name(self):
        self.name = name
```

create a class Bank Account with private attribu
account_number and balance private method t
deposite, withdraw, and check the balance.

```
class BankAcount:
    def __init__(self, acount_no, intialize_Ba)
        self.acount_number = acount_no
        self._bdance = intialize_Ba)

    def deposite (self, amount):
        if amount > 0:
            self.balance += amount
        else:
            print("Invalid deposite amount")

    def withdraw (self, amount):
        if amount >0 and amount <= self._bal
            self._balance -= amount
        else:
            print("Invdid")
```

```
def check_balance(self):
    print(f"Acoun_no:{self._acount_number}")
    print(f"current balance:{self._balance}")

acount=BankAcount("12346579233",100000)
acount.check_balance()
acount.deposite(200000)
acount.withdraw(50000)
acount.check_balance()
acount.withdraw(20000)
```

Create a class school with attributes name and student (a list of student object) private method to add and remove students.

```
class School:
    def __init__(self, name):
        self.name = name
        self.student = []

    def add_student(self, st_name):
        self.student.append(st_name)
        print(f"student {st_name} added")

    def remove_student(self, st_name):
        if st_name in self.student:
            self.student.remove(st_name)
            print(f"student {st_name} removed")
        else:
            print("no student in school")
```

```
    def set_grade(self):
        self.grade = grade

    def set_age(age self):
        self.age = age

    def display_details(self):
        print(f"name: {self.name}")
        print(f"grade: {self.grade}")
        print(f"age: {self.age}")

student = Student("Mohammad", 90, 18)

print(student.get_name())
print(student.get_grade())
print(student.get_age())

student.set_name("Mohammad Emran")
student.set_grade(82)
student.set_age(22)

student.display_details()
```

سؤال تاسع

create a class Library with attributes name and books (a list of Book objects). private method to add and remove books.

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

    def add_books(self, Name_Book):
        self.books.append(Name_Book)
        print(f"Book {Name_Book} added")

    def remove_Book(self, Name_Book):
        if Name_Book in self.books:
            self.books.remove(Name_Book)
            print(f"Book {Name_Book} removed")
        else:
            print("Invalid book in library")
```

```
def member details(self):
    if self.members:
        for mem in self.members:
            print(f"- {mem}")
    else:
        print("This Team No member")

team = Team("Javanan")

team.add_member("Hamid")
team.add_member("Ali")
team.add_member("Rahman")

team.member_details()

team.remove_member("Ahmad")
team.remove_member("Ali")

team.member_details()
```

```
def school_details(self):
    if self.student:
        for ST in self.students:
            print(f"-{ST}")
    else:
        print("No student in School")

school=School("Rahman Baba")

school.add_student("Ahmad")
schoo.add_student("javad")

school.remove_student("Ali")
school.remove_student("Javad")

school.school_details()
```

```
def print_book(self):
    if self.books:
        print("book in library")
        print("book in library")
        for book in self.books:
            print(f"- {book}")
    else:
        print("No Book in Library")

Library = Library("Albirony")
print(library.name)

library.add_books("Python")
Library.add_books("C++")
Library.add_books("Jawa")

library.remov_book("C++")
Library.remove_book("C")

Library.print_books()
```

سوال ٤

create a class Zoo with attributes name and animals (a list of animals object). provide method to add and remove animals.

```
class Zoo:
    def __init(self, name):
        self.name = name
        self.animals = []

    def add_animals(self, A_name):
        self.animals.append(A-name)
        print("Animal {A_name} added")

    def remove-animals(self, A_name):
        if A-name in self.animals:
            self.animals.remove(A_name)
            print(f"animal {A_name} remove)
        else:
            print("this animal No in Zoo")
```

Create a class company with attributes name and employees (a list of employee object). provide methods to add and remove employees.

```
class Company:
    def __init__(self, name):
        self.name = name
        self.employee = []

    def add_employee(self, emp_name):
        self.employees.append(emp_name)
        print(f"employee {emp_name} added")

    def remove_employee(self, emp_name):
        if emp_name in self.employees:
            self.employee.remove(emp_name)
            print(f"employee {emp_name} remove")
        else:
            print(f"No {emp_name} in company")
```

تاريخ: / / موضوع:

create a class Team with attributes name and member (a list of person object). private methods to add and remove member.

```
class Team:
    def __init__(self, name):
        self.name = name
        self.members = []

    def add_member(self, Me_name):
        self.members.append(Me_name)
        print(f"member {Me_name} added")

    def remove_member(self, Mem_name):
        if Mem_name in self.members:
            self.member.remove(Mem_name)
            print(f"member {Mem_name} removed")
        else:
            print(f"No {Mem_name} in Team")
```

يتبع ...

Q2 حل

تاريخ: ... موضوع: ...

Create a class Log with methods to write error messages to a log file.

```
class Log:
    def __init__(self, log_file):
        self.log_file = log_file

    def write_message(self, message):
        print(f"Error writing to Log file: {message}")

    def log_Error(self, message):
        formated_message = f"error: {message}\n"
        self.write message(formated_massge)

    def Log_writing(self, message):
        formated_message = f"writing {message}")
        self.write_message(formated_message)

log = Log("application.log")
log.log_error("This is an error massag.")
log.writing("This is warning message")
```

~~def animal_det~~

```
def Zoo_details(self)
    if self.animals:
        for anl in self.animals:
            print(f"- {anl} ")
    else:
        print("No animal in the Zoo")

Zoo = Zoo("Kabul_Zoo")

Zoo.add_animals("Wolf")
Zoo.add_animals("Laine")
Zoo.add_animals("Tiger")

Zoo.animals_details()

~~Zoo.animal~~
Zoo.remove_animals("Wolf")
Zoo.remove_animal("elephen")
```

```
def employee_details(self):
    if self.employee:
        for emp in self.employee:
            print(f"- {emp}")
    else:
        print(f"No {self.employee} in Team")

company = Company("Oracle")

company.add_employee("Jamshid")
company.add_employee("Mohammad")
company.add_employee("Mahmod")

company.remove_employee("Jamshid")

company.employee_details()
```

route a class config that reads configuration setting from a file and provides methods to access these setting.

```
class config:
    def __init__(self, config_file):
        self.config_file = config_file
        self.settings = self.load_config()

    def load_config(self):
        print(F"{self.config_file} not found")

    def get(self):
        return f"in this program error {self.config_file}"

    def set(file):
        self.setting = []
        self.save_config()

    def save_config(self)
        print(F"ENT CF: {self.config_file}")
```

```
config = Config("config.json")
print("Database Host:" Config.get())
print(config.get())
print(config.set())
```

سوال لسيت يلم

create a class fileManager with methods to read from and write to a file

```
class FileManger:
    def __init__(self, Manager_name):
        self.manager_name = Manager_name

    def write_to_file(self, content):
        print(f"the file is writing by {manger_name}")

    def read_to_file(self):
        print(f"the file is reading by {manger_name}")

file_mangrer = FileManager("Manager_Ahmed")

file_manger.write_to_file("This is an Example content.")

~~content = file_manger.read_from_file()~~

content = file_manger.read_from_file()
if content is not None:
    print("content read from file:")
    print(content)
```

```
def generate_report(self):
    try:
        data = self.read_file()
        report = self.process_data(data)
        return report
    except Exception as e:
        print("Error generator: {e}")
        return None

def process_data(self, data):
    numlines = len(data)
    report = f"RQ1 Numbers like: {num_line}"
    report += "file content:" + "\n".join(data)
    return report

report = Report("data.txt")

report_content = report.generat_report()
if report_content:
    print("Generated Report")
    print(report_content)
```

تاريخ: / / موضوع:

Create a class Ticket for a movie theatre with attributes movie_name, seat_number, and price. provide methods to display ticket details and apply discount.

```
class Ticket:
    def __init__(self, movie_name, movie_no, price):
        self.movie_number =
        self.movie_name = name
        self.movie_number = movie_no
        self.price = price

    def display_details(self):
        details = (
            f"movie name: {self.movie_name}\n"
            f"set no : {self.movie_number}\n"
            f"prime $: {self.price}"
        print(details)
```

p.151

Create a class Report that generates a report from data in a file. provide method to handle exception if the file does not exist or cannot be read.

```
import os
class Report:

    def __init__(self, file_p)
        self.file_path= file_p

    def read_File(self):
        if not so.path.isfile(self.file_path):
            raise FileNotFoundError(f" file
            does not exist:{self.file_path}")
        try:
            with open(self.file_path, 'r') as file:
                return file.readlines()
        except IOError as e:
            raise IoError(f"Error reading file:{e}")
```

Create a class shopping cart with method to add, remove, and display items. Each item should be an object of a class items with attributes name and price

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

    def __repr__(self):
        return f"{self.name} {self.price}"


class Shopping Cart:
    def __init__(self):
        self.items = []

    def add_items(self, items):
        if instance(item, Item):
            print(f"(Add {item} to the cart")
        else:
            print("cannot added")
```

```
def remove_item(self, item_name):
    for item in self.items:
        if item.name == item_name:
            self.items.remove(item)
            print(f"Removed {item} from cart")
            return item_name
    print(f"Item {item_name} not found")

def display_items(self):
    if not self.items:
        print("the cart is empty")
    else:
        print("shopping cart item")
        for item in self.items:
            print(item)

item1 = Item("apple",
item2 = Item("apple",
item3 = Item("apple",

cart = shoppingcart()
cart.add_itet(item1)
cart.apply_item()
cart.remove_item()
```

```
def apply_discount(self, discount_per):
    if 0 <= discount_per <= 100:
        discount_amount = (discount_per / 100) * self.p
        self.price -= discount_amount

        print(f"Discount apply: {discount_per}%")
    else:
        print("Invalid")

ticket = Ticket("The Matrix", "A12", 15.00)

ticket.display_details()

ticket.apply_discount(10)

ticket.display_details()
```

```
def Resturant_details(self):
    if self.costumer:
        print("costumer in Resturant")
        for cos in self.costumer:
            print(f"- {cos}")
    else:
        print("No costumer in Resturant")

resturant = Resturant("Resturant Talaei")

resturant.add_costumer("Ahmad")
resturant.add_costumer("Ali")

resturant.remove_costumer("Ali")
resturant.remove_costumer("Rahman")

resturant.Resturant_details()
```

سوال ششم

Create a class fright with attributes flight_No destination, and passenger (a list of person ob). Provide method to add and remove passengers

```
class Flight:
    def __init__(self, Cligl_name, destation):
        self.fligl_name = flight_name
        self.destation = distation
        self.passanger = []

    def add_passenger(self, passenger_name):
        self.passenger.uppen(passenger_name)
        print(f"passenger {passenger_name} added")

    def remove_passenger(self, passenger_name):
        if passenger_name in self.passengel:
            self.passenger.remove(passenger_name)
            print(f"passenger {passenger_name} removed")
        else:
            print("No passenger in flight")
```

سوال بیست و هفت

تاریخ

موضوع:

reate a class Restudant with attributes name nd menu (a list of Item objects). provide ethods to add and remove item from teh menu.

```
class Resturant:
    def __init__(self, name):
        self.name = name
        self.costumer = []

    def add_costumer(self, costomer_name):
        self.costume.append(costumer_name)
        print(f" costumer1 {costumer_name} added ")

    def remove_costumer(self, costumer_name)
        if costumer_name in self.costumer
            self.costumer.remove(costumer_name)
            print(f"{costumer_name} removed ")

    else:
        print("No costumer in Resturant")
```

سوال ۱ / تاریخ: / موضوع:

create a class counterApp that use tkinter to create a simple counter GUI with increment and decrement buttons.

```
import tkinter as tk

class counterApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Counter App")

        self.counter = 0
        self.label = tk.Label(root, text=
                self.counter, font=("Arial", 24))
        self.label.pack(pady=20)

        self.increment_button = tk.Button(root,
            text="increment", command=self.increment)
        self.increment_button.pack()
```

```
    def flight_details(self):
        if self.passenger:
            print("passenger in flight")
            for pas in self.passengers:
                print(f"-{pas}")
        else:
            print("No passenger in flight")

flight = Flight("MaidanHavaie", "To Kabul")

flight.add_passenger("Ahmad")
flight.add_passenger("Ali")

flight.remove_passenger("Ali")
flight.remove_passenger("Rahman")

~~flight.remove~~

flight.flight_details()
```