opps_assignment

July 28, 2024

[34]:
```python
#bank account with attributes account_num,account_holder
class bank:
    def __init__(self,account_number,account_holdername,intial_balance):
        self.account_number=account_number
        self.account_holdername=account_holdername
        self.balance=intial_balance

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

        else:
            print("invalid")

    def withdraw(self,amount):
        if amount<0 and amount<self.balance:
            self.balance-=amount
            print(amount)

        else:
            print("insufficient balance")

    def get_balance(self):
        return self.balance
```

[35]:
```python
sony=bank(2322,"sony",43443)
```

[36]:
```python
sony.get_balance()
```

[36]: 43443

[37]:
```python
sony.deposite(1)
```

```
1
```

[38]:
```python
sony.get_balance()
```

[38]: 43444

[1]:
```
#employee management with attributes employee_id,name,and salary

class employee:
    def __init__(self,employee_id,name,salary):
        self.employee_id=employee_id
        self.name=name
        self.salary=salary

    def bonus(self):
        bonus_percentage=0.1
        yearly_bonus=self.salary*bonus_percentage
        return yearly_bonus
    def display_employee_details(self):
        print("employee id :",self.employee_id)
        print("name :",self.name)
        print("salary :",self.salary)
```

[2]:
```
el=employee("J.Naresh","Naresh",50000)
```

[3]:
```
el.bonus()
```

[3]: 5000.0

[4]:
```
el.display_employee_details()
```

```
employee id : J.Naresh
name : Naresh
salary : 50000
```

[5]:
```
e2=employee("J.Sony","Sony",250000)
```

[6]:
```
e2.bonus()
```

[6]: 25000.0

[7]:
```
#vechicle rent with attributes rent_vechicle,return_vechicle

class vehiclerentsystem:
    def __init__(self):
        self.avalible_vechicles=[]

    def rent_vehicle(self,vehicle):
```

```
        if vehicle_type in self.avalible_vechicles :
            self.avalible_vechicle.remove(vechicle)
            return vehicle,"rented successfully"
        else:
            return vehicle,"sorry vehicle is not avaliable"
    def return_vehicle(self,vehicle):
        self.avalible_vechicles.append(vehicle)
        return vechicle,"returned successfully"

    def display_avaliable_vechiles(self):
        if self.avalible_vechicles:
            return "available vehicles:"+",".join(self.avaliable_vehicels)
        else:
            return "no vehicles avaliable for rent"
```

[11]:
```
rental_system=vehiclerentsystem()
```

[12]:
```
rental_system.available_vehicles=["car","bike","scooter"]
```

[14]:
```
rental_system.available_vehicles
```

[14]:
```
['car', 'bike', 'scooter']
```

[27]:
```
#library with attributes avaliable_books,borrow_books
class book:
    def __init__(self,title,available):
        self.title=title

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

    def add_book(self,book):
        self.books.append(book)

    def borrow_book(self,book_title):
        for book in self.books:
            if book.title==book_title and book.available:
                book.available=False
                return "book",book_title,"has been borrowed"
            return "book not available for borrowing"

    def displaly_available_books(self):
            available_books=[book for book in self.books if book.available]
```

```
            if available_books:
                for book in available_books:
                    print("title",book.title)

            else:
                print("no avaliable books in the library")
```

[28]:
```
l1=library()
```

[29]:
```
b1=book("batascience",True)
b2=book("datastructures",True)
l1.add_book(b1)
l1.add_book(b2)
```

[31]:
```
l1.displaly_available_books()
```

```
title batascience
title datastructures
```

[32]:
```
l1.borrow_book("datascience")
```

[32]:
```
'book not available for borrowing'
```

[11]:
```
#shape with attributes length width,height to calculate the area and perimeter

class shape:
    def __init__(self,length,width,height):
        self.length=length
        self.width=width
        self.height=height

    def calculate_area(self):
        if self.height:#if the shape is 3D
            area=2*(self.length*self*width+self.length*self.height+self.
 ↪width*self.height)

        else:#if the shape is 2D
            area=self.length*self.width

        return area
    def calculate_perimeter(self):
        if self.height:#if the shape is 3D
            return "perimeter calculation not applicable of 3D shapes"
        else:#if the shape is 2D
            perimeter=2*(self.length+self.width)
            return perimeter
```

[12]:
```
rectangle=shape(2,5,0)
```

[13]:
```
rectangle.calculate_perimeter()
```

[13]: 14

[57]:
```
#email with attribute to send the mail

class email:
    def __init__(self,sender,recipient,subject):
        self.sender=sender
        self.recipient=recipient
        self.subject=subject

    def send_mail(self):
        print("email sent from:",self.sender)
        print("email sent to:", self.recipient)
        print("subject:",self.subject)
        print("email sent successfully!")

    def display_details(self):
        print("sender:",self.sender)
        print("recipient:",self.recipient)
        print("subject:",self.subject)
```

[58]:
```
email1=email("sonyjarupula81@gamil.com","nareshjarupula2000@gemail.
 ↪com","meeting reminder")
```

[59]:
```
email1.send_mail()
```

```
email sent from: sonyjarupula81@gamil.com
email sent to: nareshjarupula2000@gemail.com
subject: meeting reminder
email sent successfully!
```

[60]:
```
email1.display_details()
```

```
sender: sonyjarupula81@gamil.com
recipient: nareshjarupula2000@gemail.com
subject: meeting reminder
```

[13]:
```
class products:
    def __init__(self,name,price,quantity):
        self.name=name
        self.price=price
        self.quantity=quantity
```

```
class inventory:
    def __init__(self):
        self.products=[]

    def add_products(self,products):
        self.products.append(products)

    def update_producut_quantity(self,product_name,new_quantity):
        for products in self.products:
            if products.name==product_name:
                products.quantity=new_quantity
                break

    def display_avaliable_products(self):
        print("avaliable products")
        for products in self.products:
            print("product:",products.name,"price:",products.price,"quantity:
 ↪",products.quantity)
```

[14]:
```
i1=inventory()
```

[15]:
```
p1=product("laptop",10000,10)
p2=product("phone",5000,20)
i1.add_products(p1)
i1.add_products(p2)
```

[16]:
```
i1.display_avaliable_products()
```

```
avaliable products
product: laptop price: 10000 quantity: 10
product: phone price: 5000 quantity: 20
```

[17]:
```
i1.update_producut_quantity("laptop",100000)
```

[18]:
```
i1.display_avaliable_products()
```

```
avaliable products
product: laptop price: 10000 quantity: 100000
product: phone price: 5000 quantity: 20
```

[65]:
```
#student with attributes student_id ,grades,name

class student:
    def __init__(self,student_id,name,grades):
        self.student_id=student_id
        self.name=name
        self.grades=grades
```

```
    def calculate_average_grade(self):
        if len(self.grades)==0:
            return "no grades avaliable"
        average_grade=sum(self.grades)/len(self.grades)
        return average_grade

    def dispaly_student_details(self):
        print("student id:",self.student_id)
        print("name:",self.name)
        print("grades:",self.grades)
        average_grade=self.calculate_average_grade
        print("average grade:",self.average_grade)
```

[66]:
```
sony=student("J.sony","jarupula sony",[85,90,88,92,87])
```

[67]:
```
sony.grades
```

[67]:
```
[85, 90, 88, 92, 87]
```

[68]:
```
sony.calculate_average_grade()
```

[68]:
```
88.4
```

[69]:
```
#social media with attributes add_post,display _post

class socialmedia:
    def __init__(self,username):
        self.username=username
        self.posts=[]

    def add_post(self,post_content):
        self.posts.append(post_content)
        print("post added successfully!")

    def display_posts(self):
        print("post for",self.username,":")
        for index,post in enumerate(self.posts,start=1):
            print("post",self.index,":",post)

    def search_posts_by_keyword(self,keyword):
        found_posts=[post for post in self.posts if keyword in post]
        if  found_posts:
            print("posts containig the keyword'",keyword,"':")
            for index,post in enumerate(found_posts,start=1):
                print("post",self.index,":",post)
```

```
        else:
            print("no posts found containing the keyword'",keyword,"'.")
```

[70]:
```
profile1=socialmedia("sony123")
```

[71]:
```
profile1.add_post("excited for the weekend")
```

```
post added successfully!
```

[36]:
```
class todolist:
    def __init__(self):
        self.tasks={}

    def add_task(self,task,due_date):
        self.tasks[task]=due_date


    def mark_as_complete(self,task):
        if task in self.tasks:
            del self.tasks[task]
            return "task",task,"marked as completed"
        else:
            return "task not found in the todo list"

    def display_pending_tasks(self):
        if self.tasks:
            print("pending tasks:")
            for task,due_date in self.tasks.items():
                print("tasks:",task,"due date:",due_date)
        else:
            print("nopending tasks in todolist ")
```

[37]:
```
sony_list=todolist()
```

[38]:
```
sony_list.add_task("assignment","25-06-2024")
```

[40]:
```
sony_list.display_pending_tasks()
```

```
pending tasks:
tasks: assignment due date: 25-06-2024
```

[42]:
```
sony_list.add_task("opps","27-07-2024")
```

[43]:
```
sony_list.display_pending_tasks()
```

```
pending tasks:
tasks: assignment due date: 25-06-2024
tasks: opps due date: 27-07-2024
```

[ ]: