Experiment 1 Q1.py

```python
def Greeting():
    name = input("Enter your name: ")
    div = input("Enter your division: ")
    roll = int(input("Enter your Roll No.: "))
    print(f"Hello, {name}, Division {div}, Roll No {roll}.")

Greeting()
```

Experiment 1 Q2.py

```python
import math

def CalcArea():
    print("1: Rectangle\n2: Circle\n3: Triangle")
    Selection = int(input("Choose a shape: "))
    if (Selection == 1):
        l = int(input("Enter length: "))
        b = int(input("Enter breath: "))
        print(f"Area of Rectangle: {l * b}")

    elif (Selection == 2):
        r= int(input("Enter radius: "))
        print(f"Area of Circle: {math.pi*r*r}")

    elif (Selection == 3):
        s1 = int(input("Enter length of first side: "))
        s2 = int(input("Enter length of second side: "))
        s3 = int(input("Enter length of third side: "))
        semi = (s1 + s2 + s3)/2
        area = math.sqrt(semi*(semi-s1)*(semi-s2)*(semi-s3))
        print(f"Area of Triangle: {area}")

CalcArea()
```

Experiment 1 Q3.py

```python
def Salary():
    BS = int(input("Enter Basic Salary: "))
    DA = 0.7*BS
    TA = 0.3*BS
    HRA = 0.1*BS
    print(f"Gross Salary is: {BS + DA + TA + HRA}")

Salary()
```

Experiment 1 Q4 .py

```python
def Calci():
    a = int(input("Enter first number: "))
    b = int(input("Enter second number: "))
    print("1:Add\n2:Subtract\n3:Multiply\n4:Divide")
    choice = int(input("Enter choice: "))
    if(choice == 1):
        print(f"Sum: {a+b}")
    elif(choice == 2):
        print(f"Difference: {a-b}")
    elif(choice == 3):
        print(f"Product: {a*b}")
    elif(choice == 4):
      if(b == 0):
       print("Cannot divide by 0")
      else:
        print(f"Division: {a/b}")


Calci()
```

Experiment 2 Q1.py

```python
def Task():
    tasks = []
    while True:
        print("\n1. Add Task\n2. Remove Task\n3. Update Task\n4. View Tasks\n5. Exit")
        choice = int(input("Choose an option: "))
        if choice == 1:
            task = input("Enter task: ")
            tasks.append(task)
        elif choice == 2:
            task = input("Enter task to remove: ")
            tasks.remove(task)
        elif choice == 3:
            idx = int(input("Task index to update: "))
            tasks[idx] = input("Enter updated task: ")
        elif choice == 4:
            print("Tasks:", tasks)
        elif choice == 5:
            break
Task()
```

Experiment 2 Q2.py

```python
CET = {"shiven", "anirudh", "kushag"}
JEE = {"pratham", "anirudh", "shiven"}
NEET = {"kushag", "anirudh", "animesh"}


AllStudents = CET.union(JEE, NEET)
print("Students in any exam:", AllStudents)

Common = CET.intersection(JEE, NEET)
print("Students in all exams:", Common)

CETnotJEE = CET.difference(JEE)
print("Students in CET but not JEE:", CETnotJEE)

JEEnotCETandNEET = JEE.difference(CET,NEET)
print("Students in JEE but not CET and NEET:", JEEnotCETandNEET)
```

Experiment 2 Q3.py

```python
students = {
    "Anirudh": {"grade": "A", "attendance": 90}, "Shiven": {"grade": "B", "attendance": 75}
}

students["Yash"] = {"grade": "A-", "attendance": 0}


students["Anirudh"]["grade"] = "A"
students["Shiven"]["attendance"] += 5

del students["Anirudh"]

print("Student Records:")
for name, record in students.items():
    print(f"{name}: Grade - {record['grade']}, Attendance - {record['attendance']}%")
```

Experiment 3 Q1.py

```python
def EvenOrOdd():
    n = int(input("Enter number: "))
    if(n%2 == 0):
        print("Number is even")
    else:
        print("Number is odd")
EvenOrOdd()
```

Experiment 3 Q2.py

```python
def Factorial():
    n = int(input("Enter number: "))
    for i in range(2,n):
        n=i*n
    print(n)
Factorial()
```

Experiment 3 Q3.py

```
1   def IsPrime():
2       n = int(input("Enter number: "))
3       for i in range(2,int(n**0.5)+1):
4           if(n%i==0):
5               print("Number is not prime")
6               return
7       print("Number is prime")
8
9   IsPrime()
```

Experiment 3 Q4.py

```python
def Calci():
    def add(a,b): return a+b
    def subtract(a,b): return a-b
    def divide(a,b): return a/b
    def multiply(a,b): return a*b

    a = float(input("Enter first number: "))
    b = float(input("Enter second number: "))
    print("1: Add\n2: Subtract\n3: Multiply\n4: Divide")
    choice = int(input("Enter choice: "))

    if(choice == 1):
        print("Result: ", add(a,b))

    if(choice == 2):
        print("Result: ",subtract(a,b))

    if(choice == 3):
        print("Result: ", multiply(a,b))

    if(choice == 4):
        if(b==0):
            print("Cannot divide by 0")
        else:
            print("Result: ", divide(a,b))

Calci()
```

Experiment 4 Q1.py

```
1  def Names():
2      length = int(input("Enter word length: "))
3      with open(r"C:\Users\aniru\OneDrive\Desktop\Python experiments QB solution\Names.txt", "r") as f:
4          for line in f.readlines():
5              for word in line.split():
6                  if len(word) == length:
7                      print(word)
8  Names()
```

Experiment 4 Q2.py

```
1    #.exe wala chutiyapa
```

Experiment 5 Q1.py

```python
try:
    a = int(input("Enter first number: "))
    b = int(input("Enter second number: "))
    result = a/b
    print(f" a divided by b gives {result}")
except ValueError:
    print("Enter integers only")
except ZeroDivisionError:
        print("You cannot divide by 0")
finally:
    print("code executed")
```

Experiment 5 Q2.py

```python
import logging

# Set up logging configuration
logging.basicConfig(level=logging.DEBUG, #Capture all logs from DEBUG level and above
                    format='%(asctime)s - %(levelname)s - %(message)s')
def divide(a, b):
    logging.debug(f"Trying to divide {a} by {b}")
    if b== 0:
        logging.error("Cannot divide by zero!")
        return None
    result = a/b
    logging.info(f"Divison successful: {a} / {b} = {result}")
    return result

def main():
    logging.info("Pregram started.")

    #Valid division
    divide(10,2)
    #Invalid division (by zero)
    divide(10,0)
    logging.info("Program ended.")

if __name__ == "__main__":
    main()
```

Experiment 6 Q1.py

```python
class Participant:
    def __init__(self, name, email):
        self.name = name
        self.email = email

class Organiser:
    def __init__(self, name, contact):
        self.name = name
        self.contact = contact

class Activity:
    def __init__(self, name, time, location):
        self.name = name
        self.time = time
        self.location = location

class Events:
    def __init__(self, title, date, organiser: Organiser):
        self.title = title
        self.date = date
        self.organiser = organiser
        self.participant = []
        self.activities = []

    def registerParticipant(self, participant: Participant):
        self.participant.append(participant)
        print(f"{participant.name} has been registered for {self.title}")

    def addActivity(self, activity: Activity):
        self.activities.append(activity)
        print(f"Activity '{activity.name}' added to event '{self.title}'")

    def showDetail(self):
        print(f"\nEvent: {self.title} on {self.date}")
        print(f"Organiser: {self.organiser.name} ({self.organiser.contact})")
        print("\nParticipants:")
        for p in self.participant:
            print(f"- {p.name} ({p.email})")
        print("\nActivities:")
```

```
            print(f"- {p.name} ({p.email})")
        print("\nActivities:")
        for a in self.activities:
            print(f"- {a.name} at {a.time}, Location: {a.location}")

org = Organiser("Anirudh", "9876543210")
event = Events("Tech Fest", "2025-04-10", org)

p1 = Participant("Animesh", "animesh@example.com")
p2 = Participant("Shiven", "shiven@example.com")

event.registerParticipant(p1)
event.registerParticipant(p2)

act1 = Activity("Coding Contest", "10:00 AM", "Lab 1")
act2 = Activity("Robotics Show", "1:00 PM", "Auditorium")

event.addActivity(act1)
event.addActivity(act2)

event.showDetail()
```

Experiment 6 Q2.py

```python
class Product:
    def __init__(self, name, cost, stock):
        self.name = name
        self.cost = cost
        self.stock = stock

    def updateStock(self, quantity):
        if quantity <= self.stock:
            self.stock -= quantity
            return True
        else:
            return False


class Customer:
    def __init__(self, name):
        self.name = name
        self.cart = {}

    def Cart(self, product: Product, quantity):
        if product.updateStock(quantity):
            if product.name in self.cart:
                self.cart[product.name]["quantity"] += quantity
            else:
                self.cart[product.name] = {"price": product.cost, "quantity": quantity}
            print(f"Added {quantity} x {product.name} to {self.name}'s cart.")
        else:
            print(f"Only {product.stock} items left in stock for {product.name}.")

    def calculateTotal(self):
        total = 0
        for item in self.cart.values():
            total += item["price"] * item["quantity"]
        return total

    def checkout(self):
        print(f"\n {self.name}'s Order Summary:")
        for item, details in self.cart.items():
            print(f"- {item} x {details['quantity']} = ₹{details['price'] * details['quantity']}")
```

```
        def checkout(self):
            print(f"\n {self.name}'s Order Summary:")
            for item, details in self.cart.items():
                print(f"- {item} x {details['quantity']} = ₹{details['price'] * details['quantity']}")
            print(f"\nTotal Cost: ₹{self.calculateTotal()}")
            print("Order Processed. Thank you for shopping!")


# Inventory setup
inventory = {
    "Laptop": Product("Laptop", 55000, 5),
    "Phone": Product("Phone", 20000, 10),
    "Headphones": Product("Headphones", 1500, 20)
}

# User interaction
print("Welcome to the Online Shopping System")
custName = input("Enter your name: ")
customer = Customer(custName)

while True:
    print("\nAvailable Products:")
    for name, product in inventory.items():
        print(f"{name} - ₹{product.cost} (Stock: {product.stock})")

    choice = input("\nEnter product name to add to cart (or 'done' to checkout): ").title()
    if choice == 'Done':
        break
    elif choice in inventory:
        qty = int(input(f"Enter quantity of {choice}: "))
        customer.Cart(inventory[choice], qty)
    else:
        print("Product not found.")

# Final Summary
customer.checkout()

```

Experiment 6 Q3.py

```python
# Base Class
class Vehicle:
    def _init_(self, vt, rp, agency):
        self.vt = vt
        self.rp = rp
        self.agency = agency
        self.stock = 0

    def VA(self):
        print(f"Currently, there are {self.stock} {self.vt}s available for rent at {self.agency} agency, with a daily rent of Rs {self.rp}.")

# Derived Class
class RentalAgencies(Vehicle):
    def _init_(self, agency, vt, rp, stock):
        super()._init_(vt, rp, agency)
        self.stock = stock  # Override stock in derived class

    def RentalPeriod(self):
        p = int(input("Please enter the rental duration in days: "))
        return p

# Derived Class
class RentalTransaction(RentalAgencies):
    def _init_(self, agency, vt, stock, rp):
        super()._init_(agency, vt, rp, stock)

    def pb(self, p1, price, number):
        # Calculate total amount for rental
        amount = p1 * price * number
        return amount

print("       Welcome to the Car and Bus Rental System!    ")
print("-------------------------------------------------------------------------------")
print("-------------------------------------------------------------------------------")

# Initial Stocks and prices
ne = 200  # Number of cars
pce = 25  # Rent for cars per day
nb = 100  # Number of buses
```

```python
pce = 25  # Rent for cars per day
nb = 100  # Number of buses
pb = 50   # Rent for buses per day
i = 1

while i <= 20:
    print("Please select an option from the menu:")
    print(" 1: Rent a car")
    print(" 2: Rent a bus")
    print(" 3: Exit")

    choice = int(input("Enter your choice: "))
    if (choice == 1):
        v1 = RentalTransaction("Star", "Car", ne, pce)
        v1.VA()
        d = int(input("How many cars would you like to rent? "))
        ne -= d  # Reduce available cars
        p1 = v1.RentalPeriod()
        am = v1.pb(p1, pce, d)
        print(f"Your booking for {d} car(s) from Star agency is confirmed. The total amount is Rs {am}.")

    elif (choice == 2):  # Rent a bus
        v1 = RentalTransaction("Royal", "Bus", nb, pb)
        v1.VA()
        d = int(input("How many buses would you like to rent? "))
        nb -= d  # Reduce available buses
        p1 = v1.RentalPeriod()
        am = v1.pb(p1, pb, d)
        print(f"Your booking for {d} bus(es) from Royal agency is confirmed. The total amount is Rs {am}.")

    elif (choice == 3): # Exit the program
        print("Thank you for using our rental service. Have a great day!")
        break

    else:
        print("Oops! Invalid option. Please select a valid choice.")
```

Experiment 7 Q1.py

```python
import tkinter as tk
from tkinter import ttk

def convert_value():
    try:
        value = float(Value.get())
        conversion = Convert_combobox.get()

        if conversion == "Euros to Yen":
            result = value * 160
            Result.config(text=f"Result: ¥ {round(result, 2)}")

        elif conversion == "Kelvin to Fahrenheit":
            result = (value - 273.15) * 9/5 + 32
            Result.config(text=f"Result: {round(result, 2)} °F")

        elif conversion == "Meter to Feet":
            result = value * 3.28084
            Result.config(text=f"Result: {round(result, 2)} ft")

        else:
            Result.config(text="Select a conversion type.")

    except ValueError:
        Result.config(text="Enter a valid number.")

root = tk.Tk()
root.title("Converter")
root.configure(bg='red')

frame = tk.Frame(root, bg='red')
frame.pack(padx=10, pady=10)

Label = tk.Label(frame, text="Enter a value to convert:", bg='red', fg='white')
Label.grid(row=0, column=0, columnspan=2, pady=5)

Value = tk.Entry(frame, width=10)
Value.grid(row=1, column=0, pady=5)

```

```
38  Value.grid(row=1, column=0, pady=5)
39
40  Convert = ["Euros to Yen", "Kelvin to Fahrenheit", "Meter to Feet"]
41  Convert_combobox = ttk.Combobox(frame, values=Convert, state="readonly", width=20)
42  Convert_combobox.grid(row=1, column=1, padx=5)
43  Convert_combobox.set(Convert[0])
44
45  Button = tk.Button(frame, text="Convert", bg="lightgreen", command=convert_value)
46  Button.grid(row=2, column=1, pady=10)
47
48  Result = tk.Label(frame, text="Result: ", font=("Arial", 12, "bold"), bg='red', fg='white')
49  Result.grid(row=3, column=0, columnspan=2, pady=5)
50
51  root.mainloop()
52
```

Experiment 7 Q2.py

```python
import tkinter as tk
import math

def ChooseShape(*args):
    shape = Shape.get()
    result_label.config(text="")

    if shape == "Circle":
        label1.config(text="Radius:")
        label2.pack_forget()
        entry2.pack_forget()
    else:
        label1.config(text="Length:" if shape == "Rectangle" else "Base:")
        label2.config(text="Width:" if shape == "Rectangle" else "Height:")
        if not label2.winfo_ismapped():
            label2.pack()
            entry2.pack()

def CalculateArea():
    shape = Shape.get()
    try:
        val1 = float(entry1.get())
        val2 = float(entry2.get()) if shape != "Circle" else None

        if shape == "Circle":
            area = math.pi * val1 * val1
        elif shape == "Rectangle":
            area = val1 * val2
        elif shape == "Triangle":
            area = 0.5 * val1 * val2
        else:
            result_label.config(text="Please select a shape.")
            return

        result_label.config(text=f"Area of {shape}: {round(area, 2)}")
    except ValueError:
        result_label.config(text="Enter valid numbers.")

root = tk.Tk()
```

```
root = tk.Tk()
root.title("Geometric Area Calculator")
root.geometry("250x280")
root.configure(bg='#FFD580')

Shape = tk.StringVar()
Shape.trace("w", ChooseShape)

tk.Label(root, text="Select Shape:").pack()
shape_menu = tk.OptionMenu(root, Shape, "Circle", "Rectangle", "Triangle")
shape_menu.pack()

label1 = tk.Label(root, text="Dimension 1:")
label1.pack()
entry1 = tk.Entry(root)
entry1.pack()

label2 = tk.Label(root, text="Dimension 2:")
entry2 = tk.Entry(root)
label2.pack()
entry2.pack()

calculate_button = tk.Button(root, text="Calculate Area", bg="lightblue", command=CalculateArea)
calculate_button.pack(pady=10)

result_label = tk.Label(root, text="", bg='#FFD580', font=('Arial', 10, 'bold'))
result_label.pack()

root.mainloop()
```

Experiment 7 Q3.py

```python
import tkinter as tk

def submit():
    name = nameEntry.get()
    branch = branchVar.get()
    games = [game for game, var in gameVars.items() if var.get()]

    outputText = f" Your name is {name}. \n"
    outputText += f"{name} is from {branch} Department. \n"

    if games:
        outputText += f"{name} is from {branch} Department and loves playing {', '.join(games)}. "

    outputLabel.config(text=outputText)

root = tk.Tk()
root.title("College Info ")
root.geometry("400x400")
root.configure(bg="white")


tk.Label(root, text=" Welcome ", font=("Arial", 16, "bold"), fg="black", bg="white").pack(pady=5)

tk.Label(root, text="Enter Your Name:", font=("Arial", 10, "bold"), fg="black", bg="white").pack(anchor="w", padx=10)
nameEntry = tk.Entry(root, width=30, fg="black", bg="white", insertbackground="orange")
nameEntry.pack(padx=10, pady=2)

tk.Label(root, text="Select Your Branch:", font=("Arial", 10, "bold"), fg="black", bg="white").pack(anchor="w", padx=10)
branchVar = tk.StringVar(value="Computer Engineering")
tk.Radiobutton(root, text="CE", variable=branchVar, value="CE", fg="black", bg="white", selectcolor="white").pack(anchor="w", padx=20)
tk.Radiobutton(root, text="IT", variable=branchVar, value="IT", fg="black", bg="white", selectcolor="white").pack(anchor="w", padx=20)
tk.Radiobutton(root, text="AI&DS", variable=branchVar, value="AI&DS", fg="black", bg="white", selectcolor="white").pack(anchor="w", padx=20)

tk.Label(root, text=" Select Games:", font=("Arial", 10, "bold"), fg="black", bg="white").pack(anchor="w", padx=10)
gameVars = {"Basketball": tk.BooleanVar(), "Football": tk.BooleanVar(), "Badminton": tk.BooleanVar()}
for game, var in gameVars.items():
    tk.Checkbutton(root, text=game, variable=var, fg="black", bg="white", selectcolor="white").pack(anchor="w", padx=20)

tk.Button(root, text=" Submit ", font=("Arial", 10, "bold"), bg="black", fg="white", command=submit).pack(pady=10)
```

```
38
39  tk.Button(root, text=" Submit ", font=("Arial", 10, "bold"), bg="black", fg="white", command=submit).pack(pady=10)
40
41  tk.Label(root, text=" OUTPUT: ", font=("Arial", 10, "bold"), fg="black", bg="white").pack(anchor="w", padx=10)
42  outputLabel = tk.Label(root, text="", font=("Arial", 10), fg="black", bg="white", justify="left")
43  outputLabel.pack(anchor="w", padx=10, pady=5)
44
45  root.mainloop()
```

Experiment 8 Q1.py

```python
import re

phone_pattern = r'^[6-9]\d{9}$'
email_pattern = r'^[\w\.-]+@[\w\.-]+\.\w{2,}$'

phone = input("Enter your phone number: ")
email = input("Enter your email ID: ")

if re.fullmatch(phone_pattern, phone):
    print("Valid phone number")
else:
    print("Invalid phone number. Please enter a 10-digit number starting with 6-9.")

if re.fullmatch(email_pattern, email):
    print("Valid email ID")
else:
    print("Invalid email ID. Please enter a proper format like example@domain.com")

```

Experiment 8 Q2.py

```python
import re

email_pattern = r'[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+'
phone_pattern = r'\b[6-9]\d{9}\b'
date_pattern = r'\b(0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/(\d{4})\b'

with open('data.txt', 'r') as file:
    content = file.read()

emails = re.findall(email_pattern, content)
phones = re.findall(phone_pattern, content)
dates = re.findall(date_pattern, content)

formatted_dates = ['{}/{}/{}'.format(d, m, y) for (d, m, y) in dates]

print("Emails Found:", emails)
print("Phone Numbers Found:", phones)
print("Dates Found:", formatted_dates)
```

Experiment 8 Q3.py

```python
import re

def isStrong(password):
    if len(password) < 8:
        return False

    upper = re.search(r'[A-Z]', password)
    lower = re.search(r'[a-z]', password)
    digit = re.search(r'\d', password)
    special = re.search(r'[^A-Za-z0-9]', password)

    return all([upper, lower, digit, special])

pwd = input("Enter a password to check strength: ")

if isStrong(pwd):
    print("Strong password")
else:
    print("Weak password. Must contain at least:")
    print("- 8 characters\n- 1 uppercase\n- 1 lowercase\n- 1 digit\n- 1 special character")

```

Experiment 8 Q4.py

```python
import re

Pattern = re.compile(
    r'^(https?:\/\/)?'
    r'([\w\-]+\.)+[\w]{2,}'
)

def isValid(url):
    return bool(Pattern.match(url))

url = input("Enter a URL to validate: ")

if isValid(url):
    print("Valid URL!")
else:
    print("Invalid URL.")
```

Experiment 9 Q1.py

```python
import numpy as np


Array_1D = np.array([1,2,3,4,5,6,7,8,9,10,11,12])
print(Array_1D, "\n")

Array_2D = np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
print(Array_2D, "\n")

Array_3D = np.array([[[1,2,3], [4,5,6]], [[7,8,9],[10,11,12]]])
print(Array_3D, "\n")

Array = np.array([1,5,2,6,4,9])

print("Slicing 1D Array: ", Array_1D[2:5])
print("Slicing 2D Array: ", Array_2D[2:,1])
print("Slicing 3D Array: ", Array_3D[0,1])

n = Array_1D.size
N=3
M = n//N
reshaped1D_a = Array_2D.reshape(-1)
reshaped1D_b = Array_3D.reshape(-1)
reshaped2D = Array_1D.reshape((N,M))
reshaped3D = Array_1D.reshape((2,2,-1))
print("Reshaping 1D Array to 2D: \n", reshaped2D)
print("Reshaping 1D Array to 3D: \n", reshaped3D)
print("Reshaping 2D Array to 1D: \n", reshaped1D_a)
print("Reshaping 3D Array to 1D: \n", reshaped1D_b)

NewArray = Array_1D[np.array([3,1,2])]
print("Selected Index of 1D Array: \n", NewArray)
```

Experiment 9 Q2.py

```python
import numpy as np

Array1 = np.array([[1,2,3],[4,5,6],[7,8,9]])
Array2 = np.array([[10,11,12],[13,14,15],[16,17,18]])

Sum = Array1 + Array2
Difference = Array1 - Array2
Product = Array1 * Array2
Division = Array1 / Array2
DotProduct = np.dot(Array1, Array2)
CrossProduct = np.cross(Array1, Array2)
print("First Array: \n", Array1)
print("Second Array: \n", Array2)
print("Sum of Arrays is: \n", Sum)
print("Difference of Arrays is: \n", Difference)
print("Product of Arrays is: \n", Product)
print("Division of Arrays is: \n", Division)
print("Dot product of Arrays is: \n", DotProduct)
print("Cross product of Arrays is: \n", CrossProduct)
```

Experiment 9 Q3.py

```python
import numpy as np

Array = np.array([17, 24, 89, 58, 67, 32, 11, 7, 23, 45])
Mean = np.mean(Array)
Median = np.median(Array)
StandardDeviation = np.std(Array)
Variance = np.var(Array)
Correlation = np.corrcoef(Array)

print("Mean: \n", Mean)
print("Median: \n", Median)
print("Standard Deviation: \n", StandardDeviation)
print("Variance: \n", Variance)
print("Correlation coefficients: \n", Correlation)
```

Experiment 10(1).py

```python
import pandas as pd

df = pd.read_csv(r"C:\Users\aniru\OneDrive\Desktop\Python Experiments QB solution\Iris.csv")
print(df.head(8))

print("Column names:")
print(df.columns)

df.fillna(df.mean(numeric_only=True), inplace=True)

grouped = df.groupby('Species')

for name, group in grouped:
    print(f"\nGroup: {name}")
    print(group.head())

mean_val = df['SepalLengthCm'].mean()
min_val = df['SepalLengthCm'].min()
max_val = df['SepalLengthCm'].max()

print("\nSepalLengthCm Statistics:")
print(f"Mean: {mean_val}")
print(f"Min: {min_val}")
print(f"Max: {max_val}")

```

Experiment 10(2).py

```python
import pandas as pd

df = pd.read_csv(r"C:\Users\aniru\OneDrive\Desktop\Python Experiments QB solution\Automobile.csv")

print("First 8 rows of the dataset:")
print(df.head(8))

print("\nColumn names:")
print(df.columns)

df.fillna(df.mean(numeric_only=True), inplace=True)

if 'make' in df.columns:
    grouped = df.groupby('make')
    for name, group in grouped:
        print(f"\nGroup: {name}")
        print(group.head())

mean_val = df['horsepower'].mean()
min_val = df['horsepower'].min()
max_val = df['horsepower'].max()

print("\nhorsepower stat :")
print(f"Mean: {mean_val}")
print(f"Min: {min_val}")
print(f"Max: {max_val}")
```