1. Write a Python program to calculate the gross salary of an employee. The program should prompt the user for the basic salary (BS) and then compute the dearness allowance (DA) as 70% of BS, the travel allowance (TA) as 30% of BS, and the house rent allowance (HRA) as 10% of BS. Finally, it should calculate the gross salary as the sum of BS, DA, TA, and HRA and display the result.

```
BS = float(input("Enter Basic Salary (BS): "))

DA = 0.7 * BS
TA = 0.3 * BS
HRA = 0.1 * BS
GS = BS + DA + TA + HRA

print(f"Dearness Allowance:{DA}")
print(f"Travel Allowance:{TA}")
print(f"House Rent Allowance:{HRA}")
print(f"Gross Salary:{GS}")
```

2. Write a Python program to calculate the simple interest based on user input. The program should prompt the user to enter the principal amount, the rate of interest, and the time period in years. It should then compute the simple interest using the formula Simple Interest = (Principal×Rate×Time) /100 and display the result.

```
# Simple Interest Calculator

# Input from user
principal = float(input("Enter the principal amount: "))
rate = float(input("Enter the rate of interest: "))
time = float(input("Enter the time period in years: "))

# Calculate simple interest
simple_interest = (principal * rate * time) / 100

# Display result
print("Simple Interest =", simple interest)
```

3. Develop a Python program to manage a task list using lists and tuples, including adding, removing, updating, and sorting tasks.

```
tasks = []
def taskmanager(option):
 if option==1:   #Add task
     task=input("Enter task to add:")
     tasks.append(task)
     print("Task added.")

 elif option==2:   # Remove Task
     task = input("Enter task to remove: ")
     if task in tasks:
      tasks.remove(task)
     print("Task removed!")

 elif option==3:   # Update Task
    old_task = input("Enter task to update: ")
    if old_task in tasks:
         new_task = input("Enter new task: ")
         index = tasks.index(old_task)
         tasks[index] = new_task
    print("Task updated!")

 elif option==4:   # Sort Tasks
    tasks.sort()
    print("Tasks sorted!")

 elif option==5:   # Display Tasks
    print("\nTask List:")
    for i, task in enumerate(tasks, 1):
     print(f"{i}. {task}")

while True:
     print("\nTask List Manager")
     print("1. Add Task")
     print("2. Remove Task")
     print("3. Update Task")
     print("4. Sort Tasks")
     print("5. Display Tasks")
     print("6. Exit")

     choice=int(input("Enter your choice (1-6): "))
     if choice==6:
       print("Exiting program...")
       break
     taskmanager(choice)
```

4. Create a Python code to demonstrate the use of sets and perform set operations (union, intersection, difference) to manage student enrolments in multiple courses / appearing for multiple entrance exams like CET, JEE, NEET etc.

```
cet_candidates = {"Siddharth", "Varun", "Kush", "Rishi"}
jee_candidates = {"Kripa", "Khushaan", "Angad", "Rishi"}
neet_candidates = {"Kush", "Khushaan", "Rishi", "Varun"}

all_candidates = cet_candidates | jee_candidates | neet_candidates
common_candidates = cet_candidates & jee_candidates & neet_candidates
cet_only = cet_candidates - jee_candidates - neet_candidates

print("All candidates appearing in any exam:", all_candidates)
print("Candidates appearing in all exams:", common_candidates)
print("Candidates appearing only in CET:", cet_only)
```

5. Write a Python program to create, update, and manipulate a dictionary of student records, including their grades and attendance.

```
students = {
    "Siddharth": {"grades": [99, 96, 94], "attendance": 98},
    "Kush": {"grades": [94, 95, 92], "attendance": 66},
    "Varun": {"grades": [90, 92, 82], "attendance": 46}
}

# Function to take multiple grades as input
def get_grades():
    grades = []
    n = int(input("Enter the number of grades: "))
    for i in range(n):
        grade = int(input(f"Enter grade {i+1}: "))
        grades.append(grade)
    return grades

# Function to add a new student record
def add_student(name, grades, attendance):
    if name in students:
        print(f"{name} already exists!")
    else:
        students[name] = {"grades": grades, "attendance": attendance}

# Function to update grades
def update_grades(name, new_grades):
    if name in students:
        students[name]["grades"] = new_grades
    else:
        print("Student not found!")

# Function to update attendance
def update_attendance(name, new_attendance):
    if name in students:
        students[name]["attendance"] = new_attendance
    else:
        print("Student not found!")

# Display all student records
def display_records():
    for name, record in students.items():
        print(f"{name}: Grades - {record['grades']}, Attendance - {record['attendance']}%")

# Function to calculate average grade
def average_grade(name):
    if name in students:
        avg = sum(students[name]["grades"]) / len(students[name]["grades"])
        print(f"Average grade of {name}: {avg:.2f}")
    else:
```

```
        print("Student not found!")

# Main function to run the program
def main():
    while True:
        print("\nStudent Record Keeper")
        print("1. Add Student")
        print("2. Update Grades")
        print("3. Update Attendance")
        print("4. Display Records")
        print("5. Get Average Grade")
        print("6. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":
            name = input("Enter name: ")
            grades = get_grades()
            attendance = int(input("Enter attendance percentage: "))
            add_student(name, grades, attendance)

        elif choice == "2":
            name = input("Enter name: ")
            new_grades = get_grades()
            update_grades(name, new_grades)

        elif choice == "3":
            name = input("Enter name: ")
            new_attendance = int(input("Enter new attendance percentage: "))
            update_attendance(name, new_attendance)

        elif choice == "4":
            display_records()

        elif choice == "5":
            name = input("Enter name: ")
            average_grade(name)

        elif choice == "6":
            print("Exiting program...")
            break

        else:
            print("Invalid choice! Please enter a number between 1 and 6.")

# Run the program
main()
```

6. Write a Python program to print a triangle and diamond pattern.

```
# Triangle Pattern
n = int(input("Enter the number of rows for the triangle: "))

print("Triangle Pattern:")
for i in range(1, n + 1):
    print(" " * (n - i) + "*" * (2 * i - 1))

# Diamond Pattern
m = int(input("\nEnter the number of rows for the diamond (half): "))

print("Diamond Pattern:")
# Upper part
for i in range(1, m + 1):
    print(" " * (m - i) + "*" * (2 * i - 1))
# Lower part
for i in range(m - 1, 0, -1):
    print(" " * (m - i) + "*" * (2 * i - 1))
```

7. Write a Python program to find the factorial of a number using function and anonymous function.

```
# Factorial using a regular function
def factorial_function(n):
    fact = 1
    for i in range(1, n + 1):
        fact *= i
    return fact

# Input from user
num = int(input("Enter a number: "))

# Using regular function
print("Factorial using regular function:", factorial_function(num))

# Factorial using an anonymous (lambda) function and reduce
from functools import reduce
factorial_lambda = lambda n: reduce(lambda x, y: x * y, range(1, n + 1)) if n > 0 else 1

# Using lambda function
print("Factorial using lambda function:", factorial_lambda(num))
```

8. Develop a Python program that reads a text file and prints words of specified lengths (e.g., three, four, five, etc.) found within the file.

```
while True:
    wordlength = input("Enter word length (or 'stop' to end): ")
    if wordlength == "stop":
        print("Program exited.")
        exit()
    wordlength = int(wordlength)
    with open("file.txt", "r") as file:
        for line in file:
            for word in line.split():
                if len(word) == wordlength:
                    print(word)
```

9. Write a python code to take a file which contains city names on each line. Alphabetically sort the city names and write it in another file.

```
# Read city names from a file
with open("cities.txt", "r") as infile:
    cities = infile.readlines()

# Remove newline characters and sort
cities = [city.strip() for city in cities]
cities.sort()

# Write sorted city names to another file
with open("sorted_cities.txt", "w") as outfile:
    for city in cities:
        outfile.write(city + "\n")

print("City names sorted and written to 'sorted_cities.txt'")
```

10. Write a Python program that takes two numbers as input and performs division. Implement exception handling to manage division by zero and invalid input errors gracefully.

```
try:
    num1 = float(input("Enter the first number: "))
    num2 = float(input("Enter the second number: "))

    result = num1 / num2
    print("Result of division:", result)

except ValueError:
    print("Invalid input! Please enter numeric values only.")
except ZeroDivisionError:
    print("Division by zero is not allowed.")
except Exception as e:
    print("An unexpected error occurred:",e)
```

11.Demonstrate the use of a Python debugger

---

debug_idle.py - C:\Users\Siddharth\Downloads\Python Practical TSEC\debug_idle.py (3.13.2)

File Edit Format Run Options Window Help

```
#Sample Program with Intentional Errors
# debug_idle.py

def calculate_area():
    import pdb; pdb.set_trace()  # 🔍 Debugger starts here

    length = input("Enter length: ")
    width = input("Enter width: ")

    area = length * width
    print("Area is:", area)

calculate_area()
print("This code is written by Siddharth Maru Div:A Roll No:60.")
```

IDLE Shell 3.13.2

File Edit Shell Debug Options Window Help

```
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=== RESTART: C:\Users\Siddharth\Downloads\Python Practical TSEC\debug_idle.py ==
> c:\users\siddharth\downloads\python practical tsec\debug_idle.py(4)calculate_area()
-> import pdb; pdb.set_trace()  # 🔍 Debugger starts here
(Pdb) n
> c:\users\siddharth\downloads\python practical tsec\debug_idle.py(6)calculate_area()
-> length = input("Enter length: ")
(Pdb) n
Enter length: 5
> c:\users\siddharth\downloads\python practical tsec\debug_idle.py(7)calculate_area()
-> width = input("Enter width: ")
(Pdb) p length
'5'
(Pdb) n
Enter width: 4
> c:\users\siddharth\downloads\python practical tsec\debug_idle.py(9)calculate_area()
-> area = length * width
(Pdb) p width
'4'
(Pdb) n
TypeError: can't multiply sequence by non-int of type 'str'
```

debug_idle.py - C:\Users\Siddharth\Downloads\Python Practical TSEC\debug_idle.py (3.13.2)

File Edit Format Run Options Window Help

```
#Final Fix
def calculate_area():
    import pdb; pdb.set_trace()  # Breakpoint

    length = float(input("Enter length: "))
    width = float(input("Enter width: "))
    area = length * width
    print("Area is:", area)

calculate_area()
print("This code is written by Siddharth Maru Div:A Roll No:60.")
```

IDLE Shell 3.13.2

File Edit Shell Debug Options Window Help

```
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=== RESTART: C:\Users\Siddharth\Downloads\Python Practical TSEC\debug_idle.py ==
> c:\users\siddharth\downloads\python practical tsec\debug_idle.py(3)calculate_area()
-> import pdb; pdb.set_trace()  # Breakpoint
(Pdb) n
> c:\users\siddharth\downloads\python practical tsec\debug_idle.py(5)calculate_area()
-> length = float(input("Enter length: "))
(Pdb) n
Enter length: 4.5
> c:\users\siddharth\downloads\python practical tsec\debug_idle.py(6)calculate_area()
-> width = float(input("Enter width: "))
(Pdb) p length
4.5
(Pdb) n
Enter width: 5
> c:\users\siddharth\downloads\python practical tsec\debug_idle.py(7)calculate_area()
-> area = length * width
(Pdb) p width
5.0
(Pdb) n
> c:\users\siddharth\downloads\python practical tsec\debug_idle.py(8)calculate_area()
-> print("Area is:", area)
(Pdb) p area
22.5
(Pdb) n
Area is: 22.5
--Return--
> c:\users\siddharth\downloads\python practical tsec\debug_idle.py(8)calculate_area()->None
-> print("Area is:", area)
(Pdb) q
Traceback (most recent call last):
  File "C:\Users\Siddharth\Downloads\Python Practical TSEC\debug_idle.py", line 10, in <module>
    calculate_area()
  File "C:\Users\Siddharth\Downloads\Python Practical TSEC\debug_idle.py", line 8, in calculate_area
    print("Area is:", area)
bdb.BdbQuit
```

12. Design a system using classes for vehicles, rental agencies, and rental transactions. Implement methods to handle vehicle availability, rental periods, pricing, and customer bookings.

```
#Base class
class Vehicle:
    def __init__(self,vt,rp):
        self.vt = vt
        self.rp = rp

    def VA(self):
        print(f"The available {self.vt} are {self.stock} with rent of Rs {self.rp} per day of {self.agency} agency")

#Derived class
class rentalagencies(Vehicle):
    def __init__(self,agency,vt,rp):
        self.agency = agency
        Vehicle.__init__(self,vt,rp)

    def Rentalperiod(self):
        p = int(input("Enter the rental period in days: "))
        return p

#Derived from upper derived class
class rentaltransaction(rentalagencies):
    def __init__(self,agency,vt,stock,rp):
        self.stock = stock
        rentalagencies.__init__(self,agency,vt,rp)

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

print("                This is the application for CAR and BUS Rental System              ")
print("-------------------------------------------------------------------")
print("-------------------------------------------------------------------")

#Initial stock and prices
nc = 200 #Number of cars
pc = 25 #Rent for cars per day
nb = 100 #Number of buses
pb = 50 #Rent for buses per day

i = 1

while (i<=20): #for the loop show menu
    print("Please enter the choice as per the menu")
    print("1: For renting the car")
    print("2: For renting the bus")
    print("3: For exit")
    choice = int(input("Enter the choice: "))

    if choice == 1: #Rent a car
        v1 = rentaltransaction("Star","Car",nc,pc)
        v1.VA()
        d = int(input("Enter the number of cars: "))
        if(nc == 0):
            print("Sorry all cars have been booked")
        elif(d > nc):
            print(f"Sorry only {nc} cars are available")
        else:
            nc = nc - d #Reduce available cars
            p1 = v1.Rentalperiod()
            am = v1.pb(p1,pc,d)
            print(f"Your order for {d} Cars from Star agency for {p1} days is booked. Please pay Rupees {am}.")
            print("-------------------------------------------------------------------")

    elif choice == 2: #Rent a bus
        v1 = rentaltransaction("Royal", "Bus", nb, pb)
        v1.VA()
        d = int(input("Enter the number of buses: "))
        if(nb == 0):
            print("Sorry all buses have been booked")
        elif(d > nb):
            print(f"Sorry only {nb} cars are available")
        else:
            nb = nb - d #Reduce available buses
            p1 = v1.Rentalperiod()
            am = v1.pb(p1,pb,d)
            print(f"Your order for {d} Buses from Royal agency for {p1} days is booked. Please pay rupees {am}.")
            print("-------------------------------------------------------------------")

    elif choice == 3: #Exit the program
        print("Thank you for using the rental system. Goodbye!")
        break #Exit the loop and end the program

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

13. Write a GUI program to create a student form containing Name, Age, Branch and Favourite Games. Display all the above contents in Text Box.

```
import tkinter as tk

def submit():
    name = entry_name.get()

    branch = branch_var.get()
    if branch == 1:
        branch_text = "Computer Engineering"
    elif branch == 2:
        branch_text = "Information Technology"
    else:
        branch_text = "Not Selected"

    games = []
    if var_cricket.get():
        games.append("Cricket")
    if var_football.get():
        games.append("Football")
    if var_badminton.get():
        games.append("Badminton")

    game_text = ""
    if games:
        game_text = f" and enjoy playing {', '.join(games)}"

    output_text = (
        f"OUTPUT:\n"
        f"Your name is {name}.\n"
        f"{name} is from {branch_text} Department.\n"
        f"{name} is from {branch_text} Department{game_text}."
 )

    output_label.config(text=output_text)

root = tk.Tk()
root.title("College Admission Form")

tk.Label(root, text="Enter Student Name:").grid(row=0, column=0, sticky="w")
entry_name = tk.Entry(root)
entry_name.grid(row=0, column=1, columnspan=2)

tk.Label(root, text="Select Your Branch:").grid(row=1, column=0, sticky="w")
branch_var = tk.IntVar()
tk.Radiobutton(root, text="Computer Engineering", variable=branch_var, value=1).grid(row=1, column=1, sticky="w")
tk.Radiobutton(root, text="Information Technology", variable=branch_var, value=2).grid(row=1, column=2, sticky="w")
```

```
tk.Label(root, text="Select Favorite Games:").grid(row=2, column=0, sticky="w")
var_cricket = tk.BooleanVar()
var_football = tk.BooleanVar()
var_badminton = tk.BooleanVar()
tk.Checkbutton(root, text="Cricket", variable=var_cricket).grid(row=2, column=1, sticky="w")
tk.Checkbutton(root, text="Football", variable=var_football).grid(row=2, column=2, sticky="w")
tk.Checkbutton(root, text="Badminton", variable=var_badminton).grid(row=2, column=3, sticky="w")

tk.Button(root, text="Submit", command=submit).grid(row=3, column=1, pady=10)

output_label = tk.Label(root, text="", fg="blue", justify="left", anchor="w")
output_label.grid(row=4, column=0, columnspan=4, sticky="w")
tk.Label(root, text="By Siddharth Maru, Div:A, Roll No:60").grid(row=5, column=0, columnspan=4, pady=5, sticky="w")

root.mainloop()
```

14. Write a Python script that prompts the user to enter a password. Use regular expressions to validate the password based on these criteria: At least 8 characters long, Contains at least one uppercase letter, one lowercase letter, one digit, and one special character.

```
import re

# Prompt user to enter a password
password = input("Enter a password: ")

# Define the pattern for validation
pattern = r'^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[^A-Za-z0-9]).{8,}$'

# Validate using regex
if re.match(pattern, password):
    print("Password is valid.")
else:
    print("Invalid password! It must be at least 8 characters long and include:")
    print("- At least one uppercase letter")
    print("- At least one lowercase letter")
    print("- At least one digit")
    print("- At least one special character")
```

15. Write a Python program to create a 1D, 2D, and 3D NumPy array. Perform basic operations like reshaping, slicing, and indexing. Calculate the dot product and cross product of two vectors.

```
import numpy as np

arr_1d = np.array([1, 2, 3, 4, 5])
arr_2d = np.array([[1, 2, 3], [4, 5, 6]])
arr_3d = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])

# Reshaping
reshaped = arr_1d.reshape(5, 1)

# Slicing
sliced = arr_2d[0:2,0:2]

# Indexing
indexed = arr_3d[0, 1, 1]

vector1 = np.array([1, 2, 3])
vector2 = np.array([4, 5, 6])
dot_product = np.dot(vector1, vector2)
cross_product = np.cross(vector1, vector2)

print("1D Array:\n", arr_1d)
print("2D Array:\n", arr_2d)
print("3D Array:\n", arr_3d)
print("Reshaped Array:\n", reshaped)
print("Sliced Array:\n", sliced)
print("Indexed Element:", indexed)
print("Dot Product:\n", dot_product)
print("Cross Product:\n", cross_product)
```

16. Develop a Python script to create two arrays of the same shape and perform element-wise addition, subtraction, multiplication, and division.

```
import numpy as np

A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])

add = A + B
sub = A - B
mul = A * B
div = A / B

print("Addition:\n", add)
print("Subtraction:\n", sub)
print("Multiplication:\n", mul)
print("Division:\n", div)
```

17. Using the Iris Data perform the following tasks: I. Read the first 8 rows of the dataset. ii. Display the column names of the Iris dataset. iii. Fill any missing data with the mean value of the respective column. iv. Remove rows that contain any missing values.

```
import pandas as pd

df = pd.read_csv(r"C:\Users\Siddharth\Downloads\Python Practical TSEC\archive.zip")
print("First 8 rows:")
print(df.head(8))

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

df_filled = df.fillna(df.mean(numeric_only=True))
print("\nData after filling missing values with mean:")
print(df_filled)

df_dropped = df.dropna()
print("\nData after dropping rows with missing values:")
print(df_dropped)

grouped = df.groupby('Species')
print("\nGrouped data by Species:")
print(grouped.size())

print("\nSepalLengthCm Statistics:")
print("Mean:", df['SepalLengthCm'].mean())
print("Minimum:", df['SepalLengthCm'].min())
print("Maximum:", df['SepalLengthCm'].max())
```

18. Don’t know

19. Write a python code to take a csv file as input with coordinates of points in three dimensions. Find out the two closest points.

```
import csv
import math

# Function to calculate Euclidean distance between two 3D points
def distance(p1, p2):
    return math.sqrt((p1[0] - p2[0])**2 +
                     (p1[1] - p2[1])**2 +
                     (p1[2] - p2[2])**2)

# Read coordinates from CSV file
points = []
with open("points.csv", "r") as file:
    reader = csv.reader(file)
    for row in reader:
        # Convert strings to float
        x, y, z = map(float, row)
        points.append((x, y, z))

# Find the two closest points
min_dist = float('inf')
closest_pair = (None, None)

for i in range(len(points)):
    for j in range(i + 1, len(points)):
        d = distance(points[i], points[j])
        if d < min_dist:
            min_dist = d
            closest_pair = (points[i], points[j])

# Display result
print("Closest Points:", closest_pair[0], "and", closest_pair[1])
print("Distance:", min_dist)
```

20. Develop a Python program that simulates a banking system with a function to withdraw money. Raise custom exceptions for scenarios such as insufficient funds and invalid account numbers.

```python
# Custom exception for invalid account number
class InvalidAccountError(Exception):
    pass

# Custom exception for insufficient funds
class InsufficientFundsError(Exception):
    pass

# Sample account data
accounts = {
    "12345": 5000,
    "67890": 3000
}

# Function to withdraw money
def withdraw(account_no, amount):
    if account_no not in accounts:
        raise InvalidAccountError("Invalid account number!")
    if accounts[account_no] < amount:
        raise InsufficientFundsError("Insufficient funds!")

    accounts[account_no] -= amount
    print(f"Withdrawal successful. Remaining balance: {accounts[account_no]}")

# Main code
try:
    acc_no = input("Enter account number: ")
    amt = float(input("Enter amount to withdraw: "))
    withdraw(acc_no, amt)

except InvalidAccountError as e:
    print("Error:", e)

except InsufficientFundsError as e:
    print("Error:", e)

except ValueError:
    print("Invalid input! Please enter numbers only.")
```

21. Develop a Python GUI application that performs various unit conversions such as currency (Rupees to Dollars), temperature (Celsius to Fahrenheit), and length (Inches to Feet).

```
import tkinter as tk

def convert():
    try:
        value = float(entry.get())
        conversion = conversion_var.get()

        if conversion == "Rupees to Dollars":
            result = value * 0.012  # Approximate rate
        elif conversion == "Celsius to Fahrenheit":
            result = (value * 9/5) + 32
        elif conversion == "Inches to Feet":
            result = value / 12
        else:
            result = "Invalid conversion"

        result_label.config(text=f"Result: {round(result, 2)}")
    except ValueError:
        result_label.config(text="Please enter a valid number.")

root = tk.Tk()
root.title("Unit Converter")
root.geometry("300x250")
root.resizable(False, False)

tk.Label(root, text="Enter value:").pack(pady=5)
entry = tk.Entry(root)
entry.pack(pady=5)

tk.Label(root, text="Select conversion:").pack(pady=5)
conversion_var = tk.StringVar()
conversion_var.set("Rupees to Dollars")  # Default value
options = ["Rupees to Dollars", "Celsius to Fahrenheit", "Inches to Feet"]
dropdown = tk.OptionMenu(root, conversion_var, *options)
dropdown.pack(pady=5)

convert_btn = tk.Button(root, text="Convert", command=convert)
convert_btn.pack(pady=10)

result_label = tk.Label(root, text="Result:")
result_label.pack(pady=10)
tk.Label(root, text="By Siddharth Maru, Div:A, Roll No:60").pack(pady=5)

root.mainloop()
```

22. Develop a Python GUI application that calculates the areas of different geometric figures such as circles, rectangles, and triangles. Allows users to input the necessary dimensions for various geometric figures and calculate their respective areas. The application should include input fields for the dimensions, buttons to perform the calculations, and labels to display the results.

```
import tkinter as tk
import math

def calculate_area():
    shape = shape_var.get()
    try:
        if shape == "Circle":
            radius = float(entry1.get())
            area = math.pi * radius ** 2
            result_var.set(f"Area of Circle: {area:.2f}")
        elif shape == "Rectangle":
            length = float(entry1.get())
            width = float(entry2.get())
            area = length * width
            result_var.set(f"Area of Rectangle: {area:.2f}")
        elif shape == "Triangle":
            base = float(entry1.get())
            height = float(entry2.get())
            area = 0.5 * base * height
            result_var.set(f"Area of Triangle: {area:.2f}")
    except ValueError:
        result_var.set("Invalid input. Please enter numbers.")x

def update_fields(*args):
    entry1.delete(0, tk.END)
    entry2.delete(0, tk.END)

    shape = shape_var.get()

    if shape == "Circle":
        label1.config(text="Radius:")
        label2.grid_remove()
        entry2.grid_remove()

    elif shape == "Rectangle":
        label1.config(text="Length:")
        label2.config(text="Width:")
        label2.grid()
        entry2.grid()

    elif shape == "Triangle":
        label1.config(text="Base:")
        label2.config(text="Height:")
        label2.grid()
        entry2.grid()
```

```
root = tk.Tk()
root.title("Area Calculator")
root.geometry("300x270")

shape_var = tk.StringVar()
shape_var.set("Circle")
shape_var.trace("w", update_fields)

tk.Label(root, text="Select Shape:").pack(pady=5)
tk.OptionMenu(root, shape_var, "Circle", "Rectangle", "Triangle").pack()

frame = tk.Frame(root)
frame.pack(pady=10)

label1 = tk.Label(frame, text="Radius:")
label1.grid(row=0, column=0, padx=5, pady=5)
entry1 = tk.Entry(frame)
entry1.grid(row=0, column=1)

label2 = tk.Label(frame, text="")
label2.grid(row=1, column=0, padx=5, pady=5)
entry2 = tk.Entry(frame)
entry2.grid(row=1, column=1)

result_var = tk.StringVar()
tk.Label(root, textvariable=result_var, fg="blue").pack(pady=10)

tk.Button(root, text="Calculate Area", command=calculate_area).pack()

tk.Label(root, text="By Siddharth Maru, Div:A, Roll No:60").pack(pady=5)

update_fields()
root.mainloop()
```

23. Write a Python program to explore basic arithmetic operations. The program should prompt the user to enter two numbers and then perform addition, subtraction, multiplication, division, and modulus operations on those numbers. The results of each operation should be displayed to the user.

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))

addition = num1 + num2
subtraction = num1 - num2
multiplication = num1 * num2
division = num1 / num2
modulus = num1 % num2
floor  = num1 // num2

print(f"\nResults:")
print(f"{num1} + {num2} = {addition}")
print(f"{num1} - {num2} = {subtraction}")
print(f"{num1} * {num2} = {multiplication}")
print(f"{num1} / {num2} = {division}")
print(f"{num1} % {num2} = {modulus}")
print(f"{num1} // {num2} = {floor}")
```

24. Create a Python program to check whether the given input is a digit, lowercase character, uppercase character, or a special character using an 'if else-if' ladder.

```
# Take a single character input from the user
ch = input("Enter a single character: ")

# Check the type of character using if-elif-else
if ch.isdigit():
    print("It is a digit.")
elif ch.islower():
    print("It is a lowercase character.")
elif ch.isupper():
    print("It is an uppercase character.")
else:
    print("It is a special character.")
```

25. Write a Python program to declare the result as pass, second class, first class, and distinction, using an 'if else-if' ladder.

```
# Take percentage as input
percentage = float(input("Enter your percentage: "))

# Determine result using if-elif-else
if percentage >= 75:
    print("Result: Distinction")
elif percentage >= 60:
    print("Result: First Class")
elif percentage >= 50:
    print("Result: Second Class")
elif percentage >= 35:
    print("Result: Pass")
else:
    print("Result: Fail")
```

26. Write a Python program to demonstrate multilevel inheritance to calculate the gross salary of an employee. The program should prompt the user for the basic salary (BS) in grandfather class and then compute the dearness allowance (DA) as 70% of BS, the travel allowance (TA) as 30% of BS, and the house rent allowance (HRA) as 10% of BS. Finally, it should calculate the gross salary as the sum of BS, DA, TA, and HRA and display the result.

```python
# Grandfather class: Input basic salary
class Grandfather:
    def __init__(self):
        self.bs = float(input("Enter Basic Salary (BS): "))

# Father class: Calculate DA and TA
class Father(Grandfather):
    def __init__(self):
        super().__init__()
        self.da = 0.70 * self.bs
        self.ta = 0.30 * self.bs

# Child class: Calculate HRA and Gross Salary
class Child(Father):
    def __init__(self):
        super().__init__()
        self.hra = 0.10 * self.bs
        self.gross_salary = self.bs + self.da + self.ta + self.hra

    def display(self):
        print("Basic Salary (BS):", self.bs)
        print("Dearness Allowance (DA):", self.da)
        print("Travel Allowance (TA):", self.ta)
        print("House Rent Allowance (HRA):", self.hra)
        print("Gross Salary:", self.gross_salary)

# Create object of Child class and display result
emp = Child()
emp.display()
```

27. Write a Python program to demonstrate multilevel inheritance for calculating the simple interest based on user input. The program should prompt the user to enter the principal amount, the rate of interest, and the time period in years. It should then compute the simple interest using the formula Simple Interest = (Principal×Rate×Time) /100 and display the result.

```python
# Grandfather class: Input for principal amount, rate, and time
class Grandfather:
    def __init__(self):
        self.principal = float(input("Enter the principal amount: "))
        self.rate = float(input("Enter the rate of interest: "))
        self.time = float(input("Enter the time period in years: "))

# Father class: Calculate simple interest
class Father(Grandfather):
    def __init__(self):
        super().__init__()

    def calculate_simple_interest(self):
        self.simple_interest = (self.principal * self.rate * self.time) / 100
        return self.simple_interest

# Child class: Display result
class Child(Father):
    def __init__(self):
        super().__init__()

    def display(self):
        si = self.calculate_simple_interest()
        print(f"Principal Amount: {self.principal}")
        print(f"Rate of Interest: {self.rate}%")
        print(f"Time Period: {self.time} years")
        print(f"Simple Interest: {si}")

# Create an object of Child class and display result
emp = Child()
emp.display()
```

28. Write a Python program to demonstrate List creation and 10 methods

```
# 1. List creation
my_list = [1, 2, 3, 4, 5]
print("Initial List:", my_list)

# 2. append() - Adds an item to the end of the list
my_list.append(6)
print("After append(6):", my_list)

# 3. insert() - Inserts an item at a specific index
my_list.insert(2, 10)  # Inserts 10 at index 2
print("After insert(2, 10):", my_list)

# 4. remove() - Removes the first occurrence of the specified item
my_list.remove(10)  # Removes the first occurrence of 10
print("After remove(10):", my_list)

# 5. pop() - Removes an item at the specified index and returns it
popped_item = my_list.pop(3)  # Removes item at index 3
print("After pop(3):", my_list)
print("Popped item:", popped_item)

# 6. clear() - Removes all items from the list
my_list.clear()
print("After clear():", my_list)

# Recreate list to demonstrate further methods
my_list = [1, 2, 3, 4, 5, 6]

# 7. index() - Returns the index of the first occurrence of the specified item
index_of_4 = my_list.index(4)
print("Index of 4:", index_of_4)

# 8. count() - Returns the number of occurrences of the specified item
count_of_3 = my_list.count(3)
print("Count of 3:", count_of_3)

# 9. reverse() - Reverses the order of the list
my_list.reverse()
print("After reverse():", my_list)

# 10. sort() - Sorts the list in ascending order (modifies the list)
my_list.sort()
print("After sort():", my_list)

# 11. copy() - Returns a shallow copy of the list
my_list_copy = my_list.copy()
print("Copied List:", my_list_copy)
```

29. Write a Python program to demonstrate Dictionary creation and 10 methods

```
# 1. Dictionary creation
my_dict = {
    'name': 'John',
    'age': 25,
    'city': 'New York'
}
print("Initial Dictionary:", my_dict)

# 2. update() - Updates the dictionary with elements from another dictionary or key-value pairs
my_dict.update({'age': 26, 'country': 'USA'})
print("After update({'age': 26, 'country': 'USA'}):", my_dict)

# 3. get() - Returns the value of the specified key
age = my_dict.get('age')
print("Age:", age)

# 4. keys() - Returns a view object that displays a list of all the keys
keys = my_dict.keys()
print("Keys:", keys)

# 5. values() - Returns a view object that displays a list of all the values
values = my_dict.values()
print("Values:", values)

# 6. items() - Returns a view object that displays a list of a dictionary's key-value tuple pairs
items = my_dict.items()
print("Items:", items)

# 7. pop() - Removes the item with the specified key and returns its value
popped_value = my_dict.pop('city')
print("After pop('city'):", my_dict)
print("Popped value:", popped_value)

# 8. popitem() - Removes and returns the last key-value pair
last_item = my_dict.popitem()
print("After popitem():", my_dict)
print("Last item popped:", last_item)

# 9. clear() - Removes all items from the dictionary
my_dict.clear()
print("After clear():", my_dict)
```

```
# Recreate the dictionary for further demonstration
my_dict = {
    'name': 'Alice',
    'age': 30,
    'city': 'Los Angeles',
    'country': 'USA'
}

# 10. copy() - Returns a shallow copy of the dictionary
my_dict_copy = my_dict.copy()
print("Original Dictionary:", my_dict)
print("Copied Dictionary:", my_dict_copy)
```

30. Write a Python program to demonstrate set creation and 10 methods

```
# 1. Set creation
my_set = {1, 2, 3, 4, 5}
print("Initial Set:", my_set)

# 2. add() - Adds an element to the set
my_set.add(6)
print("After add(6):", my_set)

# 3. update() - Adds multiple elements to the set (can be a list, tuple, or another set)
my_set.update([7, 8, 9])
print("After update([7, 8, 9]):", my_set)

# 4. remove() - Removes the specified element from the set. If element is not found, raises KeyError
my_set.remove(3)
print("After remove(3):", my_set)

# 5. discard() - Removes the specified element from the set if present, but does not raise an error if not found
my_set.discard(10)  # 10 is not in the set, so no error
print("After discard(10):", my_set)

# 6. pop() - Removes and returns an arbitrary element from the set
popped_item = my_set.pop()  # Since set is unordered, it pops any element
print("After pop():", my_set)
print("Popped item:", popped_item)

# 7. clear() - Removes all elements from the set
my_set.clear()
print("After clear():", my_set)

# Recreate the set for further demonstration
my_set = {1, 2, 3, 4, 5}

# 8. copy() - Returns a shallow copy of the set
my_set_copy = my_set.copy()
print("Original Set:", my_set)
print("Copied Set:", my_set_copy)

# 9. union() - Returns a new set containing all items from both sets (without duplicates)
set2 = {4, 5, 6, 7}
union_set = my_set.union(set2)
print("Union of sets:", union_set)

# 10. intersection() - Returns a new set containing only the items that exist in both sets
intersection_set = my_set.intersection(set2)
print("Intersection of sets:", intersection_set)
```

```
# 11. difference() - Returns a new set containing elements in the first set that are not in the second
difference_set = my_set.difference(set2)
print("Difference of sets (my_set - set2):", difference_set)

# 12. issubset() - Returns True if the set is a subset of the other set
is_subset = my_set.issubset(set2)
print("Is my_set a subset of set2?", is_subset)

# 13. issuperset() - Returns True if the set is a superset of the other set
is_superset = my_set.issuperset(set2)
print("Is my_set a superset of set2?", is_superset)
```

31. Write a Python program to plot multiple bar chart for placement data years = ['2020', '2021', '2022', '2023'] CSE = [50, 60, 65, 70], IT = [20, 25, 30, 35] EXTC = [15, 20, 25, 30] and AIDS = [10, 15, 20, 25]

```
import matplotlib.pyplot as plt
import numpy as np

# Data
years = ['2020', '2021', '2022', '2023']
CSE = [50, 60, 65, 70]
IT = [20, 25, 30, 35]
EXTC = [15, 20, 25, 30]
AIDS = [10, 15, 20, 25]

# Number of groups
n_years = len(years)

# Bar width
bar_width = 0.2

# Positions of bars on the x-axis
indices = np.arange(n_years)

# Creating bars for each department with custom colors
plt.bar(indices, CSE, width=bar_width, color='blue', label='CSE')
plt.bar(indices + bar_width, IT, width=bar_width, color='green', label='IT')
plt.bar(indices + 2 * bar_width, EXTC, width=bar_width, color='orange', label='EXTC')
plt.bar(indices + 3 * bar_width, AIDS, width=bar_width, color='purple', label='AIDS')

# Labeling and styling
plt.xlabel('Years')
plt.ylabel('Placement Numbers')
plt.title('Placement Data (Multiple Bar Chart)')
plt.xticks(indices + bar_width * 1.5, years)
plt.legend()

# Display the plot
plt.tight_layout()
plt.show()
```

32. Write a Python program to plot stacked bar chart for placement data years = ['2020', '2021', '2022', '2023'] CSE = [50, 60, 65, 70], IT = [20, 25, 30, 35] EXTC = [15, 20, 25, 30] and AIDS = [10, 15, 20, 25]

```
import matplotlib.pyplot as plt
import numpy as np

# Data
years = ['2020', '2021', '2022', '2023']
CSE = [50, 60, 65, 70]
IT = [20, 25, 30, 35]
EXTC = [15, 20, 25, 30]
AIDS = [10, 15, 20, 25]

# Convert years into numeric positions for the x-axis
indices = np.arange(len(years))

# Colors for each department
colors = {
    'CSE': '#1f77b4',  # Blue
    'IT': '#2ca02c',   # Green
    'EXTC': '#ff7f0e', # Orange
    'AIDS': '#d62728'  # Red
}

# Plot stacked bars with custom colors and labels
plt.bar(indices, CSE, label='CSE (Computer Science)', color=colors['CSE'])
plt.bar(indices, IT, bottom=CSE, label='IT (Information Technology)', color=colors['IT'])
plt.bar(indices, EXTC, bottom=np.array(CSE) + np.array(IT), label='EXTC (Electronics & Telecommunication)', color=colors['EXTC'])
plt.bar(indices, AIDS, bottom=np.array(CSE) + np.array(IT) + np.array(EXTC), label='AIDS (Artificial Intelligence & Data Science)', color=colors['AIDS'])

# Labeling and styling
plt.xlabel('Years', fontsize=12)
plt.ylabel('Placement Numbers', fontsize=12)
plt.title('Placement Data (Stacked Bar Chart)', fontsize=14)
plt.xticks(indices, years, fontsize=10)
plt.legend(title='Departments', fontsize=9, title_fontsize=10)
plt.grid(axis='y', linestyle='--', linewidth=0.5, alpha=0.7)

# Display the plot
plt.tight_layout()
plt.show()
```

33. Write a Python program to find the ncr by using function

```python
# Function to calculate factorial
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

# Function to calculate nCr
def ncr(n, r):
    if r > n:
        return 0
    return factorial(n) // (factorial(r) * factorial(n - r))

# Input from user
n = int(input("Enter the value of n: "))
r = int(input("Enter the value of r: "))

# Calculate nCr and display the result
result = ncr(n, r)
print(f"The value of {n}C{r} (nCr) is: {result}")
```

34. Write a Python program to demonstrate the module and package(online steps dekho gpt ya copilot se)

35. Write a Python program to check whether the given number is Armstrong number using function

```python
# Function to check Armstrong number
def is_armstrong_number(number):
    # Convert the number to a string to iterate through digits
    digits = [int(digit) for digit in str(number)]
    # Calculate the sum of digits raised to the power of their length
    armstrong_sum = sum(digit ** len(digits) for digit in digits)
    # Check if the sum is equal to the original number
    return armstrong_sum == number

# Input from the user
num = int(input("Enter a number: "))

# Check and display result
if is_armstrong_number(num):
    print(f"{num} is an Armstrong number.")
else:
    print(f"{num} is not an Armstrong number.")
```

36. Write a Python script that prompts the user to enter mobile number and mail-id. Use regular expressions to validate them.

```
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("Phone number is valid.")
else:
    print("Invalid phone number.")

if re.fullmatch(email_pattern, email):
    print("Email ID is valid.")
else:
    print("Invalid email ID.")
```