**1. Identify the Employee Category**

A company categorizes its employees based on their salaries:

Salary above ₹50,000 → "High Income"

Salary between ₹30,000 and ₹50,000 → "Medium Income"

Salary below ₹30,000 → "Low Income"

Write a Python program that takes an employee’s salary as input and prints their category.

#### **2 . Odd or Even Streetlight System**

*A city uses an even-odd rule to manage electricity usage in streetlights. Streetlights with even numbers are turned on at night, while odd-numbered lights remain off. Write a program that takes a streetlight number as input and prints whether it should be ON or OFF.*

| light\_number = int(input("Enter the streetlight number: "))  if light\_number % 2 == 0:  print("Streetlight is ON.") else:  print("Streetlight is OFF.") |
| --- |

***3. Imagine you are designing a basic ATM-like interface where users can:***

1. **Check Balance**
2. **Deposit Money**
3. **Withdraw Money**
4. **Exit the System**

**Your Task:** Write a Python program that displays a menu, takes user input, and performs the respective operation based on the user's choice.

| # Simple Menu-Driven Banking System  balance = 5000 # Initial account balance  while True:  print("\n🏦 Welcome to Simple Bank 🏦")  print("1. Check Balance")  print("2. Deposit Money")  print("3. Withdraw Money")  print("4. Exit")    choice = int(input("Enter your choice (1-4): "))   if choice == 1:  print(f"Your current balance is: ₹{balance}")   elif choice == 2:  deposit = float(input("Enter amount to deposit: ₹"))  if deposit > 0:  balance += deposit  print(f"₹{deposit} deposited successfully! New balance: ₹{balance}")  else:  print("Invalid deposit amount. Please enter a valid amount.")   elif choice == 3:  withdraw = float(input("Enter amount to withdraw: ₹"))  if withdraw > balance:  print("Insufficient funds!")  elif withdraw <= 0:  print("Invalid amount. Please enter a valid amount.")  else:  balance -= withdraw  print(f"₹{withdraw} withdrawn successfully! Remaining balance: ₹{balance}")   elif choice == 4:  print("Thank you for using Simple Bank! Have a great day! 😊")  break   else:  print("Invalid choice! Please enter a number between 1 and 4.") |
| --- |

### **4. Counting Passengers in a Bus**

*A city bus stops at different stations, and passengers board at each stop. The bus conductor notes down the number of passengers at each stop for 5 stops. Write a Python program to take input for each stop and calculate the total number of passengers at the end.*

| total\_passengers = 0  for stop in range(1, 6):  passengers = int(input(f"Enter the number of passengers at stop {stop}: "))  total\_passengers += passengers  print(f"Total passengers in the bus: {total\_passengers}") |
| --- |

### **5. ATM PIN Verification**

*A bank ATM allows a user to enter their PIN a maximum of 3 times. If the correct PIN (1234) is entered within 3 attempts, the user gains access; otherwise, the card is blocked. Write a Python program for this.*

| correct\_pin = 1234 attempts = 3  while attempts > 0:  pin = int(input("Enter your ATM PIN: "))    if pin == correct\_pin:  print("Access Granted!")  break  else:  attempts -= 1  print(f"Incorrect PIN! You have {attempts} attempts left.")  if attempts == 0:  print("Card Blocked!") |
| --- |

### **6. Reverse a Number**

*Write a Python program to reverse a given number. For example, if the input is 12345, the output should be 54321.*

| num = int(input("Enter a number: ")) rev = 0  while num > 0:  digit = num % 10  rev = rev \* 10 + digit  num //= 10  print(f"Reversed number: {rev}") |
| --- |

### **7. ATM Cash Dispensing Machine**

*An ATM machine dispenses only ₹500 and ₹2000 notes. Write a Python program that takes an amount as input and prints how many ₹2000 and ₹500 notes will be given, assuming the amount is a multiple of 500.*

| amount = int(input("Enter the amount (must be a multiple of ₹500): "))  if amount % 500 != 0:  print("Invalid amount! Enter a multiple of ₹500.") else:  notes\_2000 = amount // 2000  remaining = amount % 2000  notes\_500 = remaining // 500    print(f"₹2000 notes: {notes\_2000}")  print(f"₹500 notes: {notes\_500}") |
| --- |

### **8. Shopping Cart - List Operations**

📌 *You are creating a shopping cart program. Users should be able to add items to the cart and view the final list. Implement a program that allows a user to add items (names of products) until they type "done" and then prints all the items in the cart.*

| cart = []  while True:  item = input("Enter an item to add to the cart (or type 'done' to finish): ")  if item.lower() == 'done':  break  cart.append(item)  print("Your shopping cart contains:", cart) |
| --- |

### **9. Student Grades - Tuple**

📌 *A university stores student grades in tuples because they are immutable. Write a program that takes a student's grades for 5 subjects and prints the highest, lowest, and average grade.*

| grades = tuple(map(int, input("Enter grades for 5 subjects separated by space: ").split()))  print(f"Highest grade: {max(grades)}") print(f"Lowest grade: {min(grades)}") print(f"Average grade: {sum(grades) / len(grades):.2f}") |
| --- |

### **10 . Movie Collection - Dictionary**

📌 *You are maintaining a movie collection where each movie has a rating (out of 10). Write a program to add movies, update ratings, and display all movies with their ratings.*

| movies = {}  while True:  print("\nMovie Collection")  print("1. Add Movie")  print("2. Update Rating")  print("3. View Collection")  print("4. Exit")    choice = input("Enter your choice: ")    if choice == "1":  name = input("Enter movie name: ")  rating = float(input("Enter rating out of 10: "))  movies[name] = rating  elif choice == "2":  name = input("Enter movie name to update rating: ")  if name in movies:  rating = float(input("Enter new rating: "))  movies[name] = rating  else:  print("Movie not found.")  elif choice == "3":  print("Movie Collection:", movies)  elif choice == "4":  break  else:  print("Invalid choice.") |
| --- |

### **11. Removing Duplicates from a List - Set**

📌 *A librarian is digitizing book records but some book titles are repeated. Write a program to remove duplicates from a list using a set.*

| books = ["Python Basics", "Data Science", "Python Basics", "Machine Learning", "Data Science"]  unique\_books = set(books) print("Unique Books:", unique\_books) |
| --- |

### **12. ATM Transaction System**

📌 *You are designing an ATM system where users can check their balance, withdraw money, or deposit money. Implement this using functions.*

| *def check\_balance(balance):  print(f"Your current balance is: ₹{balance}")  def deposit(balance, amount):  balance += amount  print(f"₹{amount} deposited successfully.")  return balance  def withdraw(balance, amount):  if amount > balance:  print("Insufficient balance!")  else:  balance -= amount  print(f"₹{amount} withdrawn successfully.")  return balance  # Driver Code balance = 5000 while True:  print("\nATM Menu")  print("1. Check Balance")  print("2. Deposit Money")  print("3. Withdraw Money")  print("4. Exit")    choice = input("Enter your choice: ")    if choice == "1":  check\_balance(balance)  elif choice == "2":  amount = int(input("Enter amount to deposit: "))  balance = deposit(balance, amount)  elif choice == "3":  amount = int(input("Enter amount to withdraw: "))  balance = withdraw(balance, amount)  elif choice == "4":  print("Exiting ATM. Have a nice day!")  break  else:  print("Invalid choice!")* |
| --- |

### ***13. Student Grade Calculator***

*📌 Create a function that takes a student's marks and returns their grade based on the following criteria:*

* *90-100: A*
* *80-89: B*
* *70-79: C*
* *60-69: D*
* *Below 60: F*

| *def calculate\_grade(marks):  if marks >= 90:  return "A"  elif marks >= 80:  return "B"  elif marks >= 70:  return "C"  elif marks >= 60:  return "D"  else:  return "F"  marks = int(input("Enter student marks: ")) print("Grade:", calculate\_grade(marks))* |
| --- |

### ***14. Simple Calculator Using Functions***

*📌 \*Write a calculator function that takes two numbers and an operator (+, -, , /) and returns the result.*

| *def calculator(a, b, operator):  if operator == "+":  return a + b  elif operator == "-":  return a - b  elif operator == "\*":  return a \* b  elif operator == "/":  return a / b if b != 0 else "Division by zero error!"  else:  return "Invalid operator!"  num1 = float(input("Enter first number: ")) num2 = float(input("Enter second number: ")) op = input("Enter operator (+, -, \*, /): ")  print("Result:", calculator(num1, num2, op))* |
| --- |

### ***15. Find the Maximum of Three Numbers***

*📌 Write a function that takes three numbers as input and returns the maximum of the three.*

| *def find\_max(a, b, c):  return max(a, b, c)  num1 = int(input("Enter first number: ")) num2 = int(input("Enter second number: ")) num3 = int(input("Enter third number: "))  print("The maximum number is:", find\_max(num1, num2, num3))* |
| --- |

### ***16. Create a Student Class***

*📌 Create a class Student with attributes name and marks. Add a method to check if the student has passed (pass mark: 40).*

| *class Student:  def \_\_init\_\_(self, name, marks):  self.name = name  self.marks = marks   def has\_passed(self):  return "Passed" if self.marks >= 40 else "Failed"  student1 = Student("John", 45) print(f"{student1.name} has {student1.has\_passed()}")* |
| --- |

### ***17. Bank Account Class***

*📌 Create a class BankAccount with attributes account\_holder and balance. Add a method deposit() to add money and withdraw() to deduct money.*

| *class BankAccount:  def \_\_init\_\_(self, account\_holder, balance=0):  self.account\_holder = account\_holder  self.balance = balance   def deposit(self, amount):  self.balance += amount  print(f"Deposited {amount}. New balance: {self.balance}")   def withdraw(self, amount):  if amount > self.balance:  print("Insufficient funds!")  else:  self.balance -= amount  print(f"Withdrew {amount}. New balance: {self.balance}")  account1 = BankAccount("Alice", 500) account1.deposit(200) account1.withdraw(100)* |
| --- |

### ***18. Create a Book Class***

*📌 Create a class Book with attributes title and author. Add a method to display book details.*

| *class Book:  def \_\_init\_\_(self, title, author):  self.title = title  self.author = author   def display\_info(self):  print(f"Book: {self.title}, Author: {self.author}")  book1 = Book("1984", "George Orwell") book1.display\_info()* |
| --- |

### ***19. Create a Rectangle Class***

*📌 Create a class Rectangle with attributes length and width. Add a method area() and perimeter() to calculate the area and perimeter.*

| *class Rectangle:  def \_\_init\_\_(self, length, width):  self.length = length  self.width = width   def area(self):  return self.length \* self.width  rect1 = Rectangle(5, 3) print("Area of rectangle:", rect1.area())* |
| --- |

### ***20. Create a Shape Class and a Derived Circle Class***

*📌 Create a base class Shape with a method area(). Create a subclass Circle that overrides the method to return the area of a circle.*

| *import math  class Shape:  def area(self):  pass # Placeholder for area calculation  class Circle(Shape):  def \_\_init\_\_(self, radius):  self.radius = radius   def area(self):  return math.pi \* self.radius \*\* 2  circle1 = Circle(7) print("Area of circle:", circle1.area())* |
| --- |

### ***21. Create a Movie Class***

*📌 Create a class Movie with attributes title, genre, and rating. Add a method to display movie details.*



| *class Movie:  def \_\_init\_\_(self, title, genre, rating):  self.title = title  self.genre = genre  self.rating = rating   def show\_details(self):  print(f"Movie: {self.title}, Genre: {self.genre}, Rating: {self.rating}/10")  movie1 = Movie("Inception", "Sci-Fi", 8.8) movie1.show\_details()* |
| --- |

### ***22. Student and Teacher Classes***

*📌 Create a base class Person with attributes name and age. Create subclasses Student and Teacher with additional attributes (grade for Student and subject for Teacher).*

| *class Person:  def \_\_init\_\_(self, name, age):  self.name = name  self.age = age  class Student(Person):  def \_\_init\_\_(self, name, age, grade):  super().\_\_init\_\_(name, age)  self.grade = grade  class Teacher(Person):  def \_\_init\_\_(self, name, age, subject):  super().\_\_init\_\_(name, age)  self.subject = subject  student1 = Student("Alice", 16, "10th Grade") teacher1 = Teacher("Mr. Smith", 40, "Mathematics")  print(f"Student: {student1.name}, Age: {student1.age}, Grade: {student1.grade}") print(f"Teacher: {teacher1.name}, Age: {teacher1.age}, Subject: {teacher1.subject}")* |
| --- |

### ***23. E-Commerce Product Management***

*📌 Create a class Product with attributes name, price, and stock. Add a method purchase() that reduces stock if available.*

| *class Product:  def \_\_init\_\_(self, name, price, stock):  self.name = name  self.price = price  self.stock = stock   def purchase(self, quantity):  if quantity > self.stock:  print("Not enough stock available!")  else:  self.stock -= quantity  print(f"Purchased {quantity} {self.name}(s). Remaining stock: {self.stock}")  product1 = Product("Laptop", 60000, 5) product1.purchase(2)* |
| --- |

### ***24. Create a Traffic Light Simulation***

*📌 Create a class TrafficLight with attributes color. Add a method next\_light() to change between Red → Green → Yellow.*

| *class TrafficLight:  def \_\_init\_\_(self, color="Red"):  self.color = color   def next\_light(self):  if self.color == "Red":  self.color = "Green"  elif self.color == "Green":  self.color = "Yellow"  else:  self.color = "Red"  traffic = TrafficLight() print("Current:", traffic.color) traffic.next\_light() print("Next:", traffic.color)* |
| --- |

### ***25. Employee Salary Increment System***

*📌 Create a class Employee with attributes name and salary. Add a method apply\_raise() that increases the salary by a given percentage.*

| *class Employee:  def \_\_init\_\_(self, name, salary):  self.name = name  self.salary = salary   def apply\_raise(self, percentage):  self.salary += self.salary \* (percentage / 100)  print(f"{self.name}'s new salary: {self.salary}") emp1 = Employee("John", 50000) emp1.apply\_raise(10)* |
| --- |

### ***26. Age Calculator***

*📌 Create an age calculator where the user enters their birth year, clicks a button, and sees their age.*

| *import tkinter as tk from datetime import datetime  def calculate\_age():  birth\_year = int(entry.get())  current\_year = datetime.now().year  age = current\_year - birth\_year  result.config(text=f"You are {age} years old.")  root = tk.Tk() root.title("Age Calculator")  tk.Label(root, text="Enter Birth Year:").pack() entry = tk.Entry(root) entry.pack()  button = tk.Button(root, text="Calculate Age", command=calculate\_age) button.pack()  result = tk.Label(root, text="") result.pack()  root.mainloop()* |
| --- |

### ***27. Create a Simple "Hello, World!" Window***

*📌 Create a Tkinter window that displays "Hello, World!" in a label.*

| *import tkinter as tk  root = tk.Tk() root.title("Hello World App")  label = tk.Label(root, text="Hello, World!", font=("Arial", 16)) label.pack() root.mainloop()* |
| --- |

### ***28. Create a Simple Login Form***

*📌 Create a Tkinter form with entry fields for "Username" and "Password" and a "Login" button. Display entered username on login.*

| *import tkinter as tk  def login():  username = entry\_username.get()  label\_result.config(text=f"Welcome, {username}!")  root = tk.Tk() root.title("Login Form")  tk.Label(root, text="Username:").pack() entry\_username = tk.Entry(root) entry\_username.pack()  tk.Label(root, text="Password:").pack() entry\_password = tk.Entry(root, show="\*") entry\_password.pack()  button\_login = tk.Button(root, text="Login", command=login) button\_login.pack()  label\_result = tk.Label(root, text="") label\_result.pack()  root.mainloop()* |
| --- |

### ***29. Simple Counter App***

*📌 Create a GUI with a button that increases a counter each time it is clicked.*

| *import tkinter as tk  count = 0  def increase\_count():  global count  count += 1  label.config(text=f"Count: {count}")  root = tk.Tk() root.title("Counter App")  label = tk.Label(root, text="Count: 0", font=("Arial", 14)) label.pack()  button = tk.Button(root, text="Increase", command=increase\_count) button.pack()  root.mainloop()* |
| --- |

### ***30. Temperature Converter (Celsius to Fahrenheit)***

*📌 Create a GUI where the user enters a temperature in Celsius, clicks a button, and sees the converted Fahrenheit temperature.*

| *import tkinter as tk  def add\_task():  task = entry.get()  if task:  listbox.insert(tk.END, task)  entry.delete(0, tk.END)  root = tk.Tk() root.title("To-Do List")  entry = tk.Entry(root) entry.pack()  button = tk.Button(root, text="Add Task", command=add\_task) button.pack()  listbox = tk.Listbox(root) listbox.pack()  root.mainloop()* |
| --- |