**SAVITRIBAI PHULE PUNE UNIVERSITY  
(Formely University of Pune)**

**JSPM’s JAYAWANTRAO INSTITUTE OF MANAGEMENT STUDIES  
TATHAWADE, PUNE-33**

**M.C.A. - I**

**PYTHON LAB ASSIGNMENT**

**Semester 1**

|  |  |
| --- | --- |
| **Name of Student:** | **Manish Shetty** |
| **Roll No.** | **(H) 2560** |
| **Email:** | **manishnshetty77@gmail.com** |
| **Phone:** | **8208618905** |

**Subject Teacher/ GFM HOD**

**Academic Year 2025-27**

**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.No** | **Assignment -1** | **Date** | **Sign** |
| 1. | 1.1 Introduction  Case Study-1: College Admission System |  |  |
| 2. | 1.2 Keywords, Identifiers, Literals, Operators  Case Study-2: Employee Salary Calculation |  |  |
| 3. | 1.3 Data Types  Case Study-3: Student Data Management |  |  |
| 4. | 1.4 Python Blocks  Case Study-4: Banking Transactions |  |  |
| 5. | 1.5 Control Flow (if, else, elif)  Case Study-5: Electricity Bill Calculation |  |  |
| 6. | 1.6 Loops (while, for, break)  Case Study-6: Password Validation System |  |  |
| 7. | 1.7 Loop Manipulation using pass, continue  Case Study-7: Shopping Cart System |  |  |
| 8. | 1.8 For loop using ranges, list and dictionary  Case Study-8: Library Book Management |  |  |
| 9. | 1.9 Python Conditional and loops block  Case Study-9: ATM Cash Withdrawal |  |  |
| 10. | 1.10 Comprehensions on List, Tuple, Dictionaries  Case Study-10: Student Mark Processing |  |  |

**Assignment No-1**

**Subject:** Python Programming (PP) **Subject Teacher:** Prof. Leena Deshmukh HOD

**Topic:** Python Lab Assignment-1

**Name:** Manish Narayan Shetty **Roll No:** (H25)60 **Class:** MCA – Semester-I

**College Name:** JSPM’s Jayawantrao Institutes of Management Studies (JIMS)

**Submitted To:** **Submitted Date:**

**Case Study-1: College Admission System** **Topic: Introduction**

* Write a Python program that prints a welcome message for students applying to a college.
* Display details such as college name, program offered, and admission year using print() statements.

**Program/Code:**

print("Enter Yes or No");

study = input("Do you want to Apply for MCA at JSPM's JIMS College: ");

if study == "yes" or study == "YES" or study == "Yes":

print("Welcome to JSPM Group of Institutes :)");

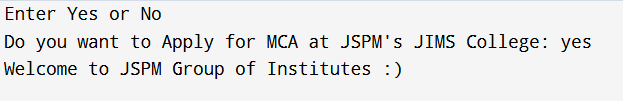
else:

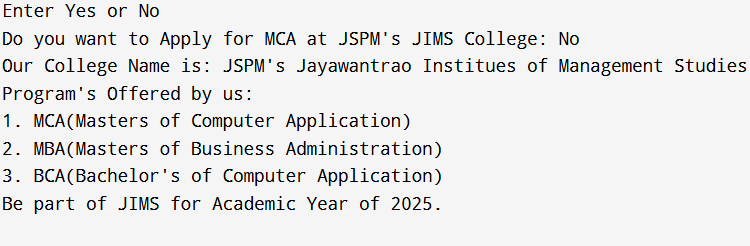
print("Our College Name is: JSPM's Jayawantrao Institues of Management Studies");

print("Program's Offered by us: \n1. MCA(Masters of Computer Application)\n2. MBA(Masters of Business Administration)\n3. BCA(Bachelor's of Computer Application)");

print("Be part of JIMS for Academic Year of 2025.");

**Output:**





**Case Study-2: Employee Salary Calculation**  **Topic: Keywords, Identifiers, Literals, Operators**

* Accept employee name, ID, and basic salary.
* Calculate Gross Salary = Basic + HRA + DA (use arithmetic operators).
* Ensure variable names follow Python identifier rules.
* Use literals to store fixed values like HRA = 0.2 \* Basic and DA = 0.5 \* Basic.

**Program/Code:**

employee\_name = input("Enter Employee Name: ");

employee\_id = input("Enter Employee ID: ");

basic\_salary = float(input("Enter Basic Salary: "));

print("\nCalculating the Allowances i.e. HRA with 0.2 and DA with 0.5 rate....");

HRA = 0.2 \* basic\_salary;

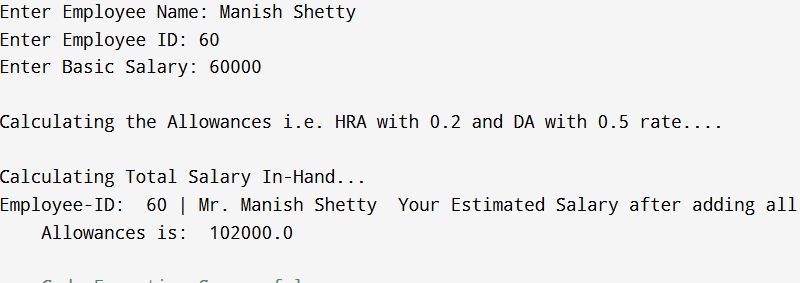
DA = 0.5 \* basic\_salary;

print("\nCalculating Total Salary In-Hand...");

salary = basic\_salary + HRA + DA;

print("Employee-ID: ",employee\_id,"| Mr.",employee\_name," Your Estimated Salary after adding all Allowances is: ",salary);

**Output:**

****

**Case Study-3: Student Data Management**

**Topic: Data Types (Numbers, Strings, Lists, Tuples, Dictionaries, Sets)**

* Store student details: name, roll number, marks in 5 subjects.
* Use list for marks, tuple for immutable details (roll number, DOB), dictionary for student profile, and set for storing unique subjects enrolled.
* Perform operations like finding average marks, highest score, and unique subjects.

**Program/Code:**

marks = [44,35,48,47,50];

details = (60,"01-01-2005");

subjects = {"Python","DS","Cloud","SEPM","Data"};

student\_profile = {

"Name": "Manish Shetty",

"Roll-no": details[0],

"Date of Birth": details[1],

"Marks": marks,

"Subjects": subjects

};

total\_marks = sum(student\_profile["Marks"]);

number\_of\_subjects = len(student\_profile["Marks"]);

average\_marks = total\_marks / number\_of\_subjects;

highest\_score = max(student\_profile["Marks"]);

is\_enrolled\_in\_BS = "Business" in student\_profile["Subjects"];

is\_enrolled\_in\_math = "Mathematics" in student\_profile["Subjects"];

# --- Display Results ---

print("Displaying Students Profile and Report...");

print("--- Student Profile ---");

print("Name: ",student\_profile['Name']);

print("Roll Number: ",student\_profile['Roll-no']);

print("Date of Birth: ",student\_profile['Date of Birth']);

print("-" \* 25)

print("\n--- Academic Details ---");

print("Marks: ",student\_profile['Marks']);

print("Unique Subjects : ",student\_profile['Subjects']);

print("\n--- Analytical Results ---");

print(f"Total Marks: {total\_marks}");

print(f"Average Marks: {average\_marks:.2f}");

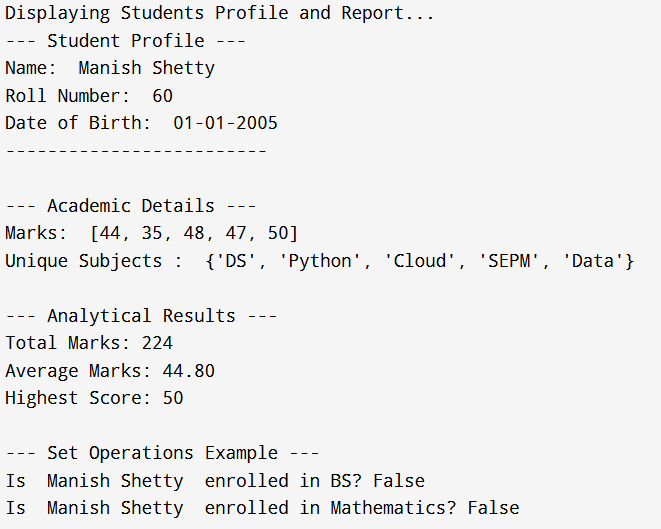
print(f"Highest Score: {highest\_score}");

print("\n--- Set Operations Example ---");

print("Is ",student\_profile["Name"]," enrolled in BS?", is\_enrolled\_in\_BS);

print("Is ",student\_profile["Name"]," enrolled in Mathematics?", is\_enrolled\_in\_math);

**Output:**

****

**Case Study-4: Banking Transactions**  **Topic: Understanding Python Blocks**

* Write a program to check if a customer can withdraw money.
* Use proper indentation and code blocks.
* Example:
* If balance ≥ withdrawal amount → show success message.
* Else → display insufficient balance.

**Program/Code:**

current\_balance = float(input("Enter your Current Account Balance (Rs.): "));

withdrawal\_amount = float(input("Enter Withdrawal Amount (Rs.): "));

print("\n--- Processing Transaction ---");

if current\_balance >= withdrawal\_amount:

new\_balance = current\_balance - withdrawal\_amount;

print("Transaction Successful!");

print("Amount Withdrawn: Rs. ",withdrawal\_amount);

print("New Balance: Rs. ",new\_balance);

else:

print("Transaction Failed.");

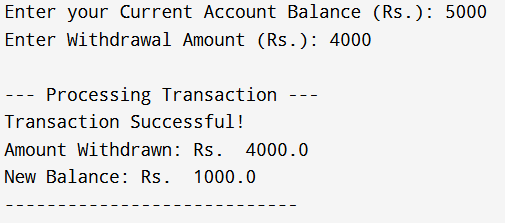
print("Insufficient Balance: Your current balance is too low for this withdrawal.");

print("Current Balance: Rs. ",current\_balance);

print("Requested Withdrawal: Rs. ",withdrawal\_amount);

print("----------------------------");

**Output:**

****

**Case Study-5: Electricity Bill Calculation**  **Topic: Control Flow (if, else, elif)**

* Input units consumed.
* Apply conditions:
* 0-100 units: ₹5/unit
* 101-300 units: ₹7/unit
* >300 units: ₹10/unit
* Display the total bill.

**Program/Code:**

units\_consumed = int(input("Enter total units consumed: "));

if units\_consumed <= 100:

bill = units\_consumed \* 5;

elif units\_consumed <= 300:

bill = (100 \* 5) + (units\_consumed - 100) \* 7;

else:

bill = (100 \* 5) + (200 \* 7) + (units\_consumed - 300) \* 10;

print("Total Electricity Bill = ₹", bill);

**Output:**

C:\Users\Manish Shetty\OneDrive\Desktop\JSPM\Python\Assignment-1>python CaseStudy5.py

Enter total units consumed: 130

Total Electricity Bill = ₹ 710

**Case Study-6: Password Validation System** **Topic: Loops (while, for, continue, break)**

* Allow the user 3 attempts to enter the correct password.
* If the password matches → print “Login Successful” and break the loop.
* If all attempts fail → print “Account Locked”.

**Program/Code**:

correct\_password = "Manish@12";

attempts = 3;

for i in range(attempts):

entered = input("Enter password: ");

if entered == correct\_password:

print("Login Successful");

break;

else:

print("Incorrect password. Attempts left:", attempts - i - 1);

else:

print("Account Locked");

**Output:**

C:\Users\Manish Shetty\OneDrive\Desktop\JSPM\Python\Assignment-1>python CaseStudy6.py

Enter password: Manish@12

Login Successful

C:\Users\Manish Shetty\OneDrive\Desktop\JSPM\Python\Assignment-1>python CaseStudy6.py

Enter password: Manish@123

Incorrect password. Attempts left: 2

Enter password: Man

Incorrect password. Attempts left: 1

Enter password: man

Incorrect password. Attempts left: 0

Account Locked

**Case Study-7: Shopping Cart System**

**Topic: Loop Manipulation using pass, continue, break and else**

* Given a list of products with prices.
* Skip products with price 0 using continue.
* If product is “Exit” → stop scanning using break.
* If all items are scanned successfully → print a message from the else block.
* Use pass for future discount implementation.

**Program/Code:**

products = [("Apple", 50), ("Banana", 0), ("Milk", 30), ("Orange", 20), ("Bread", 25)];

total = 0;

for item, price in products:

if price == 0 and item != "Exit":

print("Skipping ",item," (Price is 0)");

continue;

if item == "Exit":

print("Exit found! Stopping cart scan...");

break;

pass;

print("Adding ",item," - ₹",price);

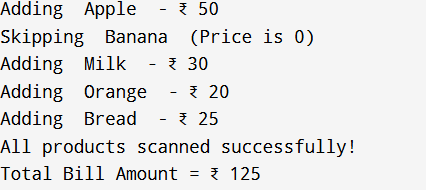
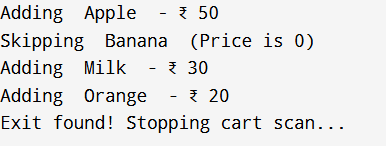
total += price;

else:

print("All products scanned successfully!");

print("Total Bill Amount = ₹", total);

**Output:**

**** ****

**Case Study-8: Library Book Management**

**Topic: For Loop using ranges, string, list and dictionaries**

* Use range to generate unique book IDs (101–110).
* Traverse a string to check if the book title contains vowels.
* Traverse a list of book titles to display available books.
* Traverse a dictionary with {Book: Author} to display book-author pairs.

**Program/Code:**

book\_ids = list(range(101, 111))

print("Generated Book IDs:", book\_ids)

print("\n--- Vowel Check in Book Title ---")

book\_title = "Python Programming"

vowels = "aeiouAEIOU"

contains\_vowel = False

for ch in book\_title:

if ch in vowels:

contains\_vowel = True

break

if contains\_vowel:

print("The book title ",book\_title," contains vowels.")

else:

print("The book title ",book\_title," does NOT contain vowels.")

print("\n--- List of Available Books ---")

books = ["Python Programming", "Data Structures", "Cloud", "SEPM"]

for book in books:

print(book)

print("\n--- Book-Author Pairs ---")

book\_authors = {

"Python Programming": "Prof. Leena Deshmukh",

"Data Structures": "Prof. Rajesh Jadav",

"Cloud": "Prof. Nikita Phalak",

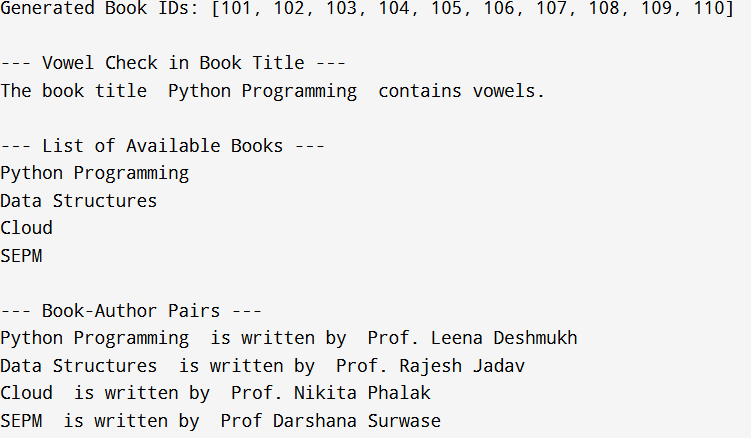
"SEPM": "Prof Darshana Surwase"

}

for book, author in book\_authors.items():

print(book," is written by ",author)

**Output:**

****

**Case Study-9: ATM Cash Withdrawal Simulation**

**Topic: Programming using Python Conditional and Loops Block**

* Input the withdrawal amount.
* Check if the balance is sufficient.
* If amount is not multiple of 100 → show an error.
* Use loops to allow multiple transactions until the user exits.

**Program/Code:**

balance = int(input("Please Enter Your Balance Amount that you want to credit: "))

print("=== Welcome to JSPM's ATM ===")

while True:

print("\nYour Current Balance: ₹",balance)

choice = input("Do you want to withdraw? (yes/no): ").lower()

if choice == "no":

print("Thank you for using our ATM. Goodbye!")

break

amount = int(input("Enter withdrawal amount: ₹"))

if amount % 100 != 0:

print("Error: Please enter amount in multiples of 100.")

continue

if amount > balance:

print("Error: Insufficient balance.")

continue

balance -= amount

print("Transaction Successful! You withdrew ₹",amount)

print("Remaining Balance: ₹",balance)

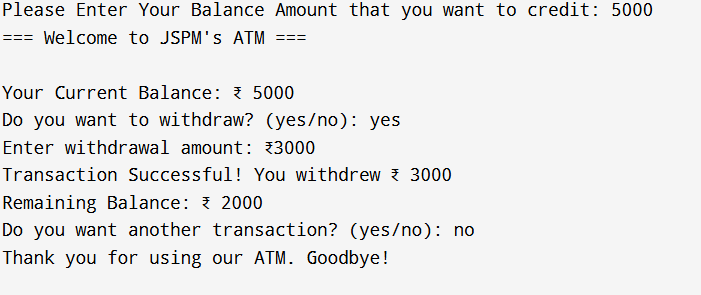
more = input("Do you want another transaction? (yes/no): ").lower()

if more == "no":

print("Thank you for using our ATM. Goodbye!")

break

**Output:**



**Case Study-10: Student Marks Processing** **Topic: Comprehensions on List, Tuple, Dictionaries**

* Given a list of marks of students.
* Use list comprehension to find marks > 40 (pass students).
* Use dictionary comprehension to create {student\_name: grade}.
* Use set comprehension to find unique grades.
* Use tuple comprehension (generator expression) to store squares of marks.

**Program/Code:**

# Student data

students = ["Ravi", "Neha", "Amit", "Priya", "Kiran", "Meena", "Arjun"]

marks = [35, 75, 40, 90, 55, 30, 88]

# 1. List comprehension → Students who passed (marks > 40)

passed\_marks = [m for m in marks if m > 40]

print("Marks of passed students:", passed\_marks)

# 2. Dictionary comprehension → {student\_name: grade}

grades = {

student: ("A" if mark >= 75 else

"B" if mark >= 60 else

"C" if mark >= 40 else

"F")

for student, mark in zip(students, marks)

}

print("\nStudent Grades:", grades)

# 3. Set comprehension → Unique grades

unique\_grades = {grade for grade in grades.values()}

print("\nUnique Grades:", unique\_grades)

# 4. Tuple comprehension (actually generator expression) → Squares of marks

squares\_gen = (m\*\*2 for m in marks) # generator expression

squares\_tuple = tuple(squares\_gen) # converting to tuple

print("\nSquares of Marks (Tuple):", squares\_tuple)

**Output:**

Marks of passed students: [75, 90, 55, 88]

Student Grades: {'Ravi': 'F', 'Neha': 'A', 'Amit': 'C', 'Priya': 'A', 'Kiran': 'C', 'Meena': 'F', 'Arjun': 'A'}

Unique Grades: {'C', 'F', 'A'}

Squares of Marks (Tuple): (1225, 5625, 1600, 8100, 3025, 900, 7744)