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

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

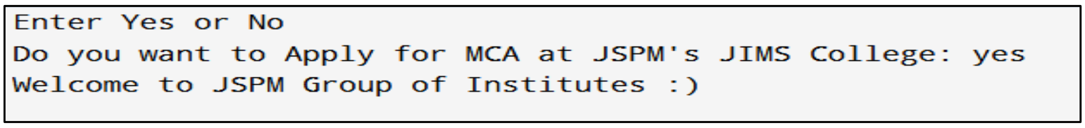
Topic: Introduction

- 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.

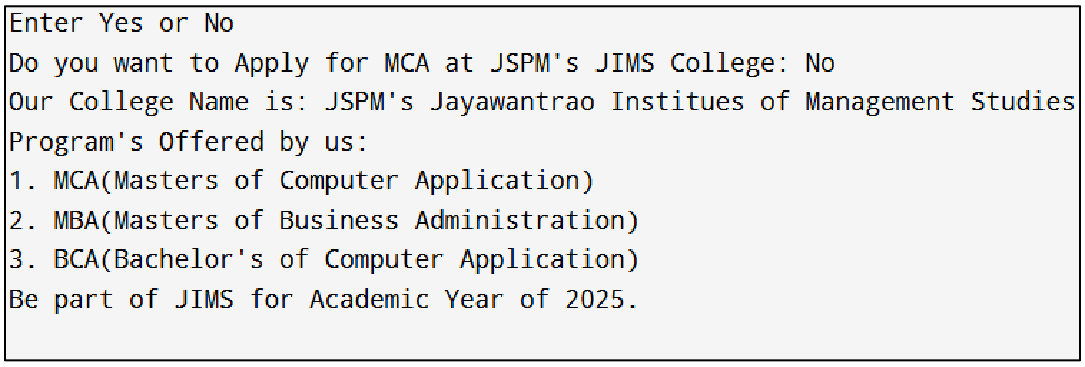
Program/Code:

```
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:



```
Enter Yes or No
Do you want to Apply for MCA at JSPM's JIMS College: yes
Welcome to JSPM Group of Institutes :)
```



```
Enter Yes or No
Do you want to Apply for MCA at JSPM's JIMS College: No
Our College Name is: JSPM's Jayawantrao Institues of Management Studies
Program's Offered by us:
1. MCA(Masters of Computer Application)
2. MBA(Masters of Business Administration)
3. BCA(Bachelor's of Computer Application)
Be part of JIMS for Academic Year of 2025.
```

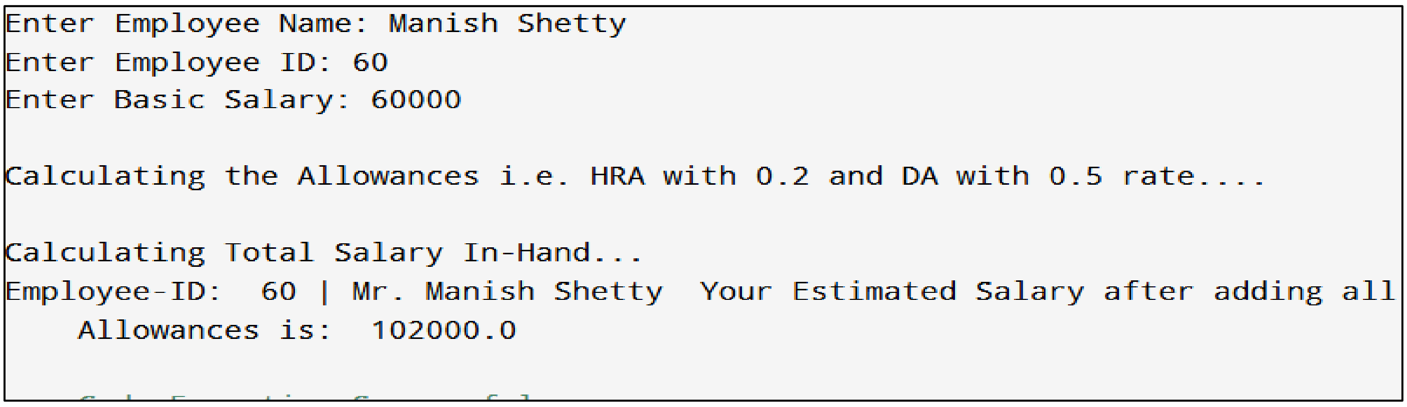
Case Study-2: Employee Salary Calculation **Topic: Keywords, Identifiers, Literals, Operators**

- 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:



```
Enter Employee Name: Manish Shetty
Enter Employee ID: 60
Enter Basic Salary: 60000

Calculating the Allowances i.e. HRA with 0.2 and DA with 0.5 rate....

Calculating Total Salary In-Hand...
Employee-ID:  60 | Mr. Manish Shetty  Your Estimated Salary after adding all
   Allowances is:  102000.0
```

Case Study-3: Student Data Management

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

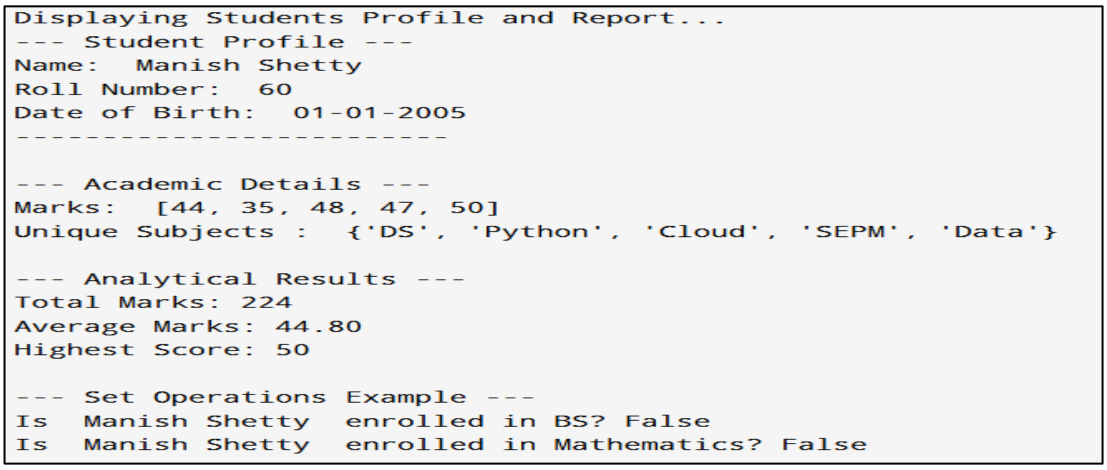
- Store student details: name, roll number, marks in 5 subjects.
- Use list for marks, tuple for immutable details (roll number, DOB), dictionary for student profile, and set for storing unique subjects enrolled.
- Perform operations like finding average marks, highest score, and unique subjects.

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:



```
Displaying Students Profile and Report...
--- Student Profile ---
Name:  Manish Shetty
Roll Number:  60
Date of Birth:  01-01-2005
-------------------------

--- Academic Details ---
Marks:  [44, 35, 48, 47, 50]
Unique Subjects :  {'DS', 'Python', 'Cloud', 'SEPM', 'Data'}

--- Analytical Results ---
Total Marks: 224
Average Marks: 44.80
Highest Score: 50

--- Set Operations Example ---
Is  Manish Shetty  enrolled in BS? False
Is  Manish Shetty  enrolled in Mathematics? False
```

<u>Case Study-4: Banking Transactions</u> **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:

```
Enter your Current Account Balance (Rs.): 5000
Enter Withdrawal Amount (Rs.): 4000

--- Processing Transaction ---
Transaction Successful!
Amount Withdrawn: Rs.  4000.0
New Balance: Rs.  1000.0
----------------------------
```

<u>Case Study-5: Electricity Bill Calculation</u> **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

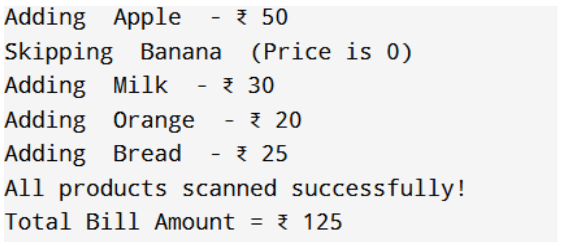
Topic: Loop Manipulation using pass, continue, break and else

- 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.

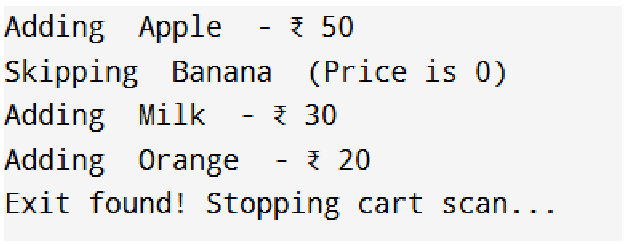
Program/Code:

```
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:



```
Adding  Apple  - ₹ 50
Skipping  Banana  (Price is 0)
Adding  Milk  - ₹ 30
Adding  Orange  - ₹ 20
Adding  Bread  - ₹ 25
All products scanned successfully!
Total Bill Amount = ₹ 125
```



```
Adding  Apple  - ₹ 50
Skipping  Banana  (Price is 0)
Adding  Milk  - ₹ 30
Adding  Orange  - ₹ 20
Exit found! Stopping cart scan...
```

Case Study-8: Library Book Management

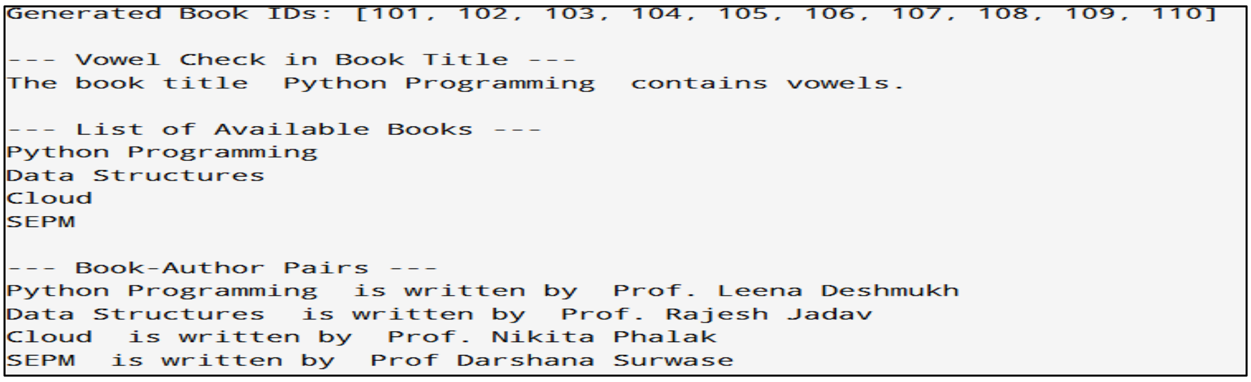
Topic: For Loop using ranges, string, list and dictionaries

- 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:



```
Generated Book IDs: [101, 102, 103, 104, 105, 106, 107, 108, 109, 110]

--- Vowel Check in Book Title ---
The book title  Python Programming  contains vowels.

--- List of Available Books ---
Python Programming
Data Structures
Cloud
SEPM

--- Book-Author Pairs ---
Python Programming  is written by  Prof. Leena Deshmukh
Data Structures  is written by  Prof. Rajesh Jadav
Cloud  is written by  Prof. Nikita Phalak
SEPM  is written by  Prof Darshana Surwase
```

<u>**Case Study-9: ATM Cash Withdrawal Simulation**</u>

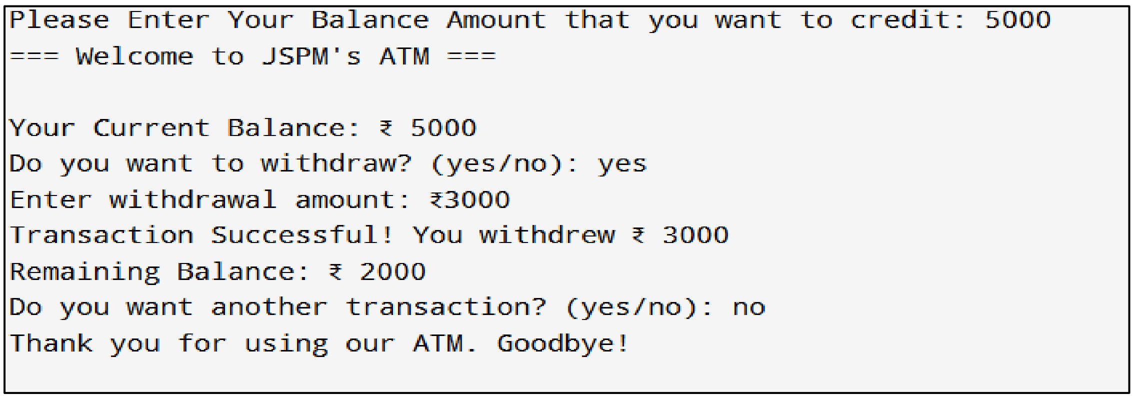
Topic: Programming using Python Conditional and Loops Block

- 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:



```
Please Enter Your Balance Amount that you want to credit: 5000
=== Welcome to JSPM's ATM ===

Your Current Balance: ₹ 5000
Do you want to withdraw? (yes/no): yes
Enter withdrawal amount: ₹3000
Transaction Successful! You withdrew ₹ 3000
Remaining Balance: ₹ 2000
Do you want another transaction? (yes/no): no
Thank you for using our ATM. Goodbye!
```

<u>**Case Study-10: Student Marks Processing**</u> **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)
```