

PYTHON PROGRAMMING LAB



Prepared by:

Name of Student: _Shikha singh_____

Roll No: _25_____

Batch: 2023-27

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Ex p. No	List of Experiment
1	1.1 Write a program to compute Simple Interest.

	1.2 Write a program to perform arithmetic, Relational operators.
	1.3 Write a program to find whether a given no is even & odd.
	1.4 Write a program to print first n natural number & their sum.
	1.5 Write a program to determine whether the character entered is a Vowel or not .
	1.6 Write a program to find whether given number is an Armstrong Number.
	1.7 Write a program using for loop to calculate factorial of a No.
	1.8 Write a program to print the following pattern
	i) * * * * * * * * * * * * * * *
	ii) 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5
	iii) *

2	2.1 Write a program that define the list of defines the list of define countries that are in BRICS.
	2.2 Write a program to traverse a list in reverse order. 1.By using Reverse method. 2.By using slicing
	2.3 Write a program that scans the email address and forms a tuple of username and domain.
	2.4 Write a program to create a list of tuples from given list having number and add its cube in tuple. i/p: c= [2,3,4,5,6,7,8,9]
	2.5 Write a program to compare two dictionaries in Python? (By using == operator)
	2.6 Write a program that creates dictionary of cube of odd numbers in the range.
	2.7 Write a program for various list slicing operation. a= [10,20,30,40,50,60,70,80,90,100] i. Print Complete list ii. Print 4th element of list iii. Print list from 0th to 4th index. iv. Print list -7th to 3rd element v. Appending an element to list. vi. Sorting the element of list. vii. Popping an element. viii. Removing Specified element. ix. Entering an element at specified index. x. Counting the occurrence of a specified element. xi. Extending list. xii. Reversing the list.
3	3.1 Write a program to extend a list in python by using given approach. i. By using + operator. ii. By using Append () iii. By using extend ()

	3.2 Write a program to add two matrices.
	3.3 Write a Python function that takes a list and returns a new list with distinct elements from the first list.
	3.4 Write a program to Check whether a number is perfect or not.
	3.5 Write a Python function that accepts a string and counts the number of upper and lower-case letters. string_test= 'Today is My Best Day'
4	4.1 Write a program to Create Employee Class & add methods to get employee details & print.
	4.2 Write a program to take input as name, email & age from user using combination of keywords argument and positional arguments (*args and **kwargs) using function,
	4.3 Write a program to admit the students in the different Departments(pgdm/btech)and count the students. (Class, Object and Constructor).
	4.4 Write a program that has a class store which keeps the record of code and price of product display the menu of all product and prompt to enter the quantity of each item required and finally generate the bill and display the total amount.
	4.5 Write a program to take input from user for addition of two numbers using (single inheritance).
	4.6 Write a program to create two base classes LU and ITM and one derived class. (Multiple inheritance).
	4.7 Write a program to implement Multilevel inheritance, Grandfather → Father- → Child to show property inheritance from grandfather to child.

	4.8 Write a program Design the Library catalogue system using inheritance take base class (library item) and derived class (Book, DVD & Journal) Each derived class should have unique attribute and methods and system should support Check in and check out the system. (Using Inheritance and Method overriding)
--	---

5	5.1 Write a program to create my_module for addition of two numbers and import it in main script.
	5.2 Write a program to create the Bank Module to perform the operations such as Check the Balance, withdraw and deposit the money in bank account and import the module in main file.
	5.3 Write a program to create a package with name cars and add different modules (such as BMW, AUDI, NISSAN) having classes and functionality and import them in main file cars.
6	6.1 Write a program to implement Multithreading. Printing “Hello” with one thread & printing “Hi” with another thread.
7.	7.1 Write a program to use ‘whether API’ and print temperature of any city, also print the sunrise and sunset times for the same humidity of that area.
	7.2 Write a program to use the ‘API’ of crypto currency.

Name of Student: Shikha singh

_____ **Roll Number: 25**

_____ **Experiment No:**

5.3

Title: 5.3 Write a program to create a package with name cars and add different modules (such as BMW, AUDI, NISSAN) having classes and functionality and import them in main file cars.

Theory:

Packages in Python:

- A package is a way of organizing related modules into a single directory hierarchy.
- In this case, the cars package is created to organize modules related to different car brands (e.g., BMW, AUDI, NISSAN).
- **Modules and Classes:**
 - Each car brand is represented by a separate module (BMW.py, AUDI.py, NISSAN.py).
 - Each module contains a class definition (e.g., BMW, AUDI, NISSAN) with associated functionalities.
- **Class Instantiation and Functionality:**
 - Instances of each class are created in the main.py file, demonstrating the creation of objects for different car brands.
 - Each class has an `__init__` method for initializing attributes (model, color) and a method (display_info) to display information about the car.
- **Importing Modules:**
 - In the main.py file, modules are imported using the `from cars.<module> import <class>` syntax.
 - This allows access to the classes and functionalities defined in each module.
- **Program Execution:**
 - The `main()` function in main.py demonstrates the creation of instances for each car brand and calls the `display_info` method to showcase their information.

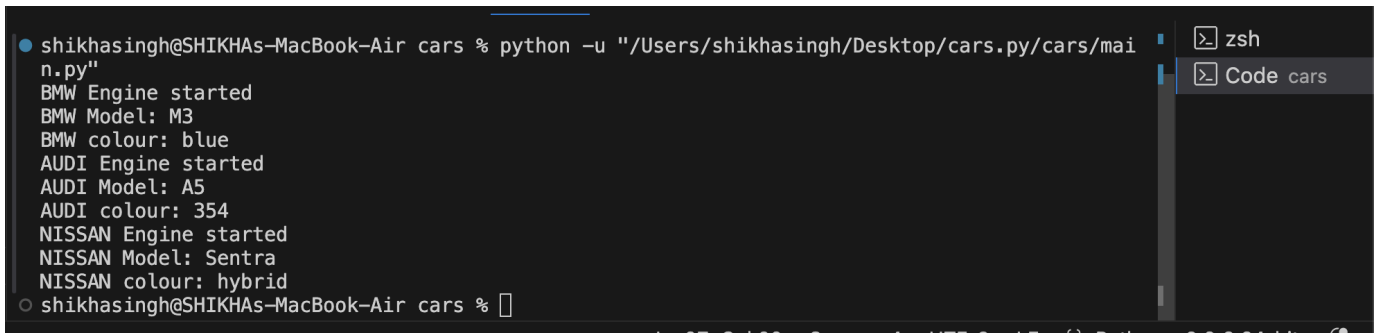
Code: # AUDI.py

```
class AUDI:
    def __init__(self, model, colour):
        self.model = model
        self.colour = colour
    def display_model(self):
        print(f"AUDI Model: {self.model}")
        print(f"AUDI colour: {self.colour}")
    def start_engine(self):
        print("AUDI Engine started")

# BMW.py
class BMW:
    def __init__(self, model, colour):
        self.model = model
        self.colour = colour
```

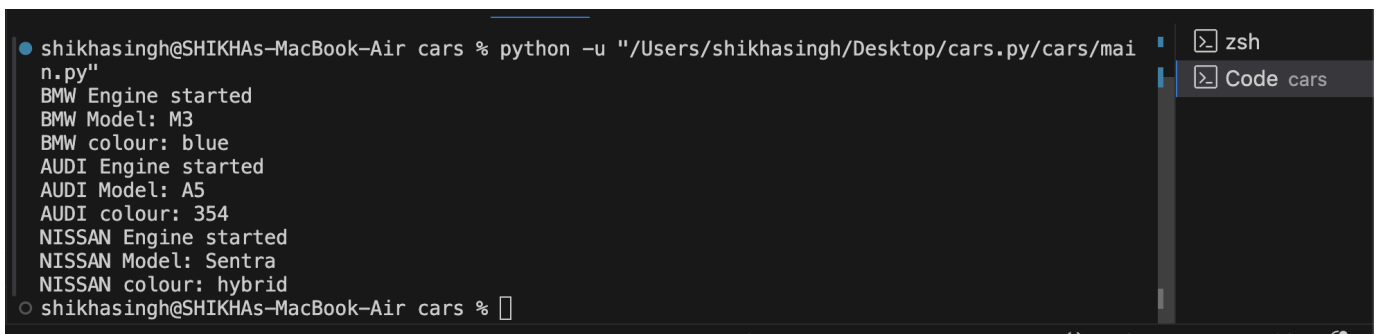
```
def display_model(self):
    print(f"BMW Model: {self.model}")
    print(f"BMW colour: {self.colour}")
def start_engine(self):
    print("BMW Engine started")
# NISSAN.py
class NISSAN:
    def __init__(self, model, colour):
        self.model = model
        self.colour = colour
    def display_model(self):
        print(f"NISSAN Model: {self.model}")
        print(f"NISSAN colour: {self.colour}")
    def start_engine(self):
        print("NISSAN Engine started")
from BMW import BMW
from AUDI import AUDI
from NISSAN import NISSAN
bmw = BMW("M3", "blue")
audi = AUDI("A5", 354)
nissan = NISSAN("Sentra", "hybrid")
bmw.start_engine()
bmw.display_model()
audi.start_engine()
audi.display_model()
nissan.start_engine()
nissan.display_model()
```

Output: (screenshot)



```
shikhasingh@SHIKHAS-MacBook-Air cars % python -u "/Users/shikhasingh/Desktop/cars.py/cars/main.py"
BMW Engine started
BMW Model: M3
BMW colour: blue
AUDI Engine started
AUDI Model: A5
AUDI colour: 354
NISSAN Engine started
NISSAN Model: Sentra
NISSAN colour: hybrid
shikhasingh@SHIKHAS-MacBook-Air cars %
```

Test Case: Any two (screenshot)



```
shikhasingh@SHIKHAS-MacBook-Air cars % python -u "/Users/shikhasingh/Desktop/cars.py/cars/main.py"
BMW Engine started
BMW Model: M3
BMW colour: blue
AUDI Engine started
AUDI Model: A5
AUDI colour: 354
NISSAN Engine started
NISSAN Model: Sentra
NISSAN colour: hybrid
shikhasingh@SHIKHAS-MacBook-Air cars %
```

Conclusion: The modular structure

allows for a cleaner organization of code,

making it easier to manage and understand.

- Creating a package (cars) facilitates grouping related modules together.
- Importing classes from modules allows for the reuse of code and separates concerns.
- The main.py file serves as the entry point and demonstrates the usage of classes from different modules within the cars package.
- This approach enhances code readability, maintainability, and scalability by encapsulating related functionalities within dedicated modules.

