PYHTON SESSIONAL
ASSIGNMENT 2

Q.1 List Manipulation: Shopping Cart

- Create an empty list called shopping_cart.
- Ask the user to input three items one by one and add each to the list.
- Print the final list.
- Then, remove the second item and print the updated list.

#CODE START

```
shopping_cart = []
for i in range(3):
    item = input(f"Enter item {i+1}: ")
    shopping_cart.append(item)
print("Final Shopping Cart:", shopping_cart)
if len(shopping_cart) >= 2:
                    # Remove the second item (index 1)
  removed_item = shopping_cart.pop(1)
    print(f"Removed: {removed_item}")
                # Print the updated list
print("Updated Shopping Cart:", shopping_cart)
```

OUTPUT

```
Enter item 1: apple
Enter item 2: banana
Enter item 3: orange
Final Shopping Cart: ['apple', 'banana', 'orange']
Removed: banana
Updated Shopping Cart: ['apple', 'orange']
```

Q.2 Tuple Basics: Student Information

- Create a tuple named student that contains a student's first name, last name, and age (e.g., ('John', 'Doe', 19)).
- Print the entire tuple.
- Print only the student's last name using indexing.

#CODE SATRT

```
# Create a tuple with student information

student = ('John', 'Doe', 19)


# Print the entire tuple

print("Student Information:", student)


# Print only the student's last name

print("Last Name:", student[1])
```

OUTPUT

Student Information: ('John', 'Doe', 19)

Last Name: Doe

Q.3 Dictionary: Phone Book

- Create an empty dictionary called phone_book.
- Add three key-value pairs to it, where the key is a person's name and the value is their phone number.
- Print the entire dictionary.
- Ask the user for a name and print that person's phone number. Handle the case where the name is not found.

#CODE START

```
# Create an empty dictionary

phone_book = {}

phone_book["Alice"] = "9876543210"

phone_book["Bob"] = "9123456780"

phone_book["Charlie"] = "9988776655"

print("Phone Book:", phone_book)

name = input("Enter a name to search: ")

if name in phone_book:

   print(f"{name}'s phone number is: {phone_book[name]}")
```

```
else:
    print("Name not found in the phone book.")
```

output

Enter a name to search: Bob

Bob's phone number is: 9123456780

Q.4 List of Tuples: Movie Collection

- Create a list named movies where each element is a tuple. Each tuple should contain a movie title and its release year (e.g., [('Inception', 2010), ('Toy Story', 1995)]).
- Add a new movie tuple to the list.
- Loop through the list and print each movie's title and year in a formatted string.

#CODE START

```
movies = [('Inception', 2010), ('Toy Story', 1995), ('The Dark Knight', 2008)]
movies.append(('Interstellar', 2014))
for movie in movies:
    title, year = movie
    print(f"Movie: {title}, Year: {year}")
```

OUTPUT

Movie: Inception, Year: 2010

Movie: Toy Story, Year: 1995

Movie: The Dark Knight, Year: 2008

Movie: Interstellar, Year: 2014

Q.5 Dictionary Nesting: Employee Directory

- Create a dictionary named employee that stores information for one employee. It should have the keys: 'name', 'department', and 'skills'.
- The value for 'skills' should be a *list* of programming languages the employee knows (e.g., ['Python', 'SQL']).
- Print the employee's name and their second skill.

#code start

```
employee = {
    'name': 'Alice Johnson',
```

```
    'department': 'IT',
   'skills': ['Python', 'SQL', 'Java']
}

# Print the employee's name and their second skill
print(f"Employee Name: {employee['name']}")
print(f"Second Skill: {employee['skills'][1]}")
```

output

Employee Name: Alice Johnson

Second Skill: SQL

Q.6 Converting Data Types

- Create a list of names: names_list = ['Alice', 'Bob', 'Charlie'].
- Convert this list into a tuple.
- Try to change the first element of the new tuple to 'David'. Observe and note what happens.

#code start

```
names_list = ['Alice', 'Bob', 'Charlie']

names_tuple = tuple(names_list)

print("Tuple:", names_tuple)
try:
   names_tuple[0] = 'David'
except TypeError as e:
   print("Observation:", e)
```

output

Tuple: ('Alice', 'Bob', 'Charlie')

Observation: 'tuple' object does not support item assignment

Q.7 Membership Testing (Lists & Dictionaries)

- Create a list fruits = ['apple', 'banana', 'mango'].
- Ask the user to input a fruit name.

- Check if the fruit is in the list and print an appropriate message (e.g., "Yes, we have bananas!").
- Now, check for the same fruit in the *keys* of this dictionary: prices = {'apple': 1.0, 'banana': 0.5, 'orange': 0.75}.

#code start

```
fruits = ['apple', 'banana', 'mango']

prices = {'apple': 1.0, 'banana': 0.5, 'orange': 0.75}

fruit_name = input("Enter a fruit name: ").lower()


if fruit_name in fruits:
    print(f"Yes, we have {fruit_name}!")
else:
    print(f"Sorry, {fruit_name} is not in the list.")
if fruit_name in prices:
    print(f"{fruit_name.capitalize()} costs ${prices[fruit_name]}")
else:
    print(f"Price for {fruit_name} is not available.")
```

output

case 1

Enter a fruit name: apple

Yes, we have apple!

Apple costs $1.0

Case 2

Enter a fruit name: orange

Sorry, orange is not in the list.

Orange costs $0.75

Case 3

Enter a fruit name: mango

Yes, we have mango!

Price for mango is not available.

Q. 8. Combining Lists and Dictionaries: Gradebook

- Create a dictionary called grades where the key is a student's name and the value is a *list* of their test scores (e.g., {'Alice': [85, 90, 78], 'Bob': [92, 88, 95]}).
- Calculate and print the average test score for a specific student (e.g., Alice). sum(list) / len(list)

#code start

```
grades = {
   'Alice': [85, 90, 78],
   'Bob': [92, 88, 95],
   'Charlie': [70, 75, 80]
}
student_name = 'Alice'


# Calculate average score
if student_name in grades:
   scores = grades[student_name]
   average = sum(scores) / len(scores)
   print(f"Average score of {student_name}: {average:.2f}")
else:
   print(f"{student_name} not found in the grade book.")
```

output

```
Average score of Alice: 84.33
```

Q. 9 Tuple Unpacking: Coordinates

- Create a tuple called point that represents a 2D coordinate (e.g., (5, -3)).
- Use tuple unpacking to assign the values to two variables, x and y.
- Print the values of x and y and then calculate the distance from the origin (0, 0) using the formula: (x**2 + y**2)**0.5.

#code start

```
point = (5, -3)
x, y = point
print(f"x = {x}, y = {y}")
distance = (x**2 + y**2) ** 0.5
```

```
print(f"Distance from origin = {distance:.2f}")
```

output

x = 5, y = -3

Distance from origin = 5.83

Q.10 Comprehensive Practice: Restaurant Order

* Create a *dictionary* called menu where the keys are item names and the values are their prices.
* Create an empty *list* called order.
* Simulate a customer ordering by asking them to input two item names from the menu. Add each item to the order list.
* Calculate the total cost of the order by looking up each item in the order list within

Output

Case 1

Enter your first item: salad

Enter your second item: pasta

Your order: ['salad', 'pasta']

Total cost: $11.50

Case 2

Enter your first item: cookies

Enter your second item: campa energy drink

Sorry, cookies is not available.

Sorry, campa energy drink is not available.

Your order: ['cookies', 'campa energy drink']

Total cost: $0.00