

UNIT-III Sample Experiments:

1. Create tuples for at least two members, concatenate and print

Problem

Create tuples containing:

(name, age, address, college)

for at least two members, concatenate the tuples, and print the result.

Program

```
# Create tuple for first member
```

```
member1 = ("Ravi", 20, "Vijayawada", "LBR College of Engineering ")
```

```
# Create tuple for second member
```

```
member2 = ("Sita", 21, "Mylavaram", "Siddartha College of Engineering")
```

```
# Concatenate the two tuples
```

```
combined = member1 + member2
```

```
# Print the tuples
```

```
print("Member 1:", member1)
```

```
print("Member 2:", member2)
```

```
print("Concatenated Tuple:", combined)
```

Output

```
Member 1: ('Ravi', 20, 'Vijayawada', 'LBR College of Engineering')
```

```
Member 2: ('Sita', 21, 'Mylavaram', 'Siddartha College of Engineering')
```

```
Concatenated Tuple: ('Ravi', 20, 'Hyderabad', 'LBR College of Engineering ', 'Sita', 21, 'Mylavaram', 'Siddartha College of Engineering ')
```

Important Concept

Tuple concatenation is performed using +.

member1 + member2

For example:

(1, 2) + (3, 4)

produces:

(1, 2, 3, 4)

Note

A tuple is **immutable**, so after creating it, individual elements cannot be changed.

2. Count the number of vowels in a string — No Control Flow Allowed

The important condition here is:

No control flow allowed

So we should **not use**:

if

for

while

We can use the count() string method.

Program

```
# Read a string
```

```
string = input("Enter a string: ")
```

```
# Count vowels
```

```
vowels = (  
    string.lower().count('a') +  
    string.lower().count('e') +  
    string.lower().count('i') +  
    string.lower().count('o') +  
    string.lower().count('u')  
)
```

```
print("Number of vowels =", vowels)
```

Sample Input

Enter a string: Hello Python

Output

Number of vowels = 3

Explanation

For:

Hello Python

The vowels are:

e

o

o

Therefore:

Number of vowels = 3

The expression:

```
string.lower().count('a')
```

counts the number of a characters.

Similarly:

```
string.lower().count('e')
```

```
string.lower().count('i')
```

```
string.lower().count('o')
```

```
string.lower().count('u')
```

count the other vowels.

Why lower()?

It converts uppercase letters to lowercase.

For example:

```
"HELLO".lower()
```

gives:

hello

Therefore, both uppercase and lowercase vowels are counted.

3. Check if a Given Key Exists in a Dictionary

Program

```
# Create a dictionary
```

```
student = {  
    "name": "Ravi",  
    "age": 20,  
    "branch": "CSE",  
    "college": "LBR College of Engineering"  
}
```

```
# Read the key
```

```
key = input("Enter the key to search: ")
```

```
# Check whether key exists
```

```
if key in student:
```

```
    print("Key exists in the dictionary")
```

```
else:
```

```
    print("Key does not exist in the dictionary")
```

Sample Input

Enter the key to search: age

Output

Key exists in the dictionary

Another Example

Input:

Enter the key to search: salary

Output:

Key does not exist in the dictionary

Important Concept

The in operator checks whether a key exists in a dictionary.

if key in student:

For example:

```
student = {  
    "name": "Ravi",  
    "age": 20  
}
```

```
print("name" in student)
```

Output:

True

4. Add a New Key-Value Pair to an Existing Dictionary

Program

Existing dictionary

```
student = {  
    "name": "Ravi",  
    "age": 20,  
    "branch": "CSE"  
}
```

```
print("Original Dictionary:")
```

```
print(student)
```

```
# Add a new key-value pair
student["college"] = "LBR College of Engineering"
```

```
print("Updated Dictionary:")
```

```
print(student)
```

Output

Original Dictionary:

```
{'name': 'Ravi', 'age': 20, 'branch': 'CSE'}
```

Updated Dictionary:

```
{'name': 'Ravi', 'age': 20, 'branch': 'CSE', 'college': 'LBR College of Engineering'}
```

User Input Version

```
student = {
```

```
    "name": "Ravi",
```

```
    "age": 20
```

```
}
```

```
key = input("Enter new key: ")
```

```
value = input("Enter value: ")
```

```
student[key] = value
```

```
print("Updated Dictionary:")
```

```
print(student)
```

Sample Input

Enter new key: college

Enter value: ABC Engineering College

Output

Updated Dictionary:

```
{'name': 'Ravi', 'age': 20, 'college': 'ABC Engineering College'}
```

Important Concept

The syntax for adding a new key-value pair is:

```
dictionary[key] = value
```

For example:

```
student["city"] = "Vijayawada"
```

5. Sum All the Items in a Given Dictionary

Suppose we have a dictionary containing subjects and marks:

```
marks = {  
    "Python": 80,  
    "Java": 75,  
    "DBMS": 85,  
    "AI": 90  
}
```

We want:

```
80 + 75 + 85 + 90 = 330
```

Method 1: Using sum() — Simple Method

Create dictionary

```
marks = {  
    "Python": 80,  
    "Java": 75,  
    "DBMS": 85,  
    "AI": 90  
}
```

```
# Calculate sum of values  
total = sum(marks.values())
```

```
print("Sum of all items =", total)
```

Output

Sum of all items = 330

Explanation

marks.values() gives all the values:

marks.values()

Conceptually:

80, 75, 85, 90

Then:

sum(marks.values())

calculates:

80 + 75 + 85 + 90

Therefore:

330

Method 2: Using a for Loop

This version is particularly useful for understanding the logic.

```
marks = {  
    "Python": 80,  
    "Java": 75,  
    "DBMS": 85,  
    "AI": 90  
}
```

```
total = 0
```

```
for value in marks.values():
```

```
    total = total + value
```

```
print("Sum of all items =", total)
```

Dry Run

Initially:

```
total = 0
```

Value Calculation Total

80	0 + 80	80
----	--------	----

75	80 + 75	155
----	---------	-----

85	155 + 85	240
----	----------	-----

90	240 + 90	330
----	----------	-----

Final result:

Sum of all items = 330