UNIT III CONCEPTS

Python Dictionaries

What is a Dictionary?

A **dictionary** is a collection of **key-value pairs**. It is:

- **Ordered** (Python 3.7+)
- **Mutable** (can be modified)
- Keys are **unique**
- Values can be duplicated
- Created using **curly braces {}**

Syntax

```
dictionary_name = {
   key1: value1,
   key2: value2
}
```

---

1. Creating Dictionary

Example 1

```
student = {
   "id": 101,
   "name": "Ram",
   "branch": "CSE"
}


print(student)
```

Output

```
{'id': 101, 'name': 'Ram', 'branch': 'CSE'}
```

Example 2

```
employee = {
   "empid": 1001,
   "name": "John",
   "salary": 50000
}

print(employee)
```

Empty Dictionary

```
d = {}
print(d)
```

Output

```
{}
```

---

2. Accessing and Modifying Key-Value Pairs

Access Using Key

```
student = {
   "name": "Ram",
   "age": 20,
   "course": "Python"
}

print(student["name"])
print(student["age"])
```

Output

```
Ram
20
```

Using get()

print(student.get("course"))

Output

Python

Modify Value

student["age"] = 21

print(student)

Output

{'name': 'Ram', 'age': 21, 'course': 'Python'}

Add New Key

student["city"] = "Hyderabad"

print(student)

Output

{'name': 'Ram', 'age': 21, 'course': 'Python', 'city': 'Hyderabad'}

---

3. Built-in Functions Used on Dictionaries

Function	Description	Example
len()	Number of key-value pairs	len(d)
type()	Returns data type	type(d)
str()	Converts to string	str(d)
dict()	Creates a dictionary	dict()
sorted()	Returns sorted list of keys	sorted(d)

Example

```
student = {
   "id":101,
   "name":"Ram",
   "course":"Python"
}

print("Length =", len(student))
print("Type =", type(student))
print("Keys =", sorted(student))
```

Output

```
Length = 3
Type = <class 'dict'>
Keys = ['course', 'id', 'name']
```

---

4. Dictionary Methods

keys()

```
student = {
   "id":101,
   "name":"Ram"
}

print(student.keys())
```

Output

```
dict_keys(['id', 'name'])
```

---

values()

print(student.values())

Output

dict_values([101, 'Ram'])

---

items()

print(student.items())

Output

dict_items([('id',101),('name','Ram')])

---

update()

student.update({"age":20})

print(student)

Output

{'id':101,'name':'Ram','age':20}

---

pop()

student.pop("age")

print(student)

Output

{'id':101,'name':'Ram'}

---

popitem()

Removes the last inserted item.

student.popitem()

print(student)

---

clear()

```
student.clear()


print(student)
```

Output

```
{}
```

---

copy()

```
d1 = {
   "A":10,
   "B":20
}


d2 = d1.copy()


print(d2)
```

---

5. del Statement

Delete One Item

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


del student["age"]


print(student)
```

Output

{'name':'Ram'}

Delete Entire Dictionary

```
student = {
   "name":"Ram"
}

del student
```

Dictionary Traversal

```
student = {
   "id":101,
   "name":"Ram",
   "course":"Python"
}

for key, value in student.items():
   print(key, ":", value)
```

Output

id : 101

name : Ram

course : Python

Summary Table (Dictionary)

Method	Purpose
keys()	Returns keys

Method	Purpose
values()	Returns values
items()	Returns key-value pairs
get()	Returns value for key
update()	Updates dictionary
pop()	Removes specified key
popitem()	Removes last item
clear()	Removes all items
copy()	Copies dictionary

Python Tuples

What is a Tuple?

A **tuple** is an **ordered, immutable** collection of elements.

- Ordered
- Immutable (cannot be modified)
- Allows duplicates
- Written using parentheses ()

---

1. Creating Tuples

numbers = (10,20,30)

print(numbers)

Output

(10,20,30)

Single Element Tuple

t = (10,)

print(type(t))

Mixed Tuple

t = (10,"Python",3.14)

print(t)

---

2. Basic Tuple Operations

Concatenation

```
t1=(1,2)
t2=(3,4)

print(t1+t2)
```

Output

(1,2,3,4)

Repetition

```
print((1,2)*3)
```

Output

(1,2,1,2,1,2)

Membership

```
t=(10,20,30)

print(20 in t)
```

Output

True

---

3. tuple() Function

```
list1=[10,20,30]

t=tuple(list1)

print(t)
```

Output

(10,20,30)

---

4. Indexing and Slicing

```
t=(10,20,30,40,50)


print(t[2])
print(t[-1])


print(t[1:4])


print(t[::-1])
```

Output

```
30
50
(20,30,40)
(50,40,30,20,10)
```

---

5. Built-in Functions on Tuples

```
t=(10,20,30,40)


print(len(t))
print(max(t))
print(min(t))
print(sum(t))
```

Output

```
4
40
10
100
```

---

6. Tuple Methods

```
t=(10,20,20,30)


print(t.count(20))


print(t.index(30))
```

Output

2

3

Difference between Tuple, List and Dictionary

Feature	**Tuple**	**List**	**Dictionary**
Syntax	(10, 20, 30)	[10, 20, 30]	{"a": 10, "b": 20}
Ordered	Yes	Yes	Yes*
Changeable	**No**	**Yes**	**Yes**
Access	By index	By index	By key
Duplicate values	Allowed	Allowed	Keys cannot be duplicate
Example	(1, 2, 3)	[1, 2, 3]	{"x": 1, "y": 2}

*Dictionaries maintain insertion order in modern Python.

Relation Between Tuples and Lists

List	Tuple
Mutable	Immutable
[]	()
append(), remove() available	Not available
Faster	Faster for read-only data
More memory	Less memory

Convert List to Tuple

```
l=[1,2,3]

t=tuple(l)

print(t)
```

Convert Tuple to List

```
t=(10,20,30)

l=list(t)

print(l)
```

Relation Between Tuples and Dictionaries

Dictionary items can be converted into tuples.

```
student={
   "id":101,
   "name":"Ram"
}

print(tuple(student.items()))
```

Output

```
(('id',101),('name','Ram'))
```

Using zip() Function

zip() combines two or more iterables element by element.

```
names=["Ram","Laxman","Sita"]

marks=[90,85,95]

result=zip(names,marks)

print(list(result))
```

Output

```
[('Ram',90),('Laxman',85),('Sita',95)]
```

Python Sets

What is a Set?

A **set** is an **unordered collection of unique elements**.

- Unordered
- Mutable
- No duplicate values
- Created using {} or set()

Creating Sets

s={10,20,30}

print(s)

Duplicate Values

s={10,20,20,30}

print(s)

Output

{10,20,30}

Set Methods

add()

s={10,20}

s.add(30)

print(s)

---

update()

s.update([40,50])

print(s)

---

remove()

s.remove(20)

print(s)

---

discard()

s.discard(100)

No error even if the element is absent.

---

pop()

s.pop()

print(s)

Removes a random element because sets are unordered.

---

clear()

s.clear()

print(s)

Set Operations

Union

A={1,2,3}

B={3,4,5}

print(A|B)

Output

{1,2,3,4,5}

Intersection

print(A&B)

Output

{3}

Difference

print(A-B)

Output

{1,2}

Symmetric Difference

print(A^B)

Output

{1,2,4,5}

Frozen Set

A **frozenset** is an immutable version of a set.

Once created, its elements **cannot be added or removed**.

fs = frozenset([10,20,30])

print(fs)

Output

frozenset({10,20,30})

Attempting to modify it results in an error:

fs.add(40) # AttributeError

Compare Tuple, List, Dictionary and Set in Python

Feature	**Tuple**	**List**	**Dictionary**	**Set**
Syntax	(1, 2, 3)	[1, 2, 3]	{"a": 1, "b": 2}	{1, 2, 3}
Ordered	Yes	Yes	Yes	No guaranteed order
Mutable	No	Yes	Yes	Yes
Duplicates	Allowed	Allowed	Keys: No	Not allowed
Access	By index	By index	By key	No indexing
Stores	Values	Values	Key-value pairs	Unique values
Example	(10, 20, 30)	[10, 20, 30]	{"x":10, "y":20}	{10, 20, 30}

Summary Table

Data Structure	Mutable	Ordered	Duplicates
List	✅ Yes	✅ Yes	✅ Yes
Tuple	❌ No	✅ Yes	✅ Yes
Dictionary	✅ Yes	✅ Yes (Python 3.7+)	❌ Duplicate keys not allowed
Set	✅ Yes	❌ No	❌ No
Frozen Set	❌ No	❌ No	❌ No

VIVA VOCE and Exam Tips

1. **Dictionary**
    - Stores data as **key-value pairs**.
    - Keys must be **unique and immutable** (e.g., strings, numbers, tuples).
2. **Tuple**
    - Tuples are **immutable**, making them suitable for fixed collections of data.
    - Only two built-in methods are available: count() and index().
3. **Set**
    - Sets automatically remove duplicate elements.
    - Because sets are unordered, you **cannot access elements by index**.
4. **zip() Function**
    - Combines corresponding elements from multiple iterables into tuples.
    - Commonly used to create dictionaries:
    - keys = ["id", "name"]
    - values = [101, "Ram"]
    -
    - d = dict(zip(keys, values))

print(d)

Output

{'id': 101, 'name': 'Ram'}

5. **frozenset**

- Immutable version of a set.
- Useful when you need a set whose contents must not change.