Lecture 7
Dictionaries

Outlines

- Introduction to dictionaries (key-value pairs)
- Dictionary methods: `keys()`, `values()`, `items()`
- Accessing, adding, and removing dictionary elements
- Iterating through dictionaries
- Dictionary membership

INTRODUCTION TO DICTIONARIES

- A dictionary is a mixed unordered collection of elements.
- While other compound data types have only value as an element, a dictionary has a key: value pair.
- The keys in a Python dictionary is separated by a colon (:) while the commas work as a separator for the elements.
- The key value pairs are enclosed with curly braces { }.
- Dictionaries are optimized to retrieve values when the key is known.
- Key in the dictionary must be unique case sensitive and can be of any valid Python type.

Difference between List and Dictionary

- List is an ordered set of elements. But, a dictionary is a data structure that is used for matching one element (Key) with another (Value).
- The index values can be used to access a particular element. But, in dictionary key represents index. Remember that, key may be a number or a string.
- Lists are used to look up a value whereas a dictionary is used to take one value and look up another value.

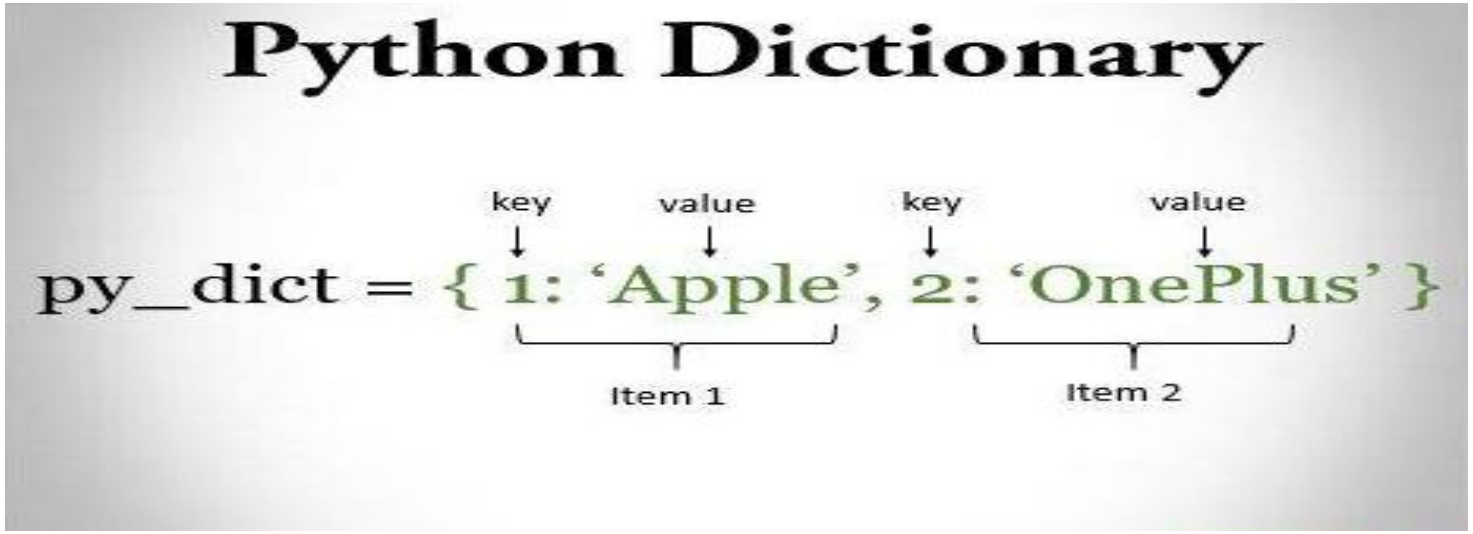
List vs Dictionary

List	Dictionary
Select, update and remove: [ ]	Select, update and remove: [ ]
Indexed by range of numbers	Indexed by unique keys
Collection of values order matters select entire subsets	Lookup table with unique keys

Creating a dictionary

- Creating a dictionary is as simple as placing items inside curly braces {} separated by comma.
- Each element in a dictionary is represented by a **key:value** pair.
- While values can be of any data type and can repeat, keys must be of immutable type and must be unique.
- Syntax:

```
Dictionary_Name = { Key_1: Value_1,  Key_2:Value_2,...., Key_n:Value_n }
```



Example

```
dictionary={1:"apple",2:"banana",3:"kiwi",4:"watermelon",5:"pinapple"}
print(dictionary)
```

Output

```
{1: 'apple', 2: 'banana', 3: 'kiwi', 4: 'watermelon', 5: 'pinapple'}
```

Creating a Dictionary

```
#empty dictionary
emp_dict={}

#dictionary with key:elements pair
dictionary={"k1":"apple","k2":"banana","k3":"kiwi","k4":"watermelon","k5":"pinapple"}
```

Dictionary Comprehensions

syntax

```
Dict = {expression for variable in
sequence [ if condition]}
```

The if condition is optional and if specified, only those values in the sequence are evaluated using the expression which satisfy the condition.

Example

```
Dict={x:2*x for x in range(1,10)}
print(Dict)
```

output

```
{1: 2, 2: 4, 3: 6, 4: 8, 5: 10, 6: 12, 7: 14, 8: 16, 9: 18}
```

Dictionary methods .keys() .values() .items()

- We can use .keys() method to return all the keys in dictionary
- We can use .values() method to return all the values in dictionary
- We can use .items() method to return all the (key,value) pairs in dictionary

```
Marks={'scince':98, 'Math':87, 'Arabic':100}
K=Marks.keys()
print(K)
V=Marks.values()
print(V)
E=Marks.items()
print(E)
```

```
dict_keys(['scince', 'Math', 'Arabic'])
dict_values([98, 87, 100])
dict_items([('scince', 98), ('Math', 87), ('Arabic', 100)])
```

Accessing Dictionary elements

- Accessing all elements from a dictionary is very similar as Lists and Tuples.
- Simple print function is used to access all the elements.
- If you want to access a particular element, square brackets can be used along with key.
- **Key** can be used either inside **square brackets** or with the **get() method**.
- The difference while using **get()** is that it returns **None** instead of KeyError, if the key is not found.

Accessing Dictionary elements

- Example of using [] and get() method

```
Marks={'scince':98, 'Math':87, 'Arabi
print(Marks['scince'])
```

```
98
```

```
print(Marks['physics'])
```

```
----------------------------------
KeyError
```

Getting error when using [] to access non-existing element

```
print(Marks.get('scince'))
```

```
98
```

```
print(Marks.get('physics'))
```

```
None
```

Getting None when using .get to access non-existing element

Example: Program to access all the values stored in a dictionary

```
MyDict = { 'Reg_No': '1221',
           'Name': 'Tamilselvi',
           'School' : 'CGHSS',
           'Address' : 'Rotler St., Chennai 112' }

print(MyDict)
print("Register Number: ", MyDict['Reg_No'])
print("Name of the Student: ", MyDict['Name'])
print("School: ", MyDict['School'])
print("Address: ", MyDict['Address'])
```

output

```
{'Reg_No': '1221', 'Name': 'Tamilselvi', 'School': 'CGHSS', 'Address': 'Rotler St., Chennai 112'}
Register Number:  1221
Name of the Student:  Tamilselvi
School:  CGHSS
Address:  Rotler St., Chennai 112
```

Add Dictionary elements

- Dictionary are mutable. We can add new items or change the value of existing items using assignment operator.
- If the key is already present, value gets updated, else a new key: value pair is added to the dictionary.

Example: Program to add a new value in the dictionary

```
MyDict = { 'Reg_No': '1221',
           'Name' : 'Tamilselvi',
           'School' : 'CGHSS',
           'Address' : 'Rotler St., Chennai 112'}

print(MyDict)

MyDict['Class'] = 'XII - A'      # Adding new value
print("Class: ", MyDict['Class'])   # Printing newly added value

print(MyDict)
```

Output

```
{'Reg_No': '1221', 'Name': 'Tamilselvi', 'School': 'CGHSS', 'Address': 'Rotler St., Chennai 112'}
Class:  XII - A
{'Reg_No': '1221', 'Name': 'Tamilselvi', 'School': 'CGHSS', 'Address': 'Rotler St., Chennai 112', 'Class': 'XII - A'}
```

Deleting dictionary elements

- We can remove a particular item in a dictionary by using the method pop(). This method removes as item with the provided key and returns the value.
- All the items can be removed at once using the clear() method.
- We can also use the del keyword to remove individual items or the entire dictionary itself.

Deleting elements from a dictionary

```
StudentDetails={'name': 'Ali', 'Course':'Data Base', 'ID': 432456, 'Mark': 90}
print(StudentDetails)
StudentDetails.pop('ID')
print(StudentDetails)
```

```
{'name': 'Ali', 'Course': 'Data Base', 'ID': 432456, 'Mark': 90}
{'name': 'Ali', 'Course': 'Data Base', 'Mark': 90}
```

```
StudentDetails.clear()
print(StudentDetails)
```

```
{}
```

Deleting elements from a dictionary

```
squares={1:1, 2:4, 3:9, 4:16, 5:25}
print(squares)
num=int(input("Enter a key to delet: "))
del squares[num]
print(squares)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
Enter a key to delet: 2
{1: 1, 3: 9, 4: 16, 5: 25}
```

To delete random element

- The method, popitem() can be used to remove and return an arbitrary item (key, value) form the dictionary.
- Raises KeyError if the dictionary is empty.

```
squares={1:1, 2:4, 3:9, 4:16, 5:25}
print(squares)
squares.popitem()
print(squares)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
{1: 1, 2: 4, 3: 9, 4: 16}
```

Clear method and del statement

- In Python dictionary, **del** keyword is used to delete a particular element. The **clear()** function is used to delete all the elements in a dictionary. To remove the dictionary, you can use **del** keyword with dictionary name.

```
# To delete a particular element
del dictionary_name[key]

# To delete all the elements
dictionary_name.clear()

# To delete an entire dictionary
del dictionary_name
```

Clear method and del statement

Example

```
Dict = {'Roll No': 12001, 'SName': 'Meena', 'Mark1': 98, 'Marl2': 86}
print("Dictionary elements before deletion: \n", Dict)

del Dict['Mark1']            # Deleting a particular element
print("Dictionary elements after deletion of a element: \n", Dict)

Dict.clear()                 # Deleting all elements
print("Dictionary after deletion of all elements: \n", Dict)

del Dict                     # Deleting entire dictionary
print(Dict)
```

Output

```
Dictionary elements before deletion:
 {'Roll No': 12001, 'SName': 'Meena', 'Mark1': 98, 'Marl2': 86}
Dictionary elements after deletion of a element:
 {'Roll No': 12001, 'SName': 'Meena', 'Marl2': 86}
Dictionary after deletion of all elements:
 {}
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-4057224931.py in <cell line: 0>()
      9
     10 del Dict                     # Deleting entire dictionary
---> 11 print(Dict)

NameError: name 'Dict' is not defined
```

Iterating Through a Dictionary

- Using a for loop we can iterate through each key in a dictionary.

```
Marks={'scince':98, 'Math':87, 'Arabic':100}
for subject in Marks:
    print(subject)
```

```
scince
Math
Arabic
```

```
Marks={'scince':98, 'Math':87, 'Arabic':100}
for subject in Marks:
    print(subject, ":", Marks[subject])
```

```
scince : 98
Math : 87
Arabic : 100
```

```python
Marks={'scince':98, 'Math':87, 'Arabic':100}
for subject,score in Marks.items():
    print(subject, ":", score)
```

```
scince : 98
Math : 87
Arabic : 100
```

Dictionary Membership Test

- We can test if a key is in a dictionary or not using the keyword in. Notice that membership test is for keys only, not for values.

```
squares={1:1, 2:4, 3:9, 4:16, 5:25, 6:36, 7:49}
print(squares)
print("is 6 in squares?", 6 in squares)
print("is 8 in squares?", 8 in squares)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49}
is 6 in squares? True
is 8 in squares? False
```