AS22-05ES10 / FUNDAMENTALS OF PYTHON PROGRAMMING.

DEPARTMENT OF CSE(AI&ML)/CSM.

UNIT-I (1 Mark Question and Answer).

① Recall the history of Python?

* Python is a Widely used general purpose high level Programming Language.
* Python was developed by Guido van Rossum in 1991.

② Describe the use of Print() function?

Print() - is a method which is used to display the message in the output Screen.

For ex, Print("Hello Word") → Hello World is message display in o/p.

③ Identify the key features of Python Programming lang.

* Python is Interactive
* Python is Interpreted
* Python is Object Oriented
* Python is open source
* Python is Beginner's Language.

④ Out line the modes available for executing Python Programs.

* There are two modes/methods used for running the Python code.
  1. Script mode → Write the Code and Save finally execute.
  2. Interactive mode → is a Command Line which gives immediate results for each Statement

⑤ List the Standard datatypes in Python.

* Python has 5 Standard datatypes.
  1. Numbers
  2. String
  3. List
  4. Tuple
  5. Dictionary.

UNIT-I (3 MARKS)

Define a list and differentiate a list from tuple.

① ~~What is a list? How lists differ from tuples?~~

* List is a collection of ordered elements and the element could be any type.
* It's a mutable object; that means we can modify the list element after creation of list.
* Tuple also collection of ordered elements like list.
* But, tuple is immutable; that means we can't modify the content after creation of tuple.

② ~~How to~~ Slicing a list in Python?

Demonstrate

* The slice operation is used to access the set of specific data item from the list.
* The Colon (:) operator is used for slice operation.

SYNTAX: list_name [start : end] (or) list_name [start : stop : step]

Where, list_name → Name of the list
start → starting position of list
stop - Ending position of list
step - No. of step to be inc.

③ ~~What operators does Python support?~~

Identify the operators supported in python

* Python supports they are seven operators including arithmetic & assignment

1. Arithmetic operator
2. Relational operator
3. Assignment operator
4. Logical operator
5. Bitwise operator
6. Membership operator
7. Identity operator.

④ Explain ~~Explain~~ type Conversion in Python with an example.

type Conversion → The process of Converting data from one type to another.

* There are two types of Conversion.

① Implicit Conversion → This type is performed by automatically by Python when you perform operation b/w different datatypes

For ex, a=10 b=13.5 c=a+b print(c)

② Explicit Conversion → This type of Conversion is done manually by the Programmers. For ex, str="100"

n=int(str)

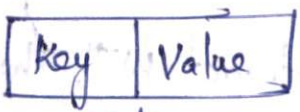
(5) Differentiate between Script mode and Interactive mode in Python

SCRIPT MODE	INTERACTIVE MODE
* Python interpreter to read and execute	* typing Commands in Command prompt
* It Can save and edit	* Cant Save and edit
* Cant see the results immediately.	* Can see the results immediately.

UNIT-I (5 Marks)

(1) Define a dictionary in python and demonstrate its operations.
~~What is meant by dictionary with an operations?~~

* The dictionary is unordered collection of items that can be in the form of key-value pair.
* It Contains Collection of indices called keys and Collection of values.
* Each key is associated with a single value.



Dictionary

Creating a Dictionary:

* Dictionary can be created by placing items within the Curly braces { } seperated by Commas.
* An item has a key and Corresponding value { key : value }
* The value may be any datatype and can be repeated, key must be immutable type and must be unique.

For ex,

```
my_dict = { }                          → Creating a Empty dictionary
my_dict = {1: "ONE", 2: "TWO"}         → Dictionary with integer key
my_dict = dict({1:"ONE"})              → Create by using dict().
```

Dictionary Methods :

1. Copy() → The method is used to return a duplicate of the dictionary.
2. get() → The method is used to return the value of specified key.
3. items() → The method is used to return a object of the dictionary item in (key, value) format.
4. keys() → The method is used to return dictionary items key
5. values() → The method is used to returns dictionary items value only.

(2) ~~Write the features of Python?~~
Identify the key features in Python and explain their significance.

Python is Interpreted : Python is processed at runtime by the interpreter;
No need to Compile the Python Program.

Python is Interactive : Interact with the interpreter directly to write a program

Python is object oriented : Python supports object oriented Concepts such as class, object, data abstraction, encapsulation

Python is Beginner's lang : It's great language for Beginners, it supports wide range of application from text processing to game development

Python is open source : It's open source, this means that source code is also available to the Public

Python is Easy to use : It's a high level programming language. Easy to learn when Compare to other language.

Python is platform Independence : It's a portal Programming language

(3) ~~What are the applications of Python?~~ List and describe various applications of Python in real-world Scenarios.

* Python is a easy to use and Versatile programming language.
* It allows to Create applications in many areas.

Web Development: Django, Flask and pyramid are popular Web frameworks in Python.
* It's used for building applications range from small to large enterprise application

Data Science & Analytics: * Python is a highly demanded language in data Science and Machine learning with popular libraries like Numpy, Pandas, Matplotlib.

Scientific Computing: * Python is widely used in Scientific Computing, with modules for numerical Calculations, optimization and Special functions.

Desktop GUI Application: * It provides several GUI toolkit like Tkinter, PyQT and wxPython.
* The toolkit allows to develop several GUI applications.

Game Development: * Python can be used to develop games with libraries like Pygame.
* It provides graphics, sounds and input tool for game development.

(4) Explain the datatypes in Python.

* There are different types of datatypes in Python.

1. Numeric datatype → int, float, Complex.
2. String datatype → str
3. Sequence datatype → list, tuple, range
4. Mapping datatype → dict
5. Set datatype → set, frozenset

(4) Differentiate between List and Tuple in Python with an example.

LIST	TUPLE
* List is a Collection of order elements and the elements could be any datatype.	* Tuple is ordered Collection of data elements, that could be any datatype.
* List is Mutable object	* Tuple is Immutable object.
* Mutable means – We can modify the list element even after the Creation of list.	* Immutable means – We cant change (or) modify the tuple element even after the Creation of tuple.
* List can be Created with in the [].	* tuple can be Created with in the ().
* For ex: list1 = [10, 20, 30, "Hi", 40]	For ex: tuple1 = (10, 20, 30, 40, 50)

Element	10	20	30	"Hi"	40
Index	0	1	2	3	4

Element	10	20	30	40	50
Index	0	1	2	3	4

(5) Explain in detail about Python type Conversion and type casting?

Type Conversion → The process of Converting one datatype to another datatype.

* There are two types of type Conversion in Python.

1. Implicit type Conversion 2. Explicit type Conversion

Implicit type: * In this type, Python automatically Converts one datatype to another datatype.

For ex,

```
a=100
b=3.14
c=a+b.
Print("type of A is:", a)
Print("The type of B is:", b)
Print("The type of C is:", c).
```

Explicit type: * In this type, user explicitly Converts one datatype to another
* The explicit is done by using int(), float() and str() method

For ex,

```
a="100"
b=3.14
Print("The type of a is:"a)
a=int(a)
Print("The type of a is:", a)
```

op: The type of a is: str
The type of a is: int

① Illustrate the basic list operation that can be performed in Python. Explain each operation with its syntax and an example.

① What are the basic list operations that can be performed in Python? Explain each operation with its syntax and example. (8)

LIST: ⟶ List as Array

* List is a collection of ordered elements and the elements could be any type.
* Lists are mutable; Mutable means ⟶ We can modify the list element even after the Creation of list.
* The element indexing is start with Zero.

CREATING A LIST:

* The list can be Created with in the square bracket "[]".
* The sequence of element seperated by Comma (,) operator.

SYNTAX:

```
List_name = [ "Sequence of element"]
```

EXAMPLE:

list1 = [1, 2, 3, 4, 5] ⟶ { Where, list1 → list name; 1,2,3,4,5 → data elements in the list1.

list2 = ["apple", "orange", "banana"]

list3 = [1, "apple", 2]

Element	apple	orange	banana
Index	0	1	2

ACCESSING THE LIST VALUE:

* The list elements are accessed by using following techniques.
  - Indexing
  - Nested Indexing
  - Reverse Indexing

1. INDEXING:

* The element in the list is accessed by the index operator [] and refering to the index no.
* The index is always start with zero.

For ex,

```
                 0        1        2
list2 = [ "apple", "orange", "banana" ]
```

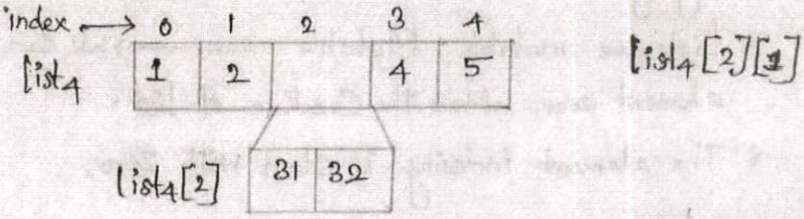
list2 [1] ⟶ it returns orange data item from the given list2.

Nested Indexing:

* The Nested list are indexed / accessed by using Nested indexing.

For ex,

$List_4 = [1, 2, [31, 32], 4, 5]$



list4[2][1]

Reverse Indexing:

* The Reverse indexing is indexed from the end of the list.
* The index starting range is -1, -2 and etc.,

SYNTAX:

List_name [-index]

For ex, list2 = ["apple", "orange", "banana"]

list2 [-1] ⟶ it returns banana data elements.

Element	apple	orange	banana
Index	0	1	2
Reverse	-3	-2	-1

LIST OPERATIONS:

SLICE THE LIST:

* The Slice operation is used to access the set of specific data items from the list.
* The Color operator (:) is used for Slice operation.

SYNTAX:

List_name [Start : end] (or) List_name [Start : End : Step].

Where, List_name ⟶ Name of the list
Start ⟶ Starting position of the list
End ⟶ Ending position of the list.
Step ⟶ No. of steps to be increment.

For ex, 0 1 2 3 4

list2 = ["apple", "orange", "banana", "mango", "kiwi"]

list2[2:4] → The search start at index 2 (included) and end at index 4 (not include) (ie) upto indexing position 3.

Print (list2[2:4]) →

Output:

["banana", "mango"].

list2[:4] → returns Starting elements to Third position.

list2[2:] → returns from 2nd index pos to end of the element

list2[:] → return entire data items from the list2.

CONCATENATING A LIST:

* The '+' operator is used to Combine / Concatenate two (or) more list.

SYNTAX:

list1 + list2

For ex,

ls1 = [1,2,3,4]
ls2 = [5,6,7,8]
ls3 = ls1 + ls2
Print (ls3)

OUTPUT:
[1,2,3,4,5,6,7,8].

Repeating List:

* The '*' operator is used to multiply the given list into n times.

SYNTAX:

list * No. of Repetition

For ex,

list1 = [1,2,3,4] *2 // Repeat the list two times.

O/P: [1,2,3,4,1,2,3,4]

Mutability:

* Here, List is a mutable objects, That means we can change the any elements from the given list.
* But String is immutable objects.

SYNTAX:

list[index] = Value.

For ex,

```
l1 = [10, 20, 30, 40, 50]
l1[2] = 100      # The value of 100 is assigned to the 2nd position
                   of the list.
Print(l1)
```

O/P: [10, 20, 100, 40, 50]

ALIASING ⟶ The object has more than one name; it is called as Aliasing

For ex,

```
l = [1, 2, 3, 4]
l1 = l
Print(l1)
```

OUTPUT: [1, 2, 3, 4]

NOTE: Here if you made any changes in the reference list; it will affect in the original list also.

For ex,

```
l = [10, 20, 30, 40, 50]
l1 = l
Print(l1)
l1[2] = 100
Print(l1)
Print(l)
```

OUTPUT:

[10, 20, 30, 40, 50]

[10, 20, 100, 40, 50]

[10, 20, 100, 40, 50]

Copy / Clone by COPY CONSTRUCTOR:

* The list assignment methods is used to Copy the List into another List (target) (source)
* If we perform any changes in the target list, it doesn't affect the source list.

For ex,

```
l = [10, 20, 30, 40, 50]
l1 = list(l)
Print(l1)
```

```
l1[2] = 100
Print(l1)
Print(l)
```

LIST METHODS:

* They are so many methods related to List operations availabl in Python.

* append(), extend(), insert(), Pop(), remove(), erase(), Sort(), Count(), reverse() - are some of List methods.

1. append() ⟶ It used to add the element at the last position of the list.

SYNTAX:

```
list.append(element)
```

For ex,

```
a=[10,20,30,40]
a.append(50)   #adding at the End of list
Print(a).
```

OUTPUT:

```
[10,20,30,40,50]
```

2. extend() ⟶ Used to add the some List at the end of given list.

SYNTAX:

```
list1.extend(list2)
```

Where, list1 — is a old list

list2 — is a new list; it going to add end of list1.

For ex,

```
l1=[10,20,30,40]
l2=["A","B","C"]
l1.extend(l2)
```

OUTPUT:

```
[10,20,30,40,"A","B","C"].
```

3. insert() ⟶ The method is used to insert the element at specified location

SYNTAX:

```
list.insert(Position, element)
```

(Position → indexing position; element → New element.)

For ex,

```
list1=[1,2,3,4,5]
list1.insert(2,100)
Print(list1)
```

OUTPUT:

```
[1,2,100,3,4,5]
```

4. POP() ⟶ The method is used to remove the specified element by the index position.

SYNTAX:

```
list1.Pop(index).
```

list1.pop() ⟶ return / remove the last element from the list.

Where, index ⟶ parameter represents remove the specified indexed pos.

index ⟶ NULL Means remove the last element from the list.

For ex,

```
list1 = [10,20,30,40,50]
list1.pop(2)    # Remove the 2nd index position.
Print (list1)
```

OUTPUT :

[10, 20, 40, 50]

For ex,

```
l1 = [1,2,3,4,5]
ret = l1.pop()
Print("The del element is:", ret)
Print("Updated list :", l1)
```

OUTPUT :

The del element is : 5

Updated list : [1,2,3,4].

5. remove() ⟶ The remove() method is used to remove / delete the element by specified value.

SYNTAX :

```
list.remove (value)
```

For ex,

```
l1 = [10,20,30,20,40,50].
l1.remove(20)
Print(l1)
```

OUTPUT :

[10, 30, 20, 40, 50].

* if more than one value is present means, the First specified element is removed from the list.

6. del() ⟶ The del() method is used to delete / remove Some Specified range of element from the list.

SYNTAX :

```
del list_name [ Start : Stop]
```

For ex,

```
l1 = [1,2,3,4,5,6]
del l1[2:5]
Print(l1)
```

OUTPUT :

[1,2,6].

7. Sort() ⟶ The method is used to Sort/arrange the elements in Ascending order (or) increasing order.

SYNTAX:

```
list.sort()
```

For ex,

```
l1 = [50,40,30,20,10]
l1.sort()
print(l1)
```

OUTPUT:

[10,20,30,40,50].

For ex,

```
l2 = ['x','a','c','w','z']
l2.sort()
print(l2)
```

OUTPUT:

['a','c','w','x','z'].

8. reverse() ⟶ The method is used to reverse the elements in the list.

SYNTAX:

```
list.reverse().
```

⟶ It describe decreasing order

For ex,

```
l1 = [10,20,30,40,50]
l1.reverse()
print(l1)
```

OUTPUT:

[50,40,30,20,10]

9. Count() ⟶ The method used to returns how many times the particular element is appeared in the list.

SYNTAX:

```
list.count(element)
```

For ex,

```
l1 = [10,20,10,30,10,40,50]
l1.count(10)
print(l1)
```

OUTPUT:

3

* The element 10 is 3 times presents in the given list.

10. max() ⟶ The method is used to returns the max element from the given list.

For ex,

```
l1 = [10,20,50,30,40]
print(max(l1))
```

o/p: 50.

11. min() ⟶ The method is used to return the minimum element from the given list.

for ex,

```
l1 = [10, 50, 20, 40, 30]
Print (min(l1))
```

OUTPUT:
10.

12. sum() ⟶ The method is used to add the all element from the given list

for ex,

```
l1 = [10, 20, 30, 40, 50]
Print (sum(l1))
```

OUTPUT:
150.

ADDITIONAL PROGRAMS:

1. Write a Python Program to finding max & min value from the list
2. Write a Python program to perform linear search operation.

1. Min & Max Value

```
n= int(input("Enter the val of N:"))
a=[]
for i in range(0,n):
    a.append(int(input()))
    mini= a[0], maxi=a[0]
for i in range(0,n):
    if a[i]<mini:
        mini=a[i]
Print("The minimum value is", mini)
:
```

2. Linear Search

```
n = int(input("Enter the Val of N:"))
a=[]
found=0
for i in range(0,n):
    a.append(int(input()))
Print("Enter the key value")
key =int(input())
for i in range(0,n):
    if a[i]==key:
        found=1
        break.
if found==0:
    Print("The key is not present")
else:
    Print("The key is present").
```

Explain in detail about tuples in Python with different method and example.

(3) TUPLE: → Like list, tuple is the Ordered Collection of data elements, that could be any datatype.

* tuples are immutable, that means we cant change (or) update after creating the list.
* Here also indexing start with Zero.
* The data element maybe enclose with in the paranthesis.

CREATING A TUPLE:

* The sequence of element always enclosed with paranthesis.( ).

SYNTAX:

tuple_name = ("Sequence of Element")

For ex,

tuple1 = (1, 2, 3, 4, 5)
(tuple1 → tuple name; 1, 2, 3, 4, 5 → Sequence of Element.)

tuple2 = ("apple", "orange", "banana").

ACCESSING ELEMENTS OF THE TUPLES:

* The element in the tuple is accessed using the index operator []
* The indexing is start with Zero.
* They are 3 ways indexing.
  1. Normal Indexing
  2. Nested Indexing.
  3. Reverse Indexing

IMMUTABILITY:

* tuples are immutable, We cant change (or) update the tuple elements.
*

For ex,

tuple1 = ("apple", "orange", "banana")

tuple1[0] = "lemon".

O/P:

TYPE ERROR: 'tuple' object does not support item assignment.

Concatenating Tuple:

* The '+' operator is used to Combine two (or) more tuples.

Syntax:

tuple1 + tuple2 ⟶

For ex,

```
t1=(10,20,30,40)
t2=("A","B","c")
t1+t2
```

Output:

(10, 20, 30, 40, "A", "B", "c").

Repeating tuple:

* The '*' operator is used to repeat the tuple at n times.

Syntax:

tuple1 * n

For ex,

```
tuple1 = (10,20,30)
tuple1 * 2
```

OUTPUT:

(10, 20, 30, 10, 20, 30)

Tuple Assignment:

* Python has powerful tuple assignment feature that allows a tuple of variable on the left side.

For ex,

```
m1, m2, m3 = 90, 95, 98
```

* The above statement is equal to three assignment statements.

Tuple Functions:

* There are many functions available related to tuple.

1. len() ⟶ The function returns the no. of elements in the tuples.
   * it used to find the length of the tuple.

Syntax: len (tuplename).

For ex,

```
t1 = (10, 20, 30, 40, 50, 60)
len(t1)
```

o/p: 6.

2. max() ⟶ The function returns the maximum element in the tuple.

For ex, t1 = (10,20,30,50,40,60)
max(t1)

o/p : 60

3. min() ⟶ The function returns the minimum element in the tuple

For ex, t1 = (10,20,30,50,60,40)
min(t1)

o/p : 10

4. Sum() ⟶ The function returns the Sum of element in the tuple.

For ex, t1 = (10,20,30,50,60,40)
Sum(t1)

o/p : 210.

5. Sorted() ⟶ The function returns the Sorted tuple.
*By default, the tuple is sorted in ascending order

SYNTAX:
Sorted(tuplename) #Sort the tuple in ascending order
Sorted(tuplename, reverse = True) # Sort in descending order.

For ex,

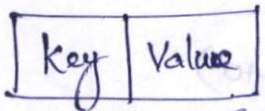
t1 = (50,30,10,20,40)
Print("The tuple1 is :", t1)
t2 = Sorted(t1)
Print("The tuple2 is :", t2)
t3 = Sorted(t1, reverse = True)
Print("The tuple3 is :", t3)

o/p:
The tuple1 is : (50,30,10,20,40)
The tuple2 is : (10,20,30,40,50)
The tuple3 is : (50,40,30,20,10).

(3) What is dictionary? Describe Python dictionary in detail discussing its op & method.

DICTIONARY: ~~and example~~ What is dictionary?

* The dictionary is unordered Collection of items that can be in the form of key-value pairs.
* It Contains collection of indices called keys and Collection of values.
* Each key is associated with a single value.



Dictionary.

CREATING A DICTIONARY:

* Dictionary can be created by placing items within the curly braces {} Seperated by Commas.
* An item has a key and Corresponding value expressed as pair {key: value}
* The Value may be any data type and can be repeated, key must be immutable type & must be unique.

For ex,

```
my_dict = {}  # Creating empty dictionary
my_dict = {1: 'Apple', 2: 'orange'}  # dictionary with integer keys.
my_dict = {'name': 'DHIYA', 1: [1,2,3]}  # dict with mixed keys.
my_dict = dict({1: 'apple', 2: 'orange'})  # Create using dict().
```

ACCESS ELEMENTS IN DICTIONARY:

* Elements of dictionary are accessed by using key.
* The keys can be used either inside square brackets [] or with the get() method.

SYNTAX:

```
dictionary_name[key]  /(or)
dictionary_name.get(key)
```

* If the square bracket [] is used, keyError is raised in case a key is not found in the dictionary.
* On the other hand, the get() method returns None if the key is not found..

For ex,

```
my_dict = {1: "Apple", 2: "Orange", 3: "banana"}.
Print (my_dict)  # Print the entire dictionary value.
Print (my_dict[1]) # Print the First pair.
Print (my_dict.get(4)) # Print the Fourth pair
Print (my_dict[4]) # Print the fourth key pair.
```

O/P:

```
{1: 'Apple', 2: 'Orange', 3: 'banana'}.
Apple
None
Print(my_dict[4]).
KeyError: 4
```

Changing and Adding Dictionary Elements:

* Dictionary are mutable, It can be added (or) changed using an assignment operator.
* If the key is already present, then the existing values get updated. In case key is not present, a new (key: value) pair is added to the dictionary.

SYNTAX:

```
dictionary name['key'] = value.
```

For ex,

```
my_dict = {'Name': 'DHIYA', 'Age': 3}
Print (my_dict)
my_dict['Age'] = 5  # Updated the dictionary.
Print (my_dict)
my_dict['Gender'] = "Female" # Add item into the dictionary.
Print (my_dict)
```

O/P: ~~{1: 'Apple', 2: 'Orange', 3: 'banana'}~~.

```
{'Name': 'DHIYA', 'Age': 3}
{'Name': 'DHIYA', 'Age': 5}
{'Name': 'DHIYA', 'Age': 5, 'Gender': 'Female'}
```

Removing Elements from Dictionary:

* The dictionary item can be removed using clear(), popitem() method
* The del keyword also used to remove individual item or entire dictionary.

clear() ⟶ The method is used to remove all the items from the dictionary.

SYNTAX:

dictionary_name.clear().

popitem() ⟶ remove the last element

pop() ⟶ The method is used to remove specified item pair from the dictionary

SYNTAX: dictionary_name.popitem().

del ⟶ The keyword remove individual item (or) entire dictionary.

SYNTAX:

```
del dictionary_name['key']   # Delete specific key pair
del dictionary_name          # delete entire dictionary.
```

For ex,

```
my_dict {1:"Apple", 2:"Orange", 3:"banana", 4:"lemon"}.
Print (my_dict)
my_dict.pop(3)
Print ("After Perform Pop operation:", my_dict)
del my_dict[2]
Print ("After perform del:", my_dict)
my_dict.clear()
print ("After clear operation:", my_dict)
```

O/P:

```
{1:'Apple', 2:'Orange', 3:'banana', 4:'lemon'}
After perform Pop operation: {1:'Apple', 2:'Orange', 4:'lemon'}
After perform del : {1:'Apple', 4:'lemon'}
After clear operation : {}
```

DICTIONARY METHODS:

*There are so many methods available for perform dictionary operation.

1. Copy() → The method is used to return a duplicate of the dictionary

SYNTAX:

```
dictionary_name = dict_name.copy()
```

For ex,

```
my_dict = {1:"Apple", 2:"orange"}
Print(my_dict)
my_dict1 = my_dict.copy()
Print(my_dict1)
```

OUTPUT:

{1:"Apple", 2:"orange"}.

{1:"Apple", 2:"orange"}.

2. get() → The method is used to return the value of specified key.

SYNTAX:

```
dict_name.get(key)
```

For ex,

```
my_dict={1:"Apple", 2:"orange"}
Print(my_dict.get(2))
```

OUTPUT:

{~~1:~~"orange"}.

3. items() → The method is used to return a new object of the dictionary items in (key, value) format.

For ex,

```
my_dict = {1:"Apple", 2:"orange"}
my_dict.items()
```

OUTPUT:

[(1,'Apple'),(2,'Orange')]

4. keys() → The keys() method is used to returns dictionary items key value

For ex,

```
my_dict={1:"Apple", 2:"orange"}.
my_dict.keys().
```

OUTPUT:

[1,2]

5. Values() → The values() method is used to returns dictionary items value only.

For ex,

```
my_dict={1:"Apple", 2:"Orange"}.
my_dict.values()
```

OUTPUT:

['Apple', 'Orange']

6. popitem() ⟶ The method is used to remove & return arbitary item from the dictionary

For ex, my_dict = {1: "Apple", 2: "orange"}

print(my_dict.popitem())

OUTPUT: {2: "Apple"}

* it return & remove the first pair from the dictionary.

7. pop() ⟶ The method is used to remove item with help of key value.

* It returns Corresponding values.

For ex, my_dict = {1: "Apple", 2: "orange"}.

Print(my_dict.pop(2))

OUTPUT: 'orange'.

8. clear() ⟶ It's used to clear the entire dictionary data item.