Two dictionaries can be merged into one by using update() method. It merges the keys and values of one dictionary with another and overwrites values of the same key.

Syntax:

```
Dic_name1.update(Dic_name2)
```

Using this, `Dic_name2` is added to `Dic_name1`.

For example,

```
>>> d1={1:10,2:20,3:30}
>>> d2={4:40,5:50}
>>> d1.update(d2)
>>> print(d1)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
>>> print(d2)
{4: 40, 5: 50}
```

For example,

```
>>> d1={1: 10, 2:30, 3: 30, 5: 40, 6: 60}
>>> d2={1:10,2:20,3:30}
>>> d1.update(d2)
>>> print(d1)
{1: 10, 2: 20, 3: 30, 5: 40, 6: 60}
>>> print(d2)
{1: 10, 2: 20, 3: 30}
```

Value of key 2 is replaced with 20 in the dictionary d1.

8.16 REMOVING AN ITEM FROM DICTIONARY

We can remove an item from the existing dictionary by using del command or pop() method;

1. **using del command:**

   The keyword 'del' is used to delete the key present in the dictionary. If the key to be deleted is not found, then it raises an error.

   Note: We can remove an item from the existing dictionary by using del command or using pop() method.

   Syntax:

   ```
   del dicname[key]
   ```

   For example,

   ```
   >>> A={"mon":"monday","tue":"tuesday","wed":"wednesday",
         "thu":"thursday"}
   >>> del A['wed']
   >>> print(A)
   {'mon': 'monday', 'tue': 'tuesday', 'thu': 'thursday'}
   >>> del A['fri']
   ```

   If particular key is not there, then it will result in KeyError.

```
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    del A['fri']
KeyError: 'fri'
```

If you want to delete the entire dictionary, then give the dictionary name along with del keyword.

For example,

```
>>>del A
>>>A
Traceback (most recent call last):
  File "<pyshell#31>", line 1, in <module>
    A
NameError: name 'A' is not defined
```

The dictionary named A no longer exists in the memory.

2. **using pop() method:** pop() method will not only delete the item specified by the key from the dictionary but also return the deleted value.

Syntax:

```
Dictname.pop(key)
>>> D3={'mon':'Monday','tue':'tuesday','wed':'wednesday'}
>>> D3.pop('tue')
'Tuesday'
>>> print(D3)
{'mon': 'monday', 'wed': 'wednesday'}
```

It will return the deleted value.

8.17 'in' AND 'not in' MEMBERSHIP OPERATOR (in Dictionary)

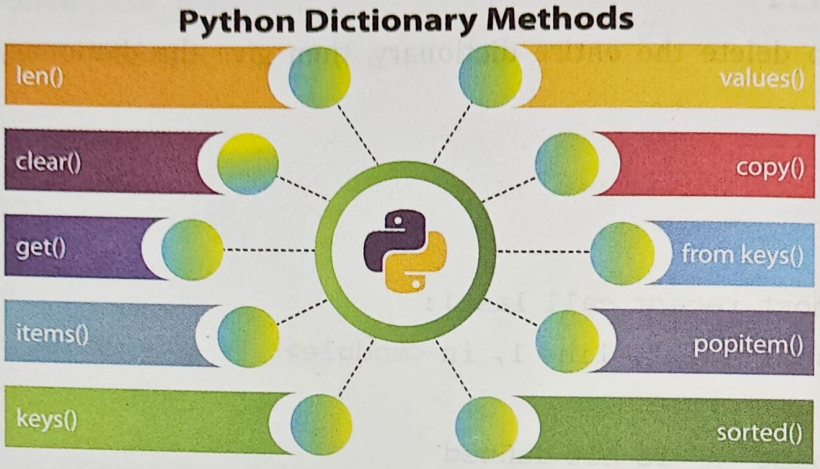
The 'in' operator checks whether a particular key is there in the dictionary. It returns True if specified key element appears in the dictionary, otherwise returns False.

```
>>> A={"mon":"monday","tue":"tuesday","wed":"wednesday",
  "thu":"thursday"}
>>> 'thu' in A
True
>>> 'fri' in A
False
>>> 'fri' not in A
True
```

CTM: Membership operation in, and functions like min(), max() and sum() apply only to the keys in a dictionary.

8.18 COMMON DICTIONARY FUNCTIONS AND METHODS

Python provides you with a number of ways to manipulate the dictionary. Let's check out some of the important ones with examples.



len()

This method returns number of key-value pairs in the given dictionary.

Syntax:

```
len(d) # d is dictionary
```

returns number of key-value pairs in the dictionary.

```
>>> d1={1: 10, 2: 30, 3: 30, 5: 40, 6: 60}
>>> len(d1)
5
```

clear()

It removes all items from the particular dictionary.

Syntax:

```
d.clear() # d is dictionary
```

For example,

```
>>> D={1:'one',2:'two',3:'three'4:'four'}
>>> print(D)
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
>>> D.clear()
>>> print(D)
{}
```

get() method

The get() method returns a value for the given key. If key is not available, then returns default value None.

Syntax:

Following is the syntax for **get()** method:

```
dict.get(key, default = None)
```

key: This is the key to be searched in the dictionary.

default: This is the value to be returned in case the key does not exist.

This method returns a value for the given key. If key is not available, then returns default value None.

For example,

```
>>> D1={'sun':'Sunday','mon':'Monday','tue':'Tuesday','wed':'wednesday',
        'thu':'Thursday'}
>>> D1.get ('wed')
'Wednesday'
>>> print(D1.get('fri'))
None
```

will return None as key does not exist

You can specify your own message also:

```
>>> D1.get('fri','never')
'never'
>>> D1.get('mon','never')
'Monday'
```

Here, specified key is not present in the dictionary, so the specified message has been displayed.

If the specified key is present in the dictionary, then the associated value has been displayed regardless of the customized message.

items()

It returns the content of dictionary (dictionary items) as a list of tuples having key-value pairs.

Syntax:

```
D.items()  #D dictionary
```

Note: items() returns a list of tuples having key-value pairs.

For example,

```
>>> D={'sun':'Sunday','mon':'Monday','tue':'Tuesday','wed':'Wednesday',
       'thu':'Thursday'}
>>> D.items()
dict_items([('sun', 'Sunday'), ('mon', 'Monday'), ('tue', 'Tuesday'),
           ('wed', 'Wednesday'), ('thu', 'Thursday')])
```

keys()

It returns a list of keys from key-value pairs in a dictionary.

Syntax:

```
D.keys()  #D dictionary
```

For example,

```
>>> D={'sun':'Sunday','mon':'Monday','tue':'Tuesday','wed' :'Wednesday',
       'thu':'Thursday'}
>>> D.keys()
dict_keys(['sun', 'mon', 'tue', 'wed', 'thu'])
```

values()

It returns a list of values from key-value pairs in a dictionary.

Syntax:

```
D.values()  #D dictionary
```

```
>>> D={'sun':'Sunday','mon':'Monday','tue':'Tuesday','wed':'Wednesday',
       'thu':'Thursday'}
>>> D.values()
dict_values(['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday'])
```

copy()

We cannot copy a dictionary by using assignment ('=') operator as it will create a reference to the same dictionary variable and modifications will also be reflected in both dictionaries. We can use built-in function copy() to create a new dictionary and modifications done on base dictionary will not be reflected.

Syntax:

```
Dict_new=Dict_old.copy()
```

OR

```
D.copy()#D dictionary
```

For example,

```
>>> d1={"January":31,"February":28 }
>>> d2=d1
>>> d1
{'January': 31, 'February': 28 }
>>> d2
{'January': 31, 'February': 28 }
>>> d2["March"]=31
>>> d2
{'January': 31, 'February': 28, 'March': 31 }
>>> d1
{'January': 31, 'February': 28, 'March': 31 }
>>> d3=d1.copy()
>>> d3
{'January': 31, 'February': 28, 'March': 31}
>>> d1["April"]=30
>>> d3
{'January': 31, 'February': 28, 'March': 31 }
>>> d1
{'January': 31, 'February': 28, 'March': 31, 'April': 30}
```

Assigning a dictionary

Changes made to d2 are reflected in d1

Copying a dictionary

Changes made in d1 are not reflected in d3

prog_dict_copy1.py - C:/Users/preeti/AppData/Local/Programs/Python/Python37-32/prog_...

File Edit Format Run Options Window Help

```
D1 ={'Name':'Radhika','DOB':'2002-03-11','Marks':'98'}
print("D1 :",D1)
D2 =D1.copy()
print("D2 :",D2)
print("Location of D1",id(D1))
print("Location of D2",id(D2)) #Copy maintained at different id
```

Ln: 10 Col: 0

```
>>>
 RESTART: C:/Users/preeti/AppData/Local/Programs/Python/Python37-32
py1.py
D1 : {'Name': 'Radhika', 'DOB': '2002-03-11', 'Marks': '98'}
D2 : {'Name': 'Radhika', 'DOB': '2002-03-11', 'Marks': '98'}
Location of D1 43201520
Location of D2 45896496
>>>
```

Practical Implementation–17

WAP to store students' information like admission number, roll number, name and marks in a dictionary and display information on the basis of admission number.

Code:

prgm_6.py - C:/Chapter 8/prgm_6.py (3.6.5)

File Edit Format Run Options Window Help

```
SCL=dict()
i= 1
n=int(input("Enter number of entries:"))
while i<=n:
    Adm=input("\nEnter admission no of a  student:")
    nm=input("Enter name of the student:")
    section=input("Enter class and section:")
    per=float(input("Enter percentage of a student:"))
    b=(nm,section,per)
    SCL[Adm]=b
    i = i+1
l = SCL.keys()

for i in l:
    print ("\nAdmno- ", i," :")
    z= SCL[i]
    print ("Name\t","class\t","per\t")
    for j in z:
        print(j, end="\t")
```

Ln: 21 Col: 0

Output:

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Type "copyright", "credits" or "license()" for more information.
>>>
===================== RESTART: C:/Chapter 8/prgm_6.py =====================
Enter number of entries:4

Enter admission no of a  student:100
Enter name of the student:AUSHIM
Enter class and section:XIA
Enter percentage of a student:98

Enter admission no of a  student:200
Enter name of the student:NISHANT
Enter class and section:XIB
Enter percentage of a student:97

Enter admission no of a  student:300
Enter name of the student:RAM
Enter class and section:XIC
Enter percentage of a student:85

Enter admission no of a  student:400
Enter name of the student:RIYA
Enter class and section:XID
Enter percentage of a student:78

Admno-  100  :
Name      class    per
AUSHIM  XIA      98.0
Admno-  200  :
Name      class    per
NISHANT XIB      97.0
Admno-  300  :
Name      class    per
RAM     XIC      85.0
Admno-  400  :
Name      class    per
RIYA    XID      78.0
Ln: 7  Col: 0
```

Practical Implementation–18

Write a program to enter names of employees and their salaries as input and store them in a dictionary.

```
#Program to create a dictionary with names of the employees and their salary
num = int(input("Enter the number of employees whose data to be stored: "))
count = 1
employee = dict() #create an empty dictionary
while count <= num:
    name = input("Enter the name of the Employee: ")
    salary = int(input("Enter the salary: "))
    employee[name] = salary
    count += 1
print("\n\nEMPLOYEE_NAME\tSALARY")
for k in employee:
    print(k,'\t\t',employee[k])
```

```
>>>
 RESTART: C:/Users/preeti/AppData/Local/Programs/Python/P
t.py
Enter the number of employees whose data to be stored: 2
Enter the name of the Employee: Teena
Enter the salary: 45000
Enter the name of the Employee: Ritu
Enter the salary: 55000


EMPLOYEE_NAME    SALARY
Teena             45000
Ritu              55000
```

Practical Implementation–19

Write a program to count the number of times a character appears in a given string using a dictionary.

```
#Program to count the number of times a character appears in a given string
str = input("Enter a string: ")
dict1 = {} #creates an empty dictionary
for ch in str:
    if ch in dict1: #if next character is already in the dictionary
        dict1[ch] += 1
    else:
        dict1[ch] = 1 #if ch appears for the first time
for key in dict1:
    print(key,':',dict1[key])
```

```
>>>
 RESTART: C:/Users/preeti/AppData
arater.py
Enter a string: Hello Python
H : 1
e : 1
l : 2
o : 2
  : 1
P : 1
y : 1
t : 1
h : 1
n : 1
```

fromkeys()

This function is used to create dictionary from a collection of keys (tuple/list).

Syntax:

```
D.fromkeys(<ck>,<v>)    #D dictionary, <ck> collection of keys
                                       <v> default value to be assigned
```

The values passed to this function is a default value for all the defined indexes, else None is assigned to all the items.

```
prog_from_dict_keys.py - C:\Users\preeti\AppData\Local\Programs\Python\Pyth...
File  Edit  Format  Run  Options  Window  Help
Keys1 =[1,2,3,4]; values1=1000
D1=dict.fromkeys(Keys1,values1) #values1 is default value
print(D1)
Keys2 =('A','B','C','D'); values2="Undefined"
D2=dict.fromkeys(Keys2,values2)
print(D2)
Keys3 =('100','200','300','400')
D3=dict.fromkeys(Keys3) #values argument is missing
print(D3) #None is assigned if values is missing
D1=dict.fromkeys([100,200,300,400],[1,2,3,4])
print(D1)
```

```
>>>
 RESTART: C:\Users\preeti\AppData\Local\Programs\Python\Python37-32\prog_from_di
ct_keys.py
{1: 1000, 2: 1000, 3: 1000, 4: 1000}
{'A': 'Undefined', 'B': 'Undefined', 'C': 'Undefined', 'D': 'Undefined'}
{'100': None, '200': None, '300': None, '400': None}
{100: [1, 2, 3, 4], 200: [1, 2, 3, 4], 300: [1, 2, 3, 4], 400: [1, 2, 3, 4]}
```

Alternatively, zip() method in Python can also be used to create a dictionary by zipping two or more lists and creating a dictionary using dict() taking zipped lists as arguments to it. You can create a dictionary with a list of zipped keys and values, as exhibited in the example below:

```
>>> D1=dict(zip([100,200,300,400],[1,2,3,4,5]))
>>> D1
{100: 1, 200: 2, 300: 3, 400: 4}
```

```
>>> keys=[1,2,3,4,5]; values=['a','b','c','d','e']
>>> D=dict(zip(keys,values))
>>> D
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}
```

popitem()

This method removes last item (last added key-value pair) from dictionary and returns this deleted item. Pairs are returned in Last In First Out (LIFO) order.

Syntax:

```
D.popitem()                    #D dictionary
```

```
prog_pop_item.py - C:/Users/preeti/AppData/Local/Programs/Python/Python37...
File Edit Format Run Options Window Help
Student= {"Rollno":10,"Name":"Harsh","Stream":"Science"}
print(Student)
S =Student.popitem()
print(S,end="removed from")
print(Student)
```

```
>>>
 RESTART: C:/Users/preeti/AppData/Local/Programs/Python/Python37-32/
m.py
{'Rollno': 10, 'Name': 'Harsh', 'Stream': 'Science'}
('Stream', 'Science')removed from{'Rollno': 10, 'Name': 'Harsh'}
>>>
```

setdefault()

This method returns the value of the item with the specified key. If the key does not exist, it inserts the key with the specified value.

Syntax:

```
<Value>=<Dict>.setdefault(<key>,<default_value>)
```

The setdefault() function returns:

- Value of the key, if it is in the dictionary
- None, if key is not in the dictionary and default_value is not specified
- Default_value, if key is not in the dictionary and default_value is specified

```
prog_set_default1.py - C:/Users/preeti/AppData/Local/Programs/Pytho...
File Edit Format Run Options Window Help
D= {'Name': "Shaurya",'Gender':'Male'}
print(D)
D_Name= D.setdefault('Name',"Name not Available")
D_DOB= D.setdefault('DOB',"Date not Available")
D_Gender= D.setdefault('Gender')
D_Mobile= D.setdefault('Mobile')
print("Name :",D_Name)
print("DOB :",D_DOB)
print("Gender :",D_Gender)
print("Mobile :",D_Mobile)
```

```
>>>
 RESTART: C:/Users/preeti/AppData/Local/
ault1.py
{'Name': 'Shaurya', 'Gender': 'Male'}
Name : Shaurya
DOB : Date not Available
Gender : Male
Mobile : None
>>>
```

max() and min()

The method max() returns key having maximum value. On the contrary, min() returns key having minimum value.

```
prog_max_min_dict.py - C:/Users/preeti/AppData/Local/Programs/Pyth...
File Edit Format Run Options Window Help
D={'Student1':80,'Student2':78,'Student3':76}
print("Highest Key with its value: ",max(D.items()))
print("Lowest Key with its value: ",min(D.items()))
print("Highest Key:",max(D))
print("Lowest Key:",min(D))
print("Highest Value:",max(D.values()))
print("Lowest Value:",min(D.values()))
```

```
>>>
 RESTART: C:/Users/preeti/AppData/Local/Programs/
_dict.py
Highest Key with its value:  ('Student3', 76)
Lowest Key with its value:  ('Student1', 80)
Highest Key: Student3
Lowest Key: Student1
Highest Value: 80
Lowest Value: 76
>>>
```

sorted()

This method sorts the elements of a dictionary by its key or value.

```
prog_sorted_dict.py - C:/Users/preeti/AppData/Local/Progra...
File Edit Format Run Options Window Help
D={'Student1':80,'Student2':78,'Student3':76}
L1= sorted(D)
print(L1)
L2=dict(sorted(D.items()))
print(L2)
L3=sorted(D.values())
print(L3)
```

```
>>>
 RESTART: C:/Users/preeti/AppData/Local/Programs/
dict.py
['Student1', 'Student2', 'Student3']
{'Student1': 80, 'Student2': 78, 'Student3': 76}
[76, 78, 80]
>>>
```

8.19 DIFFERENCE BETWEEN LIST, TUPLE AND DICTIONARY

The following table explains the difference between list, tuple and dictionary:

List	Tuple	Dictionary
A list is mutable.	A tuple is immutable.	A dictionary is mutable.
Lists are dynamic.	Tuples are of fixed size.	Keys are of immutable type but values can be mutable.
Lists are enclosed in square brackets [] and each element is separated by a comma.	Tuples are enclosed in parentheses () and each element is separated by a comma.	Dictionaries are enclosed in curly braces {} and consist of key-value pair. Each key is separated from its value by a colon.
Heterogeneous	Heterogeneous	Heterogeneous

Example:
`List = [10, 'hello', 15.5]`
	Example:
	`Words = ("spam", "eggs")`
	Or
	`Words = "spam", "eggs"`
		Example:
		`Dict = {"ram": 26, "abi": 24}`

Access: `print(List[0])`	Access: `print(Words[0])`	Access: `print(Dict["ram"])`
Can contain duplicate elements.	Can contain duplicate elements. Faster compared to lists.	Cannot contain duplicate keys, but can contain duplicate values.
Slicing can be done.	Slicing can be done.	Slicing cannot be done.
Usage: • List is used if a collection of data doesn't need a random access. • List is used when data can be modified frequently.	Usage: • Tuple can be used when data cannot be changed. • A tuple is used in combination with a dictionary, *i.e.*, a tuple might represent a key.	Usage: • Dictionary is used when there is a logical association between key-value pair. • Used when in need of fast look-up for data, based on a custom key. • Dictionary is used when data is being constantly modified.

MEMORY BYTES

- A tuple is an immutable sequence of values which can be of any type and they are indexed by an integer.
- Creating a tuple is as simple as putting values separated by a comma. Tuples can be created using parentheses ().
- To create a tuple with single element, the final comma is necessary.
- Python provides various operators like '+', '*', 'in', 'not in', etc., which can be applied to tuples.

- Traversing a tuple means accessing each element of a tuple. This can be done by using looping statement either *for* or *while*.
- Python provides us with a number of tuple methods like len(), count(), any(), tuple(), max(), min(), mean(), sum(), sorted(), index(), etc.
- We can compare two tuples by using comparison operators, *i.e.*, <,>,==,!=, etc.
- For tuple assignment, number of variables on the left side of assignment and the number of elements on the right side must match.
- del statement is used to delete a tuple.
- A Python dictionary is a mapping of unique keys to values. It is a collection of key-value pairs. Dictionaries are mutable which means they can be changed.
- In a dictionary, each key maps to a value. The association of a key and a value is called a **key-value pair**.
- To create a dictionary, key-value pairs are separated by a comma and are enclosed in two curly braces {}. In key-value pair, each key is separated from its value by a colon (:).
- Keys are unique and act as the index.
- Keys of a dictionary are of immutable type but values can be mutable.
- We can add new elements to the existing dictionary, extend it with single pair of values or join two dictionaries into one.
- You can also update a dictionary by modifying existing key-value pair or by merging another dictionary with an existing one.
- Python provides you with a number of dictionary methods like len(), pop(), items(), keys(), values(), get(), etc.

OBJECTIVE TYPE QUESTIONS

1. Fill in the blanks.

(a) Elements in a tuple can be of ..... type.

(b) Tuples are indexed by an ..... .

(c) Tuples are ....., you cannot update or edit the elements.

(d) ..... function is used to convert a sequence data type into tuple.

(e) Like lists, dictionaries are ..... which means they can be changed.

(f) A Python dictionary is a ..... of unique keys to values.

(g) The association of a key and a value is called a ..... .

(h) To create a dictionary, key-value pairs are separated by ..... .

(i) In key-value pair, each key is separated from its value by a ..... .

(j) ..... function returns the keys of the dictionary.

(k) The ..... statement removes a dictionary object along with its items.

(l) ..... function will delete the last inserted key-value pair from the dictionary.

2. State whether the following statements are True or False.

(a) An element in a dictionary is a combination of key-value pair.

(b) Internally, dictionaries are indexed on the basis of values.

(c) We can repeat a key in dictionary.

(d) In dictionary, two unique keys can have the same values.

(e) A tuple can be used as the key in a dictionary if it contains immutable elements.

(f) clear() function is used to delete a dictionary.

(g) A tuple is a mutable data type.

(h) Lists can be used as keys in a dictionary.

(i) Updating an element in a dictionary at runtime is possible.

(j) We can create immutable dictionaries in Python.

(k) A tuple can store elements of different data types.

(l) A tuple cannot store list as an element.

(m) Creating individual values from a tuple's elements is called packing.

(n) We cannot delete a dictionary once created.

3. Multiple Choice Questions (MCQs):

(a) Which of the statement(s) is/are correct?
(i) Python Dictionary is an ordered collection of items.
(ii) Python Dictionary is a mapping of unique keys to values.
(iii) Dictionary is mutable.
(iv) All of these

(b) If `tup = (20,30,40,50)`, which of the following is incorrect?
(i) `print(tup[3])` (ii) `tup[2] = 56`
(iii) `print(max(tup))` (iv) `print(len(tup))`

(c) Consider two tuples given below:
```
tup1 = (1,2,4,3)
tup2 = (1,2,3,4)
```
What will the following statement print: `print(tup1 < tup2)`
(i) `True` (ii) `False` (iii) `Error` (iv) None of these

(d) Which function returns the number of elements in the tuple?
(i) `len()` (ii) `max()` (iii) `min()` (iv) `count()`

(e) To create an empty dictionary, we use
(i) `d1={}` (ii) `d1=[]` (iii) `d1=()` (iv) `d1=={}`

(f) In a dictionary, the elements are accessed through
(i) Key (ii) Value (iii) Index (iv) None of these

(g) Keys of a dictionary must be
(i) Similar (ii) Unique
(iii) Can be similar or unique (iv) All of these

(h) To create new dictionary with no items:
(i) `Dict` (ii) `dict()` (iii) `d1=[ ]` (iv) None of these

(i) Which function is used to return a value for the given key?
(i) `len()` (ii) `get()` (iii) `keys()` (iv) None of these

(j) Which function is used to remove all items from a particular dictionary?
(i) `clear()` (ii) `pop()` (iii) `delete()` (iv) `rem()`

(k) Which one of the following is correct to insert a single element in a tuple?
(i) `T=4` (ii) `T=(4)` (iii) `T=(4,)` (iv) `T =[4,]`

(l) Which of the following functions will return key-value pairs of the dictionary?
(i) `key()` (ii) `values()` (iii) `items()` (iv) `get()`

(m) Which of the following will delete key-value pair for key = "Red" from a dictionary D1?
(i) `delete D1("Red")` (ii) `del.D1("Red")`
(iii) `del D1["Red"]` (iv) `del D1`

(n) What will be the output?
```
D1={"Rahul":56,"Virat":99}
print("virat" in D1)
```
(i) `True` (ii) `False` (iii) No output (iv) `Error`

(o) Which of the following creates a tuple?
(i) `t1=("a","b")` (ii) `t1[2]=("a","b")`
(iii) `t1=(5)*2` (iv) None of these

SOLVED QUESTIONS

1. What is a tuple?

Ans. A tuple is an immutable sequence of elements of any data type each of which is indexed by an integer.

2. What are the differences between lists and tuples?

Ans. The differences between tuples and lists are that tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

3. What is the similarity between strings and tuples?

Ans. Tuples are immutable like strings, *i.e.*, once created, we cannot update or edit their elements.

4. Tuples can be used as keys in Python dictionaries. Explain.

Ans. Yes, tuples can be used as keys in Python dictionaries as keys of dictionary cannot be modified and tuples are immutable by default and hence can be used.

5. Tuple is an ordered immutable sequence of objects. Justify your answer.

Ans. Tuple is an ordered sequence of objects as each element is accessed by its index and it is immutable as we cannot change the values in place once created.

6. (a) What is the length of the tuple shown below?

```
T = ( ( ( ( 'a', 1 ), 'b' , 'c' ), 'd' , 2 ) , 'e' , 3 )
```

Ans. 3

(b) What is the difference between (30) and (30,)?

Ans. When we use type function, then (30) is a type of 'int' class whereas (30,) is a type of tuple which contains one element.

7. What is a dictionary?

Ans. A Python dictionary is a mapping of unique keys to values. It is a collection of key-value pairs. It is mutable which means it can be changed.

8. What are the differences between dictionary and list?

Ans. The differences between dictionary and list are:

(i) In dictionary, key-value pairs are enclosed within curly braces whereas in lists, elements are enclosed within square brackets.

(ii) In dictionary, keys act as index and are unique but in list, index value is an integer.

9. Can list be used as a key to dictionary? Justify your answer.

Ans. Lists cannot be used as key to dictionary as they are mutable while keys of dictionary are immutable and unique.

10. Write different ways of creating tuples with one item and print the same.

Ans. (a)
```
tup1=5,
print(tup1)
```
(b)
```
tup1=(5,)
print(tup1)
```
(c)
```
tup1=tuple((10,))
print(tup1)
```

11. Write a statement to add a single item in a tuple.

Ans.
```
>>> t=(10,20,30,40)
>>> t=t+(60,)
```

12. Write different ways of creating tuples with the following numbers:
5,10,15,20

Ans. (a) `t1=(5,10,15,20)`
(b) `t1=5,10,15,20`
(c) `t1=5,10,15,20,`
(d) `t1=tuple((5,10,15,20))`
(e) `t1=tuple([5,10,15,20])`

13. Following are the statements for creating tuples. Find the errors in the statements and rewrite the same after correcting them.

(a) `tup1= 1,6,a,8,`
(b) `tup2=(10)`
(c) `tup1=((0,1,2,3)['my','book'])`
(d) `tup1=((0,1,2,3)(4,5,6))`
(e) `tup1=('a','b','c'[1,2,3])`

Ans. (a) `tup1= 1,6,'a',8,`
(b) `tup2=(10,)`
(c) `tup1=((0,1,2,3),['my','book'])`
(d) `tup1=((0,1,2,3),(4,5,6))`
(e) `tup1=('a','b','c',[1,2,3])`

14. Consider the following code and find out the error:

(a)
```
tup1=(1,'school',6,7)
print(max(tup1))
```
(b)
```
t=(1,'s','h',5,'p')
t[1]='o'
```
(c)
```
T1=(7)
T2=(1,2,3)
T3=T1+T2
```
(d)
```
T1=(1,2,3)
T2=(4,5,6)
T3=(8,9)
a,b,c = T1,T2
```
(e)
```
T1=(1,2,3)
T4= T1*(2,)
```
(f)
```
tup1=(1,2,3,4)
tup1=tup1+10
print(tup1)
```
(g)
```
tup1=("5",)*'3'
print(tup1)
```
(h)
```
t1=(1,2,3,4)
a,b,c=t1
```

Ans. (a) max() will work only if elements in a tuple are of the same data type.
(b) Tuples are immutable, we cannot change the value of any element.
(c) '+' operator joins two tuples, here (7) is 'int', so will give error.
(d) For tuple assignment, left-hand side and right-hand side elements should be the same. T3 is not there on right-hand side.
(e) '*' operator requires a tuple and an integer as operands.
(f) The '+' operator simply performs a concatenation with tuples. Here 10 is an integer.
(g) '*' operator requires a tuple and an integer as operands. Here '3' is a string.
(h) For unpacking a tuple, number of variables on the left side of assignment and the number of elements in the tuple must match.

15. Find errors, underline them and rewrite the same after correcting the following code:

```
d1 = dict[]
i = 1
n = input("Enter number of entries:")
while i<=n:
  a = input("Enter name:")
  b = input("Enter age:")
  d1(a) = b
  i = i + 1
l = d1.key[]
for i in l:
  print(i, '\t', 'd1[i]')
```

Ans.
```
d1 = dict()
i = 1
n = int(input("Enter number of entries:"))
while i<=n:
  a = input("Enter name:")
  b = input("Enter age:")
  d1[a] = b
  i = i + 1
l = d1.keys()
for i in l:
  print(i, '\t', d1[i])
```

16. Write the output of the following codes:

(a)
```
>>> d1={1:10,2:20,3:30,4:40,5:50}
>>> d1.items()
>>> d1.keys()
>>> d1.values()
```

(b)
```
d1={1:10,2:20,3:30,4:40}
d2={5:50,6:60,7:70}
d1.update(d2)
print(d1)
```

(c)
```
d1={1:10,2:20,3:30,4:40,5:50,6:60,7:70}
del d1[3]
print(d1)
```

(d)
```
d1={1:10,2:20,3:30,4:40,5:50,6:60,7:70}
d1.pop(5)
print(d1)
```

(e)
```
d1={1:10,2:20,3:30,4:40,5:50,6:60,7:70}
len(d1)
d1.get(6,60)
d1[6]=65
d1[8]=35
print(d1)
d1.clear()
print(d1)
```

(f) What will be stored in variables a, b, c, d, e, f, g, h after the following statements?
```
perc = (88, 85, 80, 88, 83, 86)
a = perc[2:2]
b = perc[2:]
c = perc[:2]
d = perc[:-2]
e = perc[-2]
f = perc[2:-2]
g = perc[-2:2]
h = perc[:]
```

Ans. (a)
```
dict_items[(1, 10), (2, 20), (3, 30), (4, 40), (5, 50)]
dict_keys[1, 2, 3, 4, 5]
dict_values[10, 20, 30, 40, 50]
```

(b) {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60, 7: 70}

(c) {1: 10, 2: 20, 4: 40, 5: 50, 6: 60, 7: 70}

(d) { 5 : 5 0 }

{1: 10, 2: 20, 3: 30, 4: 40, 6: 60, 7: 70}

(e) 7
60
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 65, 7: 70, 8: 35}
{}

(f) ()
(80, 88, 83, 86)
(88, 85)
(88, 85, 80, 88)
83
(80, 88)
()
(88, 85, 80, 88, 83, 86)

17. Write a Python program to display all the elements of the following tuple except 'd'.

```
t1=('w','o','r','d','s')
```

Ans.

```
t1=('w','o','r','d','s')
t1=t1[0:3]+t1[-1:]
print(t1)
```

Output:

```
('w', 'o', 'r', 's')
```

18. Write a Python program to remove an element '3' from the following tuple:

```
t1=(1,2,3,4,5)
```

Ans.

```
t1 = (1,2,3,4,5)
L = list(t1)
L.remove(3)
t1 = tuple(L)
print(t1)
```

Output:

```
(1, 2, 4, 5)
```

Alternatively, using pop() method:
with pop() we have to give index of element 3 as shown:

```
t1 = (1,2,3,4,5)
L = List(t1)
L.pop(2) #2 is the index value for element 3
O
R
del L[2]
t1 = tuple(L)
print(t1)
```

Output:

```
(1, 2, 4, 5)
```

19. `t1=(10,20,"book",30,9.5,"item",[12,13],(3,4),30,5,30)`

Consider the above tuple 't1' and answer the following questions:

(a) `len(t1)` (b) `t1[:6]`
(c) `t1[-8:-4]` (d) `t1[5:]`
(e) `t1.index(20)` (f) `t1.index(30)`
(g) `t1.index(30,7,10)` (h) `t1.count(30)`
(i) `t1[:1]*5` (j) `any(t1)`

Ans.
(a) 11 (b) (10, 20, 'book', 30, 9.5, 'item')
(c) (30, 9.5, 'item', [12, 13]) (d) ('item', [12, 13], (3, 4), 30, 5, 30)
(e) 1 (f) 3
(g) 8 (h) 3
(i) (10, 10, 10, 10, 10) (j) True

20. `T1 = ("TEA POT",[1,2,'3'],'S',(3,4,6),"book",10)`

Based on tuple T1 above, write the output for the following expressions:

(a) `T1[1][1]` (b) `10 in T1`
(c) `50 not in T1` (d) `T1[4][3]`
(e) `T1[1][2]*3` (f) `T1[-5:-3]+T1[3]`
(g) `T1[0]+T1[2]` (h) `T1[4:4]`

Ans. (a) 2 (b) True
(c) True (d) 'k'
(e) '333' (f) ([1, 2, '3'], 'S', 3, 4, 6)
(g) 'TEA POTS' (h) ()

21. Write the output of the following statements:

(a) `>>> tuple("book")` (b) `>>> tuple([1,2,4])`
(c) `>>> ("Riya",)*3` (d) `>>> max(10,12,73,34)`
(e) `>>> min(40,20,100,60)`

Ans. (a) ('b', 'o', 'o', 'k') (b) (1, 2, 4)
(c) ('Riya', 'Riya', 'Riya') (d) 73
(e) 20

22. Compare the tuples and write the output:

(a)
```
>>> tup1=(1,2,3)
>>> tup2=(1.0,2.0,3.0)
>>> tup1==tup2
```

(b)
```
>>> tup1=('FARIDA','VAIBHAV','MINI','RAGHAV')
>>> tup2=('Ravi','Mala','AMAN','SAKSHI')
>>> tup1<tup2
```

(c)
```
>>> tup1=(10,20,30,40,50)
>>> tup2=(4,5,10,20,30,40,50)
>>> tup2>tup1
```

(d)
```
>>> tup1!=tup2
```

Ans. (a) True (b) True
(c) False (d) True

23.
```
t1=(1,2,3,4)
t2=(10,20,30,40)
t3=(100,200,300,400)
```

Write the output of the following:

(a)
```
t1,t2=t2,t1
print(t1)
print(t2)
```

(b)
```
t1,t3=t3,t1
print(t1)
print(t3)
```

(c)
```
t3,t2,t1=t1,t2,t3
print(t1)
print(t2)
print(t3)
```

Ans. (a) (10, 20, 30, 40)
(1, 2, 3, 4)
(b) (100, 200, 300, 400)
(10, 20, 30, 40)
(c) (100, 200, 300, 400)
(10, 20, 30, 40)
(1, 2, 3, 4)

24. Neeti has partially developed the code given below to create a dictionary comprising month name with number of days in Python. Help her in completing the program:

```
Days={"January":31,"February":28,"March":31,"April":30,"May":31,\
      "June":30,"July":31,"August":31,"September":30,"October":31,\
      "November":30," December":31}
#Display months having 31 days
for Key,Value in _________: #(a)
if _____________: #(b)
  print(Key)
  #display Months and days starting with "A" or "M"
  for Key in Days.keys():
        if _____________: #(c)
  print(_________) #(d)
#change number of days in February as 29
_________________ #(e)
print(Days)
```

(a) Write the function to get both keys and values.
(b) Write a condition to check if number of days is 31.
(c) Write a condition to check month should start with "A" or "M".
(d) Display month and days.
(e) Change number of days as 29 for the month of February.

Ans. (a) `Days.items()`
(b) `Value ==31:`
(c) `Key.startswith(('a','A','m','M')):`
(d) `print(Key,":",Days[key])`
(e) `Days["February"]=29`

25. Write a menu-driven program to input names of n students, total marks, average and result, and store in nested tuple display the following menu:

(a) Result: Display name, marks, average and assign result as "pass" if average is more than 32 and "fail" if less than 33
(b) Display: Display name, total marks and average of all students
(c) Search: Accept name and display other details if found and display not found if not existing
(d) Merit List: Display students getting average marks above 74

Ans.
```
n=int(input("No of students:"))
tup=()
names=[]
for i in range(n):
    name=input("Name:")
    names.append(name)
    t_marks=int(input("Total marks:"))
    avg=float(input("Average:"))
    t=(name,t_marks,avg)
    tup+=(t,)
while True:
    print("MENU")
    print("1. Accept")
    print("2. Display")
    print("3. Search")
    print("4. Merit list")
    print("5. Exit")
    ch=int(input("Enter your choice(1-5):"))
    if ch==1:
        for i in tup:
            print(i[0],"-",i[2])
```

```
            if i[2]>32:
                print("Result: Pass")
            else:
                print("Result: Fail")
        elif ch==2:
                print("Name\tTotal Marks\tAverage")
            for i in tup:
                print(i[0],i[1],"\t",i[2])
        elif ch==3:
                name1=input("Name:")
                if name1 in names:
                        print("Name\tTotal Marks\tAverage")
                        pos=names.index(name1)
                        print(tup[pos][0],tup[pos][1],tup[pos][2])
                else:
                        print("Name not found")
        elif ch==4:
                print("MERIT LIST")
                for i in tup:
                    if i[2]>74:
                        print(i[0])
        elif ch==5:
                break
        else:
                print("Invalid choice")
```

26. Write a Python program to input 'n' names and phone numbers to store it in a dictionary and print the phone number of a particular name.

Ans. Code:

```
phone=dict()
i = 1
n=int(input("Enter number of entries:"))
while i<=n:
    a=input("Enter Name:")
    b=input("Enter Phone no:")
    phone[a]=b
    i = i+1
l = phone.keys()
x=input("Enter name to be searched:")
for i in l:
  if i==x:
    print (x,": phone no is:",phone[i])
    break
else:
    print(x, "does not exist")
```

Output:

```
Enter number of entries:4
Enter Name:Riya
Enter Phone no:9999
Enter Name:Aman
Enter Phone no:1111
Enter Name:Pulkit
Enter Phone no:5555
Enter Name: Ziya
Enter Phone no:2222
Enter name to be searched Aman
Aman : phone no is: 1111
```

27. Write a Python program to input names of 'n' employees and their salary details like basic salary, house rent and conveyance allowance. Calculate total salary of each employee and display.

Ans. Code:

```
d1=dict()
i= 1
n=int(input("Enter number of entries:"))
while i<=n:
        Nm=input("\nEnter name of the employee:")
        basic=int(input("Enter Basic salary:"))
        =int(input("Enter house rent allowance:"))
        ca=int(input("Enter conveyance allowance: "))
        d1[Nm]=[basic,hra,ca]
        i = i+1
l = d1.keys()
print("\nName",'\t\t',"Net Salary")
for i in l:
    salary = 0
    z= d1[i]
    for j in z:
        salary=salary+j
    print(i,'\t\t',salary)
```

Note: list is stored in dictionary as value.

Output:

```
Enter number of entries:4
Enter name of the employee: Nishant
Enter Basic salary: 100000
Enter house rent allowance: 30000
Enter conveyance allowance: 5000
Enter name of the employee: Ram
Enter Basic salary: 50000
Enter house rent allowance: 15000
Enter conveyance allowance: 5000
Enter name of the employee: Aushim
Enter Basic salary: 200000
Enter house rent allowance: 50000
Enter conveyance allowance: 10000
Enter name of the employee: Riya
Enter Basic salary: 25000
Enter house rent allowance: 2000
Enter conveyance allowance: 1000
Name        Net Salary
Nishant     135000
Ram         70000
Aushim      260000
Riya        28000
```

28. Write a Python program to input names of 'n' countries and their capital and currency, store it in a dictionary and display in tabular form. Also search and display for a particular country.

Ans. Code:

```
d1=dict()
i= 1
n=int(input("Enter number of entries:"))
while i<=n:
```

```
    c=input("Enter Country:")
    cap=input("Enter Capital:")
    curr=input("Enter currency of country:")
    dl[c]=(cap,curr)
    i = i+1
l = dl.keys()
print ("\nCountry\t\t","Capital\t\t","Currency")
for i in l:
    z= dl[i]
    print ('\n',i,'\t\t',end="")
    for j in z:
    print (j,'\t\t',end="\t\t")
# searching
x=input("\nEnter Country to be searched: ")
for i in l:
  if i==x:
     print ("\nCountry\t\t","Capital\t\t","Currency\t\t")
     z= dl[i]
     print ('\n',i,'\t\t',end="")
    for j in z:
        print (j,'\t\t',end="\t\t")
    break
```

Note: Tuple is stored in dictionary as value.

Output:

```
Enter number of entries:5
Enter Country:Austria
Enter Capital:Vienna
Enter currency of country:Euro
Enter Country:China
Enter Capital:Beijing
Enter currency of country:Yuan
Enter Country:India
Enter Capital:New Delhi
Enter currency of country:Indian Rupee
Enter Country:France
Enter Capital:Paris
Enter currency of country:Euro
Enter Country:United Kingdom
Enter Capital:London
Enter currency of country:Pound
Country          Capital        Currency
Austria          Vienna         Euro
China            Beijing        Yuan
India            New Delhi      Indian Rupee
France           Paris          Euro
United Kingdom   London         Pound
Enter Country to be searched: India
Country          Capital        Currency
India            New Delhi      Indian Rupee
```

29. Write a program to convert a number entered by the user into its corresponding number in words. For example, if the input is 985, then the output should be 'Nine Eight Five'.

Ans.
```
#Program to convert number into corresponding number in words
#numberNames is a dictionary of digits and corresponding number
numberNames = {0:'Zero',1:'One',2:'Two',3:'Three',4:'Four',\
               5:'Five',6:'Six',7:'Seven',8:'Eight',9:'Nine'}
num = input("Enter any number: ") #number is stored as string
result = ''
for ch in num:
    key = int(ch) #converts character to integer
    value = numberNames[key]
    result = result + ' ' + value
print("The number is:",num)
print("The numberName is:",result)
```

30. Write a program to accept and store Employee name as a key and his salary, allowance and deduction as value in a dictionary Dict for 10 employees. Display the following menu:

(a) Display Total Salary

(b) Display Total Allowance and Deductions

(c) Search an Employee

- When user selects option 1, display a report having Name, Salary, Allowance, Deduction, Gross salary (Salary + allowance) and Net Salary (Gross salary – Deductions)
- When user selects option 2, find total allowance and deductions of all 10 employees.
- When user selects option 3, accept name of the employee and search in the dictionary.

Ans.
```
D={}
for a in range(5):
  name=input("Enter name:")
  salary=float(input("Enter salary:"))
  allowance=float(input("enter allowance:"))
  deduction=float(input("enter deduction:"))
  D[name]=[salary,allowance,deduction]
ch=0
while True:
  print("""
  1. Display the total salary
  2. Display total allowance and deduction
  3. search an employee
  4. exit""")
  ch=int(input("Enter your choice:"))
  if ch==1:
    for Key,value in D.items():
      gross=value[0]+value[1]
      net=gross-value[2]
      print(Key,gross,net)
  elif ch==2:
    total_allowance=0
    total_deduction=0
    for Key,value in D.items():
      total_allowance+=value[1]
      total_deduction+=value[2]
      print("Total allowance:",total_allowance)
      print("Total deduction:",total_deduction)
```

```
elif ch==3:
    n=input("Enter employee name to search:")
    for Key,value in D.items():
        if Key==n:
            print(Key,value[0],value[1],value[2])
        else:
            print("Not found")
elif ch==4:
            break
else:
    print("Wrong choice")
```

31. Write a menu-driven program to input your friends' names and their phone numbers and store them in the dictionary as key-value pair. Perform the following operations on the dictionary:

(a) Display the name and phone numbers of all your friends.

(b) Add a new key-value pair in this dictionary and display the modified dictionary.

(c) Delete a particular friend from the dictionary.

(d) Modify the phone number of an existing friend.

(e) Check if a friend is present in the dictionary or not.

(f) Display the dictionary in sorted order of names.

Ans.

```
n=int(input("No of friends: "))
dict={}
for i in range(n):
    name=input("Name:")
    ph_no=int(input("Phone number:"))
    dict[name]=ph_no
while True:
    print("MENU\n1. Display name & Phone numbers\n\
    2. Add new name\n\
    3. Delete a name\n\
    4. Modify an existing number\n\
    5. Check if a name is there\n\
    6. Display dictionary in sorted order\n\
    7. Exit")
    ch=int(input("Enter your choice(1-6):"))
    if ch==1:
        for i in dict:
            print(i,"-",dict[i])
    elif ch==2:
        name1=input("Enter name:")
        ph_no1=int(input("Enter phone number:"))
        dict[name1]=ph_no1
        print("Modified dictionary:",dict)
    elif ch==3:
        name1=input("Name of friend to be deleted:")
        dict.pop(name1)
        print("Modified dictionary:",dict)
    elif ch==4:
        name1=input("Name of friend:")
        ph_no1=int(input("New ph no:"))
        dict[name1]=ph_no1
```

```
elif ch==5:
    name1=input("Name of friend:")
    if name1 in dict:
        print("Friend is in dictionary")
    else:
        print("Friend is not in dictionary")
elif ch==6:
    D=dict
    dict1=sorted(D.items())
    print("Modified dictionary: ",dict1)
elif ch==7:
    break
else:
    print("Invalid choice")
```

ASSERTION AND REASONING BASED QUESTIONS

The following questions contain two statements: Assertion and Reasoning. Each question has four choices—(i), (ii), (iii), (iv)—only one of which is correct. In the light of these statements, choose the most appropriate option.

(i) Both A and R are true and R is the correct explanation of A.
(ii) Both A and R are true but R is not the correct explanation of A.
(iii) A is true but R is false.
(iv) A is false but R is true.

1. **Assertion (A):** Tuple in Python is an ordered and immutable data type.
   Reasoning (R): Tuples can contain heterogenous data and permit duplicate values as well.
2. Consider the given two statements:
   Statement 1: `t1=tuple('python')`
   Statement 2: `t1[4]='z'`
   Assertion (A): The above code will generate an error.
   Reasoning (R): Tuple is immutable by nature.
3. **Assertion (A):** In Python, tuple is an immutable sequence of data.
   Reasoning (R): Immutable means that any change or alteration in data is mentioned in the same place. The updated collection will use the same address for its storage.
4. Consider the given statements for creating dictionaries in Python:
   `D1= {'A':'CS', 'B':'IP'}`
   `D2= { 'B':'IP', 'A':'CS'}`
   Assertion (A): Output of print (D1==D2) is True.
   Reasoning (R): Dictionary is a collection of key-value pairs. It is not a sequence.
5. **Assertion (A):** Dictionaries are enclosed by curly braces { }.
   Reasoning (R): The key-value pairs are separated by commas (,).
6. **Assertion (A):** The pop() method can be used to delete elements from a dictionary.
   Reasoning (R): The pop() method deletes the key-value pair and returns the value of deleted element.
7. **Assertion (A):** Keys of the dictionaries must be unique.
   Reasoning (R): The keys of a dictionary can be accessed using values.
8. **Assertion (A):** clear() method removes all elements from the dictionary.
   Reasoning (R): len() function cannot be used to find the length of a dictionary.

Ans. **1.** (i) **2.** (i) **3.** (iii) **4.** (ii) **5.** (ii) **6.** (i) **7.** (iii) **8.** (iii)

CASE-BASED/SOURCE-BASED INTEGRATED QUESTIONS

1. Rajmandir Basket is a department store with a wide range of merchandise including groceries, fruits, vegetables and household accessories. It aims to develop a complete software for Billing, Inventory and Accounts management and to provide consumers a user-friendly interface to view item codes and prices of various items.

   Write a program in Python that repeatedly asks the user to enter product names and prices. Store all of them in a dictionary whose keys are product names and values are prices. Also write a code to search an item from the dictionary.

Ans.

prog_departmental_store_dict1.py - C:\Users\preeti\AppData\Local\Programs\Python\Py...

File Edit Format Run Options Window Help

```
#program to store the details of groceries and their prices
dict={}
ch ='y'
while ch =='y' or ch=='Y':
    name=input("Enter the product name:")
    price=eval(input("Enter the product price :"))
    dict[name] =price
    ch=input("Do you want to add more items (Y/N)")
print(dict)
prod_name=input("Enter the product to be searched :")
for i in dict:
    if i == prod_name:
        print("Product found",i,"with price as:",dict[i])
```

```
>>>
 RESTART: C:\Users\preeti\AppData\Local\Programs
ental_store_dict1.py
Enter the product name:Sugar
Enter the product price :60
Do you want to add more items (Y/N)y
Enter the product name:Rice
Enter the product price :80
Do you want to add more items (Y/N)y
Enter the product name:Pulses
Enter the product price :100
Do you want to add more items (Y/N)n
{'Sugar': 60, 'Rice': 80, 'Pulses': 100}
Enter the product to be searched :Rice
Product found Rice with price as: 80
>>>
```

2. The above shopping destination for groceries, Rajmandir, stores mobile numbers of its customers. Mobile numbers are to be stored in the form of a dictionary. Write a Python program to input 'n' names and mobile numbers to store them in a dictionary and to input any name and to print mobile number of that particular customer.

Ans.

prog_cutomer_dict2.py - C:\Users\preeti\AppData\Local\Programs\Python\Python37-32\prog_cut...

File Edit Format Run Options Window Help

```
#program to store the details of customer names and mobile numbers
dict={}
ch ='y'
while ch =='y' or ch=='Y':
    custname=input("Enter the customer name:")
    phone=input("Enter the customer mobile number :")
    dict[custname] =phone
    ch=input("Do you want to add more customer details (Y/N)")
print(dict)
cust_name=input("Enter name of the customer to be searched for :")
for i in dict:
    if i == cust_name:
        print("Customer",cust_name,"has mobile number as:",dict[i])
```

```
>>>
 RESTART: C:\Users\preeti\AppData\Local\Programs\Python\Python37-32\prog_c
_dict2.py
Enter the customer name:Adish
Enter the customer mobile number :98766544232
Do you want to add more customer details (Y/N)y
Enter the customer name:Teena
Enter the customer mobile number :9987654390
Do you want to add more customer details (Y/N)y
Enter the customer name:Ajay
Enter the customer mobile number :9807765488
Do you want to add more customer details (Y/N)n
{'Adish': '98766544232', 'Teena': '9987654390', 'Ajay': '9807765488'}
Enter name of the customer to be searched for :Ajay
Customer Ajay has mobile number as: 9807765488
>>>
```

UNSOLVED QUESTIONS

1. What advantages do tuples have over lists? (NCERT)
2. What are the similarities and differences between tuples and lists?
3. What is a key-value pair in dictionary?
4. What are the differences between tuple and dictionary?
5. What are the differences between strings and tuples?
6. Can we always have same type of values in a tuple? State True or False with justification.
7. What is nested tuple? Explain with example.
8. Write the code to create an empty tuple.
9. Write a Python program to create a tuple with different data types and display.
10. What is the difference between "book" and ('book',)?
11. Write a Python program to combine first 3 elements and last 3 elements from a tuple.

    `T1=('a',1,2,3,4,'b','c','book',10)`

    Output should be:

    `('a',1,2,'c','book',10)`
12. Find errors and rewrite the same after correcting the following code:

    (a) `d1={1:10,2.5:20,3:30,4:40,5:50,6:60,7,70}`
    (b) `d1(9)=90`
    (c) `del d1(2)`
    (d) `pop d1[4]`
    (e) `d1.item()`
    (f) `d1.key()`
    (g) `d1.value()`
    (h) `d1.gets(4,80)`
    (i) `d1.len()`
    (j) `d1.clears()`
13. Suppose

    `d1={1:'one',2:'two',3:'three',4:'four'}`

    `d2={5:'five',6:'six'}`

    Write the output of the following code:

    (a)
    ```
    >>> d1.items()
    >>> d1.keys()
    >>> d1.values()
    >>> d1.update(d2)
    >>> len(d1)
    ```
    (b)
    ```
    >>> del d1[3]
    >>> print(d1)
    >>> d1.pop(4)
    >>> print(d1)
    >>> d1[8]='eight'
    >>> print(d1)
    >>> d1.clear()
    >>> print(d1)
    ```
14. Consider the following code and find out the error:

    (a) `tup1=(10,20,30)`
    `tup2= tup1*(3,)`

    (b) `tup1=('a', 'b', 'c')`
    `tup1=tup1+'d'`

    (c) `tup1=("Riya",)*'4'`

    (d) `t1=(10,20,30,40)`
    `X,Y,Z=t1`

(e) `tup1=('A','school',10.5,70)`
`print max(tup1)`

(f) `t=(10,20,30,40)`
`t[1]='o'`

(g) `T1=('A')`
`T2=('B','C',3)`
`T3=T1+T2`

(h) `T1=(10,20,30,40)`
`T2=(40,50,60)`
`T1,T2,T3=T1,T2`

15. `t1=(100,200,"Global",3,3.5,"Exam",[1,2],(30,40),(3,5,3))`

Consider the above tuple 't1' and answer the following questions:

(a) `len(t1)`
(b) `20 not in t1`
(c) `t1[-8:-4]`
(d) `t1[5:]`
(e) `t1.index(5)`
(f) `t1.index(3,6)`
(g) `10 in t1`
(h) `t1.count(3)`
(i) `any(t1)`
(j) `t1[2]*2`

16. Write the output of the following statements:

(a) `>>> tuple([10,20,40])`
(b) `>>> ("Tea",)*5`
(c) `>>> tuple("Item")`

17. Consider two tuples t1 & t2 given below:

```
t1=(100,200,300)
t2=(10,20,30,40)
```

Write the output of the following statements:

(a)
```
t1,t2=t2,t1
print(t1)
print(t2)
```
(b) `print(t1!=t2)`
(c) `print(t1<t2)`

18. WAP to accept values from a user. Add a tuple to it and display its elements one by one. Also display its maximum and minimum value.

19. Write a Python program to input any values for two tuples. Print it, interchange it, and then compare them.

20. Write a Python program to input 'n' classes and names of their class teachers to store it in a dictionary and display the same. Also, accept a particular class from the user and display the name of the class teacher of that class.

21. Write a Python program to store student names and their percentage in a dictionary, delete a particular student's name from the dictionary. Also, display dictionary after deletion.

22. Write a Python program to input names of 'n' customers and their details like items bought, cost and phone number. Store it in a dictionary and display all the details in a tabular form.