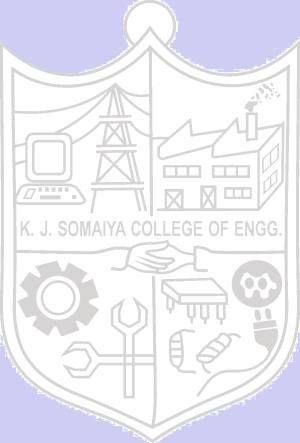
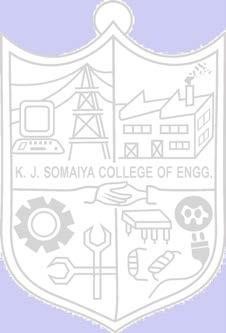
**Experiment No. 5**

**Title: Persistent Object Storage using Shelves**



**Batch:B1 Roll No:1914049 Experiment No.:5**

**Aim:** Persistent Storage of arbitrary python objects using shelve

**Resources needed**: Python IDE

**Theory:**

**Note the diff between dbm and shelve**

The shelve module in Python’s standard library is a simple yet effective tool for persistent data storage when using a relational database solution is not required. The shelf object defined in this module is dictionary-like object which is persistently stored in a disk file. This creates afile similar to dbm database on UNIX like systems. Only string data type can be used as key in this special dictionary object, whereas any picklable object can serve as value.

A “shelf” is a persistent, dictionary-like object. The difference with “dbm” databases is that the values (not the keys!) in a shelf can be essentially arbitrary Python objects — anything that the [pickle](https://docs.python.org/3/library/pickle.html#module-pickle) module can handle. This includes most class instances, recursive data types, and objects containing lots of shared sub-objects. The keys are ordinary strings.

**DBM** stands for DataBase Manager.  
  
These databases use binary encoded string objects as key, as well as value. The database stores data by use of a single key (a primary key) in fixed-size buckets and uses hashing techniques to enable fast retrieval of the data by key.

The shelve module defines three classes as follows −

|  |  |
| --- | --- |
| Sr.No | Shelve Module & Description |
| 1 | Shelf  This is the base class for shelf implementations. It is initialized with dict-like object. |
| 2 | BsdDbShelf  This is a subclass of Shelf class. The dict object passed to its constructor must support first(), next(), previous(), last() and set\_location() methods. |
| 3 | DbfilenameShelf  This is also a subclass of Shelf but accepts a filename as parameter to its constructor rather than dict object. |

The open() function defined in shelve module which return a DbfilenameShelf object.

**open(**filename, flag='c', protocol=None, writeback=False)

Open a persistent dictionary. The filename specified is the base filename for the underlying database. As a side-effect, an extension may be added to the filename and more than one file may be created. By default, the underlying database file is opened for reading and writing. The optional flag parameter has the same interpretation as the *flag* parameter of [dbm.open().](https://docs.python.org/3/library/dbm.html#dbm.open)By default, version 3 pickles are used to serialize values. The version of the pickle protocol can be specified with the protocol parameter. writeback parameter by default is false. If set to true, the accessed entries are cached. Every access calls sync() and close() operations hence process may be slow. **Example:** import shelve s = shelve.open('test\_shelf') try:

s['key1'] = { 'int': 10, 'float':9.5, 'string':'Sample data' }

finally:

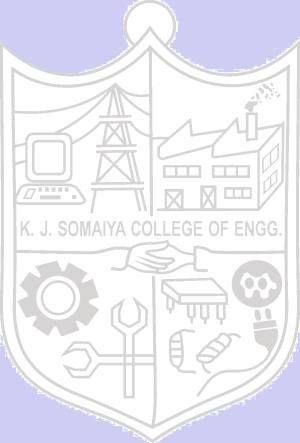
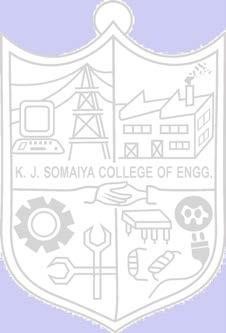
s.close()

The Shelf object has following methods available −

|  |  |
| --- | --- |
| **Sr.No.** | **Methods & *Description*** |
| 1 | **close()** synchronise and close persistent dict object. |
| 2 | **sync()**  Write back all entries in the cache if shelf was opened with writeback set to True. |
| 3 | **get()**  returns value associated with key |
| 4 | **items()**  list of tuples – each tuple is key value pair |

|  |  |
| --- | --- |
| 5 | **keys()**  list of shelf keys |
| 6 | **pop()** remove specified key and return the corresponding value. |
| 7 | **update()**  Update shelf from another dict/iterable |
| 8 | **values()**  list of shelf values |

The [pickle](https://docs.python.org/3/library/pickle.html#module-pickle) module implements binary protocols for serializing and de-serializing a Python object structure. “Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream (from a [binary file](https://docs.python.org/3/glossary.html#term-binary-file) or [bytes-like object](https://docs.python.org/3/glossary.html#term-bytes-like-object)) is converted back into an object hierarchy.



**Activities:**

1.

Use shelve module to create Student

database

to store details of

the students(

at least

10

student details)

2.

Perform read, update and

delete operations on the

Student

Database.

Result: (script and output)

Create a DbfilenameShelf object ,write values and implement various methods :

Code :

import shelve

student = shelve.open("StudentData")

#This will create StudentData.dir file in current directory and store key-value data in hashed form.

print("Type of the shelf object created:");

print(type(student));

#storing dictionary entries in the file.

student['name'] = "Riya"

student['age'] = 19

student['marks'] = 98

student['department'] = "IT"

#To access value of a particular key in shelf.

print(student['marks'])

#changing the marks of that student

student['marks']=30

#Using the get method which returns value associated with key

student.get('marks')

#Using the items method which returns a list of tuples – each tuple is key value pair

print(list(student.items()))

#Using the keys method which returns a list of shelf keys

print(list(student.keys()))

#Using the values method which returns a list of shelf values

print(list(student.values()))

#using pop method to remove a key-value pair from shelf

student.pop('name')

print(list(student.items()))

#using the update method to merge items of another dictionary with shelf

toadd = {'name':'Rita','division':"B"}

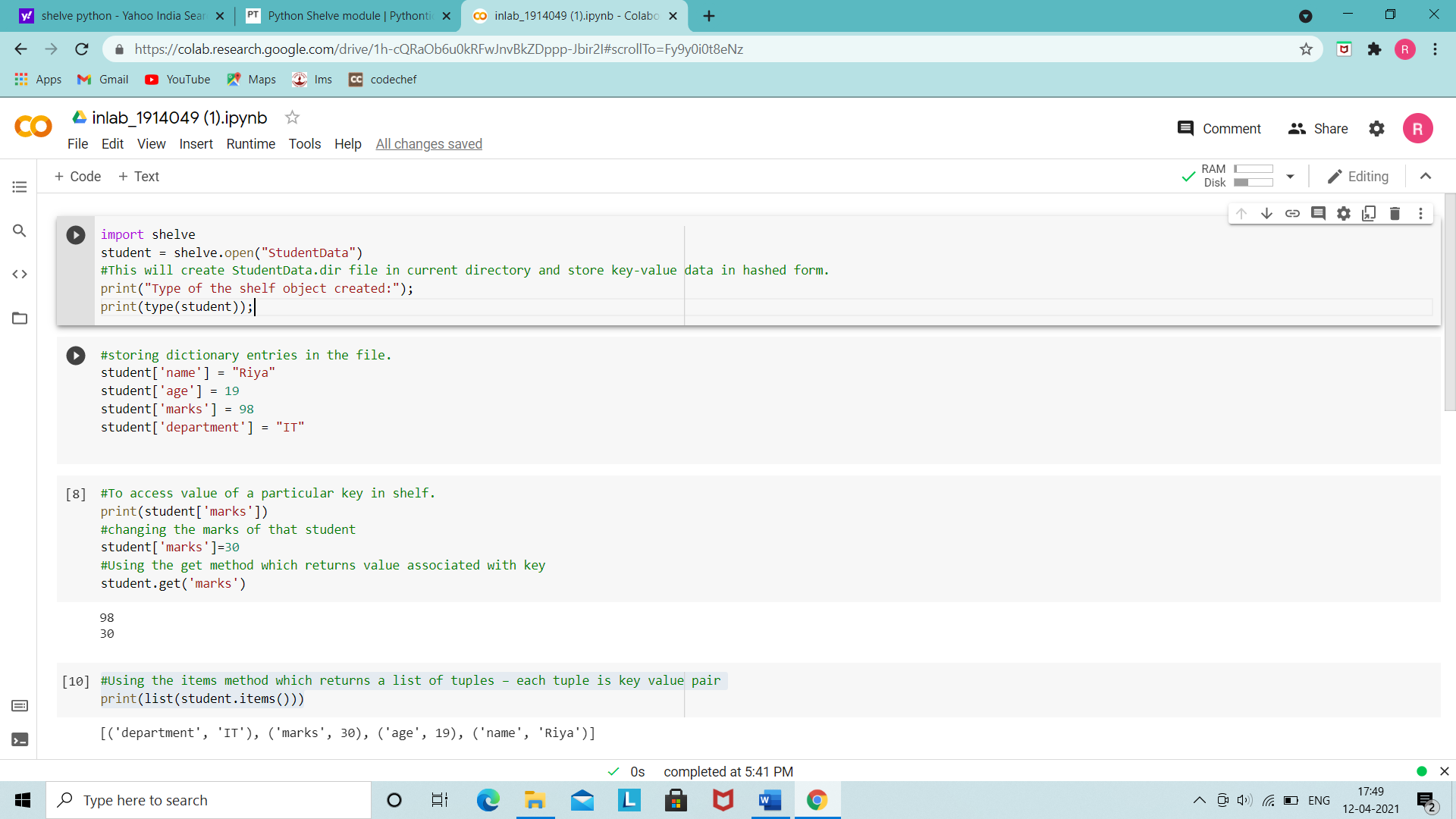
student.update(toadd)

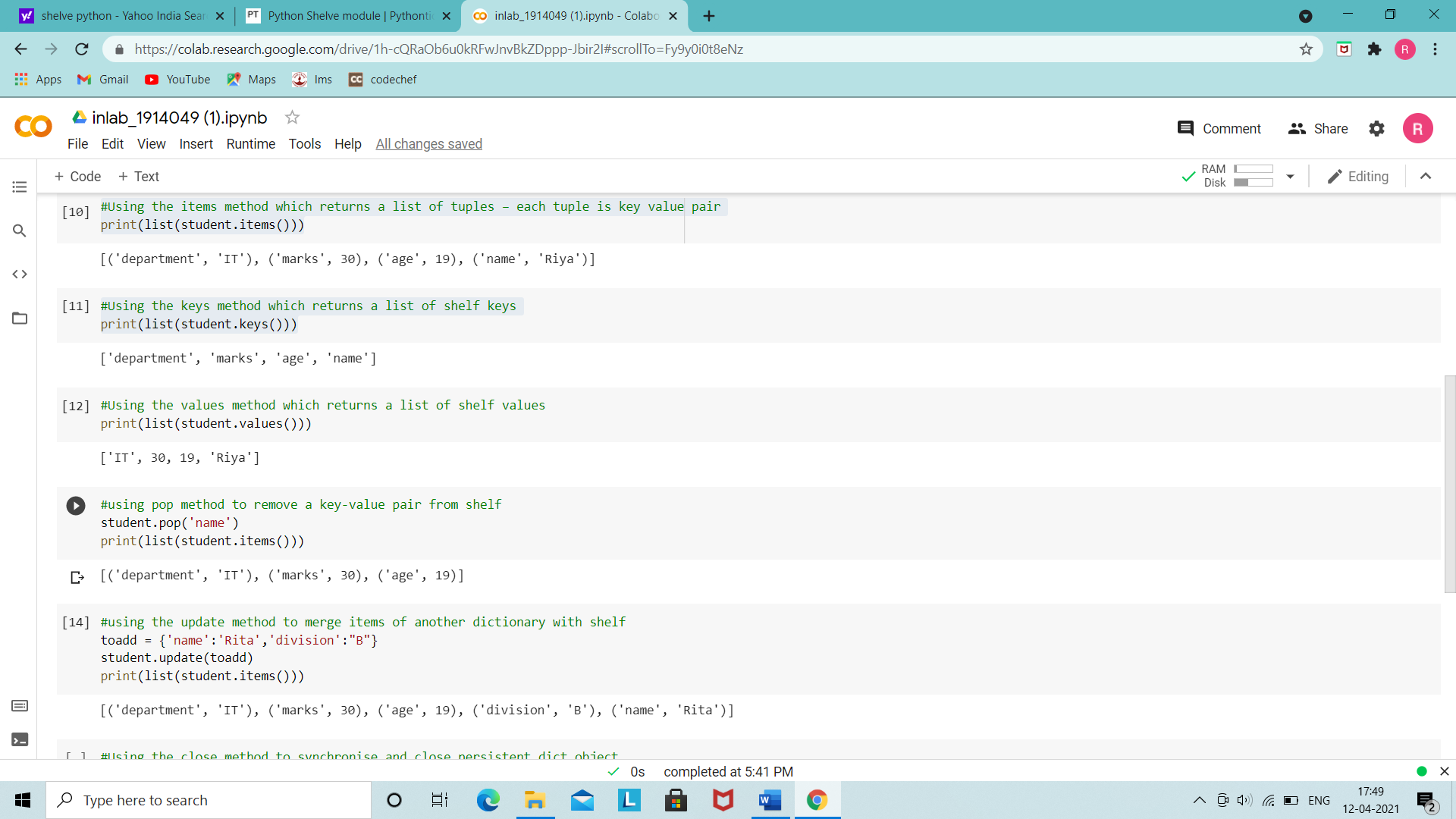
print(list(student.items()))

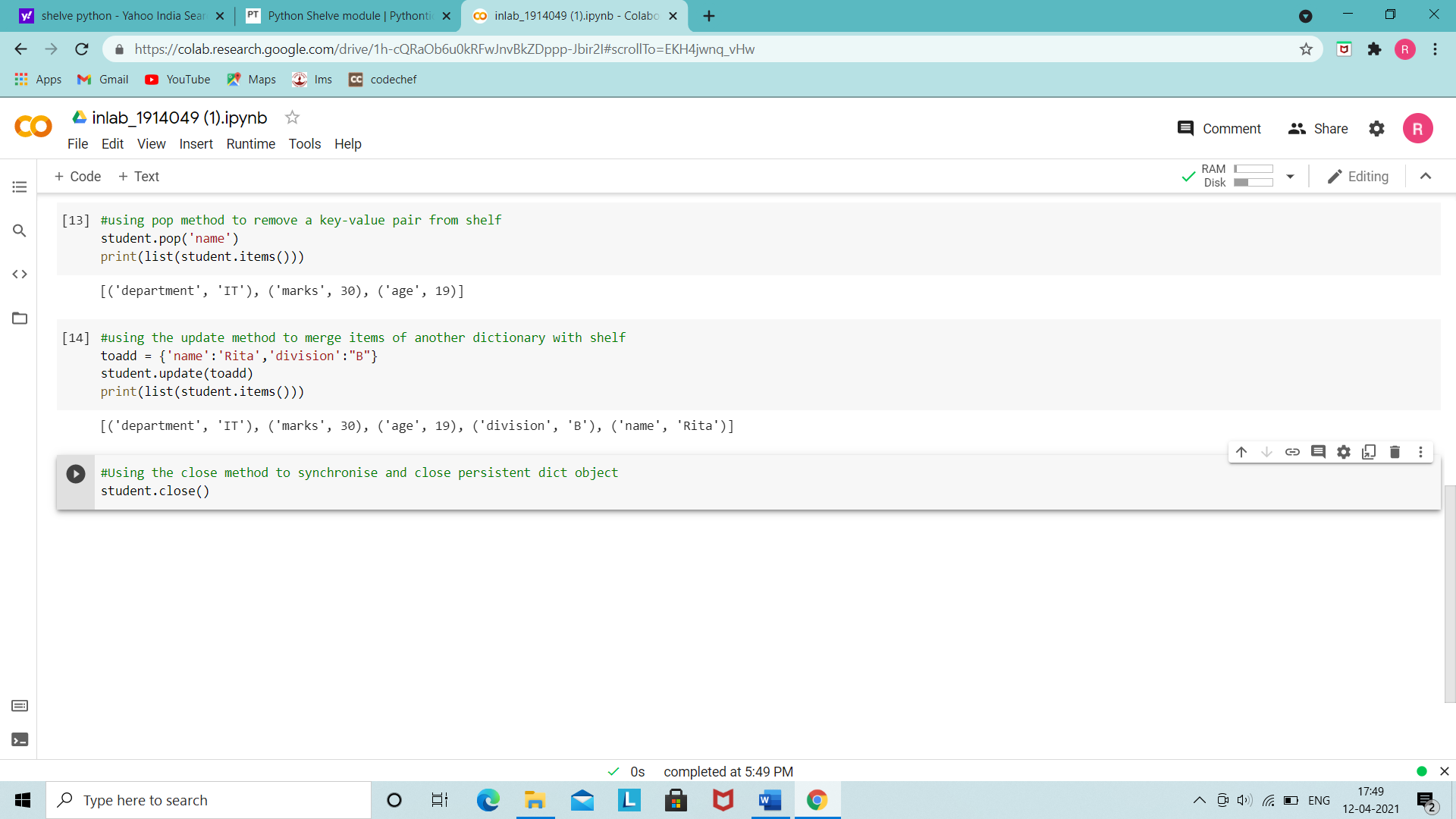
#Using the close method to synchronise and close persistent dict object

student.close()

OUTPUT:







Python program to create an in-memory shelf :

Code :

import shelve

import pickle

# Class definition

class Student:

    # Initialiser

    def \_\_init\_\_(self, id, name, department, division, marks):

        self.id = id;

        self.name = name;

        self.department = department;

        self.division = division;

        self.marks = marks;

    def \_\_repr\_\_(self):

        return "Student Info >>>Id:%d, Name:%s, Department:%s, Division:%s, Marks:%d"%(self.id, self.name, self.department, self.division, self.marks);

# Adding 10 students to our database

s1 = Student(1, "Riya", "IT", "B", 99);

s2 = Student(2, "Aditi", "IT", "B", 70);

s3 = Student(3, "Hardik", "IT", "A", 95);

s4 = Student(4, "Sparsh", "Comps", "B", 100);

s5 = Student(5, "Vidi", "Comps", "A", 89);

s6 = Student(6, "Sejal", "EXTC", "B", 60);

s7 = Student(7, "Rohan", "Mech", "A", 98);

s8 = Student(8, "Shilpa", "EXTC", "A", 80);

s9 = Student(9, "Harsh", "IT", "B", 79);

s10 = Student(10, "Viraj", "Mech", "B", 59);

d1 = {};

student = shelve.Shelf(d1);

# Python internally pickles the values

student["1"] = s1;

student["2"] = s2;

student["3"] = s3;

student["4"] = s4;

student["5"] = s5;

student["6"] = s6;

student["7"] = s7;

student["8"] = s8;

student["9"] = s9;

student["10"] = s10;

# Python internally unpickles the values

print(student["1"]);

print(student["2"]);

print(student["3"]);

print(student["4"]);

print(student["5"]);

print(student["6"]);

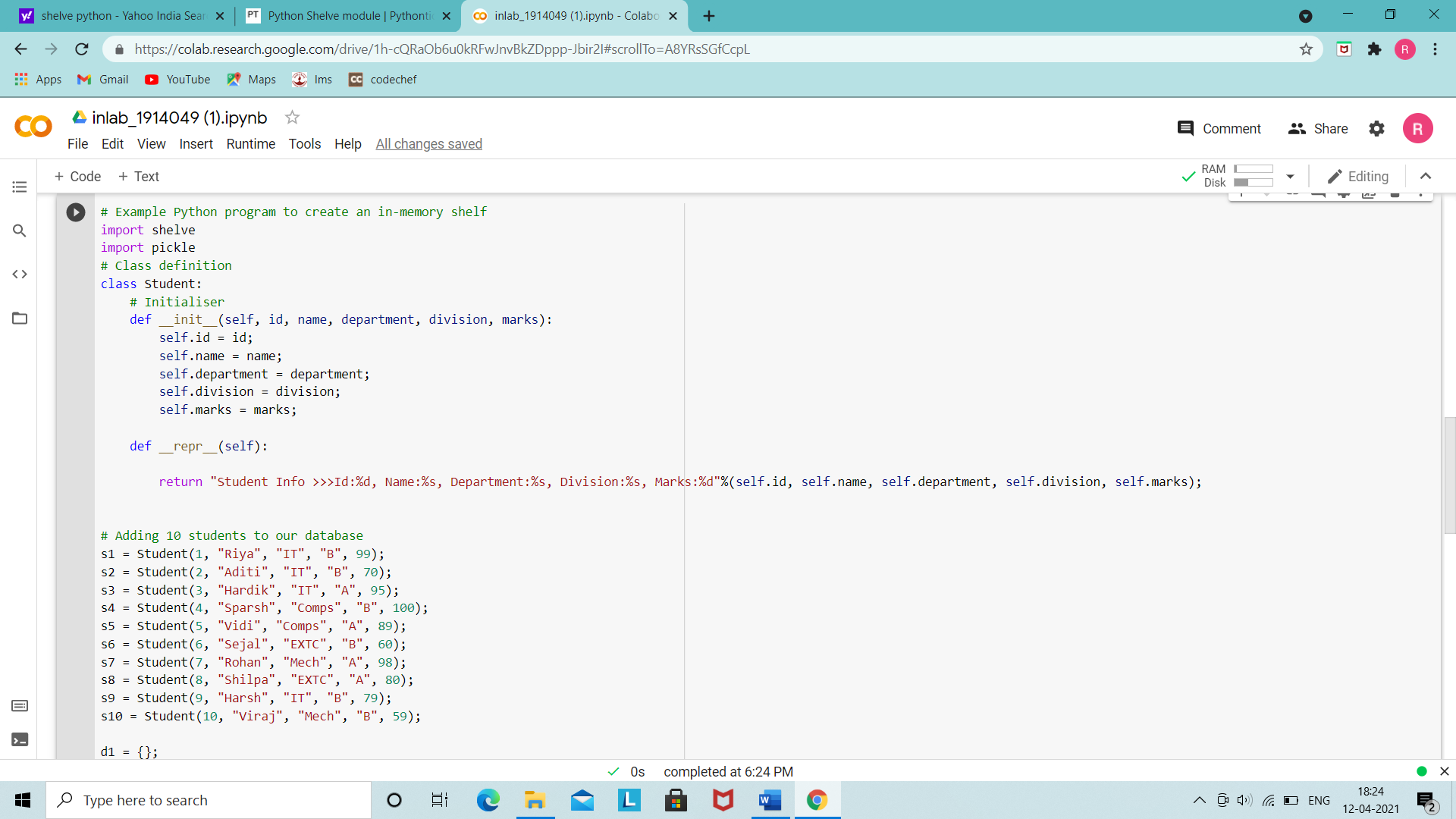
print(student["7"]);

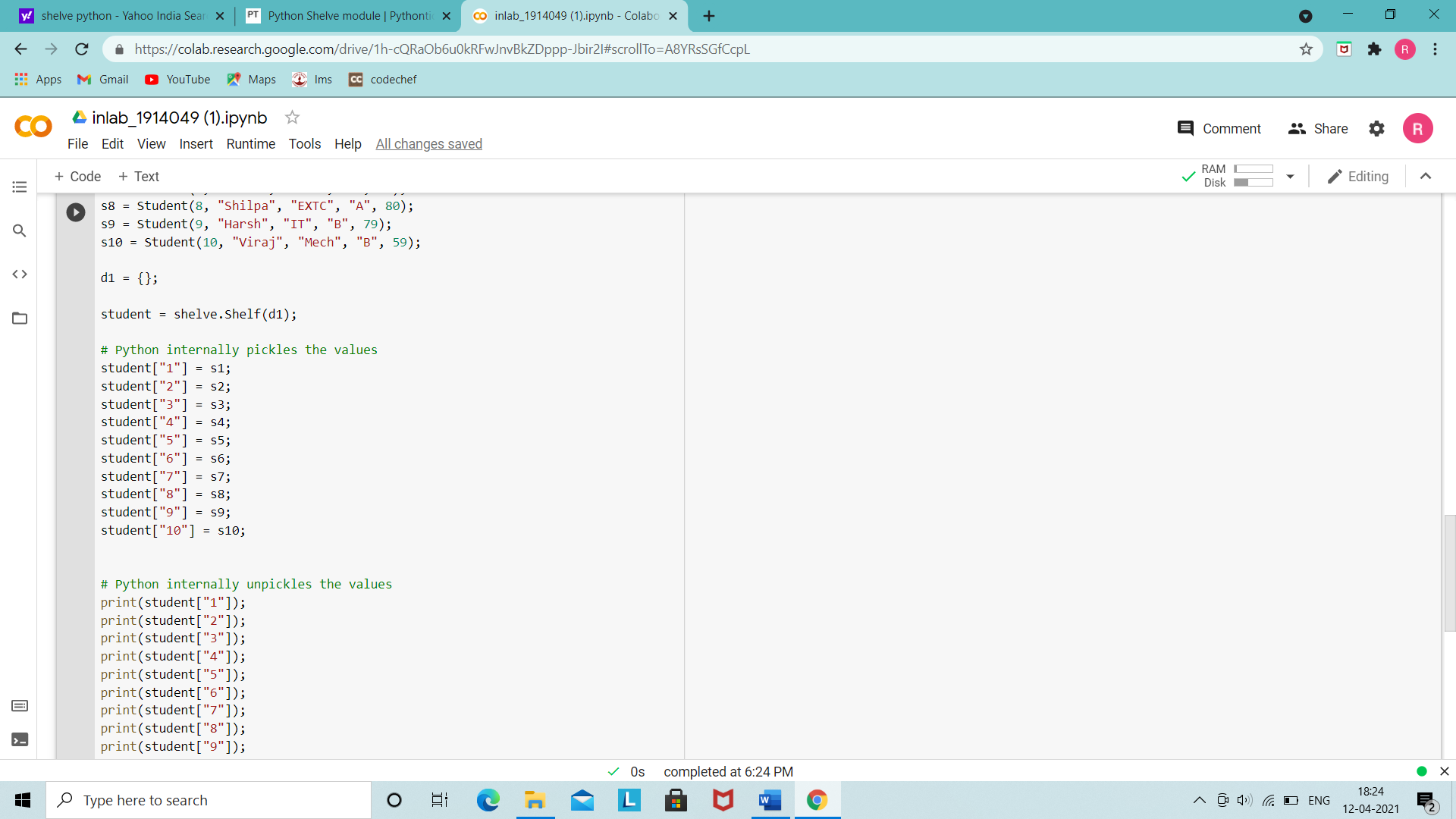
print(student["8"]);

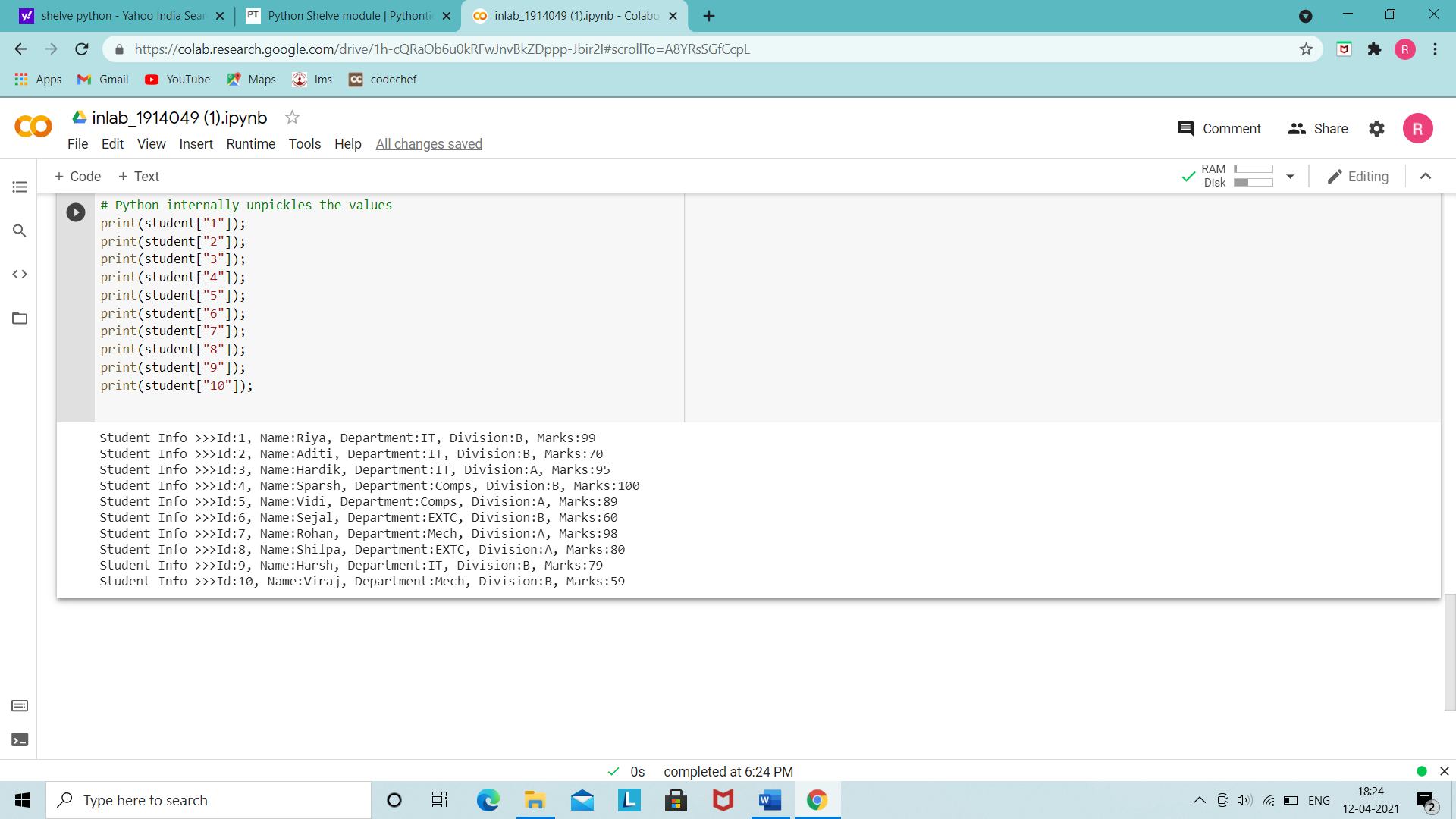
print(student["9"]);

print(student["10"]);

OUTPUT:







Outcomes:

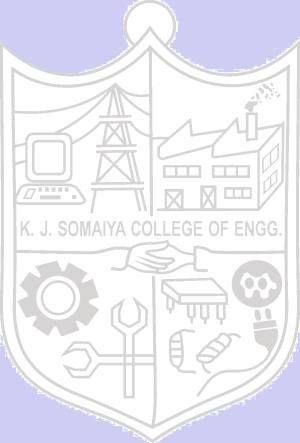
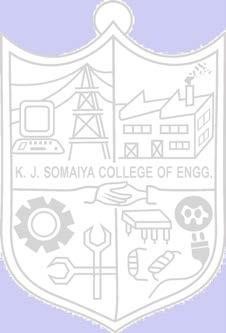
CO3 : Demonstrate handling database with python

**Conclusion:** (Conclusion to be based on the objectives and outcomes achieved)

In this experiment we learnt about ‘shelve’ module ,its methods and applications . We made a database ‘Student’ using the shelve module and implemented its method on the same.

**References:**

1. Yuxi Liu ; Python Machine Learning By Example: The easiest way to get into machine learning 1st Edition, Kindle Edition , Packt publishing Ltd , 1st edition 2017.
2. Martin C. Brown, The Complete Reference Paperback, Osborne, 2nd edition 2001
3. Frank Millstein, Data Analytics with Python: Data Analytics In Python Using Pandas, Copyright at Frank Millstein, 1st edition 2018



1. <https://docs.python.org/3/library/shelve.html>