**Practical No 4**

**PART 69**

**A.1 AIM: - To study and understand Tuple, Dictionaries in python**

**A.2 Prerequisite**

Programming for problem-solving and Object Oriented Programming

**A.3 Outcome**

After successful completion of this experiment, students will be able to understand and implement

1. Tuple
2. Dictionaries

**A.4 Theory**

A tuple in Python is similar to a [list](https://www.programiz.com/python-programming/list). The difference between the two is that we cannot change the elements of a tuple once it is assigned whereas we can change the elements of a list.

Creating a Tuple

A tuple is created by placing all the items (elements) inside parentheses (), separated by commas. The parentheses are optional, however, it is a good practice to use them.

A tuple can have any number of items and they may be of different types (integer, float, list, [string](https://www.programiz.com/python-programming/string), etc.).

# Different types of tuples

# Empty tuple

my\_tuple = ()

print(my\_tuple)

# Tuple having integers

my\_tuple = (1, 2, 3)

print(my\_tuple)

# tuple with mixed datatypes

my\_tuple = (1, "Hello", 3.4)

print(my\_tuple)

# nested tuple

my\_tuple = ("mouse", [8, 4, 6], (1, 2, 3))

print(my\_tuple)

[Run Code](https://www.programiz.com/python-programming/online-compiler)

Output

()

(1, 2, 3)

(1, 'Hello', 3.4)

('mouse', [8, 4, 6], (1, 2, 3))

In the above example, we have created different types of tuples and stored different data items inside them.

As mentioned earlier, we can also create tuples without using parentheses:

my\_tuple = 1, 2, 3

my\_tuple = 1, "Hello", 3.4

Create a Python Tuple With one Element

In Python, creating a tuple with one element is a bit tricky. Having one element within parentheses is not enough.

We will need a trailing comma to indicate that it is a tuple,

var1 = ("Hello") # string

var2 = ("Hello",) # tuple

We can use the type() function to know which class a variable or a value belongs to.

var1 = ("hello")

print(type(var1)) # <class 'str'>

# Creating a tuple having one element

var2 = ("hello",)

print(type(var2)) # <class 'tuple'>

# Parentheses is optional

var3 = "hello",

print(type(var3)) # <class 'tuple'>

[Run Code](https://www.programiz.com/python-programming/online-compiler)

Here,

* ("hello") is a string so type() returns str as class of var1 i.e. <class 'str'>
* ("hello",) and "hello", both are tuples so type() returns tuple as class of var1 i.e. <class 'tuple'>

Access Python Tuple Elements

Like a [list](https://www.programiz.com/python-programming/list), each element of a tuple is represented by index numbers (0, 1, ...) where the first element is at index 0.

We use the index number to access tuple elements. For example,

1. Indexing

We can use the index operator [] to access an item in a tuple, where the index starts from 0.

So, a tuple having 6 elements will have indices from 0 to 5. Trying to access an index outside of the tuple index range(6,7,... in this example) will raise an IndexError.

The index must be an integer, so we cannot use float or other types. This will result in TypeError.

Likewise, nested tuples are accessed using nested indexing, as shown in the example below.

# accessing tuple elements using indexing

letters = ("p", "r", "o", "g", "r", "a", "m", "i", "z")

print(letters[0]) # prints "p"

print(letters[5]) # prints "a"

[Run Code](https://www.programiz.com/python-programming/online-compiler)

In the above example,

* letters[0] - accesses the first element
* letters[5] - accesses the sixth element

2. Negative Indexing

Python allows negative indexing for its sequences.

The index of -1 refers to the last item, -2 to the second last item and so on. For example,

# accessing tuple elements using negative indexing

letters = ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')

print(letters[-1]) # prints 'z'

print(letters[-3]) # prints 'r'

[Run Code](https://www.programiz.com/python-programming/online-compiler)

In the above example,

* letters[-1] - accesses last element
* letters[-3] - accesses third last element

3. Slicing

We can access a range of items in a tuple by using the slicing operator colon :.

# accessing tuple elements using slicing

my\_tuple = ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')

# elements 2nd to 4th index

print(my\_tuple[1:4]) # prints ('r', 'o', 'g')

# elements beginning to 2nd

print(my\_tuple[:-7]) # prints ('p', 'r')

# elements 8th to end

print(my\_tuple[7:]) # prints ('i', 'z')

# elements beginning to end

print(my\_tuple[:]) # Prints ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')

[Run Code](https://www.programiz.com/python-programming/online-compiler)

Output

('r', 'o', 'g')

('p', 'r')

('i', 'z')

('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')

Here,

* my\_tuple[1:4] returns a tuple with elements from index 1 to index 3.
* my\_tuple[:-7] returns a tuple with elements from beginning to index 2.
* my\_tuple[7:] returns a tuple with elements from index 7 to the end.
* my\_tuple[:] returns all tuple items.

Note: When we slice lists, the start index is inclusive but the end index is exclusive.

Python Tuple Methods

In Python ,methods that add items or remove items are not available with tuple. Only the following two methods are available.

Some examples of Python tuple methods:

my\_tuple = ('a', 'p', 'p', 'l', 'e',)

print(my\_tuple.count('p')) # prints 2

print(my\_tuple.index('l')) # prints 3

[Run Code](https://www.programiz.com/python-programming/online-compiler)

Here,

* my\_tuple.count('p') - counts total number of 'p' in my\_tuple
* my\_tuple.index('l') - returns the first occurrence of 'l' in my\_tuple

Iterating through a Tuple in Python

We can use the [for loop](https://www.programiz.com/python-programming/for-loop) to iterate over the elements of a tuple. For example,

languages = ('Python', 'Swift', 'C++')

# iterating through the tuple

for language in languages:

print(language)

[Run Code](https://www.programiz.com/python-programming/online-compiler)

Output

Python

Swift

C++

Check if an Item Exists in the Python Tuple

We use the in keyword to check if an item exists in the tuple or not. For example,

languages = ('Python', 'Swift', 'C++')

print('C' in languages) # False

print('Python' in languages) # True

[Run Code](https://www.programiz.com/python-programming/online-compiler)

Here,

* 'C' is not present in languages, 'C' in languages evaluates to False.
* 'Python' is present in languages, 'Python' in languages evaluates to True.

Advantages of Tuple over List in Python

Since tuples are quite similar to lists, both of them are used in similar situations.

However, there are certain advantages of implementing a tuple over a list:

* We generally use tuples for heterogeneous (different) data types and lists for homogeneous (similar) data types.
* Since tuples are immutable, iterating through a tuple is faster than with a list. So there is a slight performance boost.
* Tuples that contain immutable elements can be used as a key for a dictionary. With lists, this is not possible.
* If you have data that doesn't change, implementing it as tuple will guarantee that it remains write-protected.

Python dictionary is an ordered collection (starting from Python 3.7) of items. It stores elements in key/value pairs. Here, keys are unique identifiers that are associated with each value.

Let's see an example,

If we want to store information about countries and their capitals, we can create a dictionary with country names as keys and capitals as values.

|  |  |
| --- | --- |
| Keys | Values |
| Nepal | Kathmandu |
| Italy | Rome |
| England | London |

Create a dictionary in Python

Here's how we can create a dictionary in Swift.

capital\_city = {"Nepal": "Kathmandu", "Italy": "Rome", "England": "London"}

print(capital\_city)

[Run Code](https://www.programiz.com/python-programming/online-compiler)

Output

{'Nepal': 'Kathmandu', 'Italy': 'Rome', 'England': 'London'}

In the above example, we have created a dictionary named capital\_city. Here,

1. Keys are "Nepal", "Italy", "England"
2. Values are "Kathmandu", "Rome", "London"

Note: Here, keys and values both are of string type. We can also have keys and values of different data types.

Example 1: Python Dictionary

# dictionary with keys and values of different data types

numbers = {1: "One", 2: "Two", 3: "Three"}

print(numbers)

[Run Code](https://www.programiz.com/python-programming/online-compiler)

Output

[3: "Three", 1: "One", 2: "Two"]

In the above example, we have created a dictionary named numbers. Here, keys are of integer type and values are of string type.

Add Elements to a Python Dictionary

We can add elements to a dictionary using the name of the dictionary with []. For example,

capital\_city = {"Nepal": "Kathmandu", "England": "London"}

print("Initial Dictionary: ",capital\_city)

capital\_city["Japan"] = "Tokyo"

print("Updated Dictionary: ",capital\_city)

[Run Code](https://www.programiz.com/python-programming/online-compiler)

Output

Initial Dictionary: {'Nepal': 'Kathmandu', 'England': 'London'}

Updated Dictionary: {'Nepal': 'Kathmandu', 'England': 'London', 'Japan': 'Tokyo'}

In the above example, we have created a dictionary named capital\_city. Notice the line,

capital\_city["Japan"] = "Tokyo"

Here, we have added a new element to capital\_city with key: Japan and value: Tokyo.

Change Value of Dictionary

We can also use [] to change the value associated with a particular key. For example,

student\_id = {111: "Eric", 112: "Kyle", 113: "Butters"}

print("Initial Dictionary: ", student\_id)

student\_id[112] = "Stan"

print("Updated Dictionary: ", student\_id)

[Run Code](https://www.programiz.com/python-programming/online-compiler)

Output

Initial Dictionary: {111: 'Eric', 112: 'Kyle', 113: 'Butters'}

Updated Dictionary: {111: 'Eric', 112: 'Stan', 113: 'Butters'}

In the above example, we have created a dictionary named student\_id. Initially, the value associated with the key 112 is "Kyle". Now, notice the line,

student\_id[112] = "Stan"

Here, we have changed the value associated with the key 112 to "Stan".

Accessing Elements from Dictionary

In Python, we use the keys to access their corresponding values. For example,

student\_id = {111: "Eric", 112: "Kyle", 113: "Butters"}

print(student\_id[111]) # prints Eric

print(student\_id[113]) # prints Butters

[Run Code](https://www.programiz.com/python-programming/online-compiler)

Here, we have used the keys to access their corresponding values.

If we try to access the value of a key that doesn't exist, we'll get an error. For example,

student\_id = {111: "Eric", 112: "Kyle", 113: "Butters"}

print(student\_id[211])

# Output: KeyError: 211

[Run Code](https://www.programiz.com/python-programming/online-compiler)

Removing elements from Dictionary

We use the del statement to remove an element from the dictionary. For example,

student\_id = {111: "Eric", 112: "Kyle", 113: "Butters"}

print("Initial Dictionary: ", student\_id)

del student\_id[111]

print("Updated Dictionary ", student\_id)

[Run Code](https://www.programiz.com/python-programming/online-compiler)

Output

Initial Dictionary: {111: 'Eric', 112: 'Kyle', 113: 'Butters'}

Updated Dictionary {112: 'Kyle', 113: 'Butters'}

Here, we have created a dictionary named student\_id. Notice the code,

del student\_id[111]

The del statement removes the element associated with the key 111.

We can also delete the whole dictionary using the del statement,

student\_id = {111: "Eric", 112: "Kyle", 113: "Butters"}

# delete student\_id dictionary

del student\_id

print(student\_id)

# Output: NameError: name 'student\_id' is not defined

[Run Code](https://www.programiz.com/python-programming/online-compiler)

We are getting an error message because we have deleted the student\_id dictionary and student\_id doesn't exist anymore.

Python Dictionary Methods

Methods that are available with a dictionary are tabulated below. Some of them have already been used in the above examples.

|  |  |
| --- | --- |
| Function | Description |
| len() | Return the length (the number of items) in the dictionary.  languages = ['Python', 'Java', 'JavaScript']  # compute the length of languages  length = len(languages)  print(length)  # Output: 3 |
| sorted() | Return a new sorted list of keys in the dictionary.  numbers = [4, 2, 12, 8]  sorted\_numbers = sorted(numbers)  print(sorted\_numbers)  # Output: [2, 4, 8, 12] |
| clear() | Removes all items from the dictionary.  # dictionary  numbers = {1: "one", 2: "two"}  # removes all the items from the dictionary  numbers.clear()  print(numbers)  # Output: {} |

Dictionary Membership Test

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

# Membership Test for Dictionary Keys

squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

# Output: True

print(1 in squares) # prints True

print(2 not in squares) # prints True

# membership tests for key only not value

print(49 in squares) # prints false

[Run Code](https://www.programiz.com/python-programming/online-compiler)

Output

True

True

False

Iterating Through a Dictionary

We can iterate through each key in a dictionary using a for loop.

# Iterating through a Dictionary

squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

for i in squares:

print(squares[i])

[Run Code](https://www.programiz.com/python-programming/online-compiler)

Output

1

9

25

49

81

Here, we have iterated through each key in the squares dictionary using the for loop.

**Tasks:**

1.Create a tuple for storing student information and perform the following functions:

1. Print an item of the tuple
2. Check if an element exists within a tuple
3. Check number of times an item has repeated
4. Remove an item from a tuple
5. Slice a tuple
6. Get the index of an item of the tuple
7. Print the size of a tuple
8. Modify items of a tuple

2.Develop a menu driven program for a food hub.

Example:

Please select option:

1- Veg Menu

2- Non-Veg Menu

0- Exit

Enter your choice: 1

You have selected option 1 – Veg

Select from the following options for Veg

1- Veg Burger

2- Veg Pizza

3- Dosa

Enter your choice: 2

Do you wish to continue [Y/N] ? Y

Select from the following options for Veg

1- Veg Burger

2- Veg Pizza

3- Dosa

Enter your choice: 1

Do you wish to continue [Y/N] ? N

Your order is : [“Veg Pizza”, “Veg Burger”]

Perform operations accordingly.

1. For the same above example include price. The output will look like

{“Veg Pizza” : 120, “Veg Burger”: 110}. Also print the total price of user choice items.

3. Write a Python script to concatenate following dictionaries to create a new one

Sample Dictionary :

dic1={1:10, 2:20}

dic2={3:30, 4:40}

dic3={5:50,6:60}

Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

4. Write a Python script to print a dictionary where the keys are numbers between 1 and 15 (both included) and the values are square of keys.

Sample Dictionary

{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121, 12: 144, 13: 169, 14: 196, 15: 225}

5.Write a Python program to print all unique values in a dictionary

Sample Data : [{"V":"S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"}, {"VII":"S005"}, {"V":"S009"},{"VIII":"S007"}]

Expected Output : Unique Values: {'S005', 'S002', 'S007', 'S001', 'S009'}

6.Write a Python program to combine values in python list of dictionaries.

Sample data: [{'item': 'item1', 'amount': 400}, {'item': 'item2', 'amount': 300}, {'item': 'item1', 'amount': 750}]

Expected Output: Counter({'item1': 1150, 'item2': 300})

7. Write a Python program to get the top three items in a shop.

Sample data: {'item1': 45.50, 'item2':35, 'item3': 41.30, 'item4':55, 'item5': 24}

Expected Output:

item4 55

item1 45.5

item3 41.3

8. Write a Python program to filter the height and width of students, which are stored in a dictionary

Original Dictionary:

{'Cierra Vega': (6.2, 70), 'Alden Cantrell': (5.9, 65), 'Kierra Gentry': (6.0, 68), 'Pierre Cox': (5.8, 66)}

Height > 6ft and Weight> 70kg:

{'Cierra Vega': (6.2, 70)}

9.Write a Python program to extract a list of values from a given list of dictionaries.

Original Dictionary:

[{'Math': 90, 'Science': 92}, {'Math': 89, 'Science': 94}, {'Math': 92, 'Science': 88}]

Extract a list of values from said list of dictionaries where subject = Science

[92, 94, 88]

Original Dictionary:

[{'Math': 90, 'Science': 92}, {'Math': 89, 'Science': 94}, {'Math': 92, 'Science': 88}]

Extract a list of values from said list of dictionaries where subject = Math

[90, 89, 92]

10.Write a Python program to print a tuple with string formatting.

Sample tuple : (100, 200, 300)

Output : This is a tuple (100, 200, 300)

11.Write a Python program to sort a tuple by its float element.

Sample data: [('item1', '12.20'), ('item2', '15.10'), ('item3', '24.5')]

Expected Output: [('item3', '24.5'), ('item2', '15.10'), ('item1', '12.20')]

12.Write a Python program calculate the product, multiplying all the numbers of a given tuple.

Original Tuple:

(4, 3, 2, 2, -1, 18)

Product - multiplying all the numbers of the said tuple: -864

Original Tuple:

(2, 4, 8, 8, 3, 2, 9)

Product - multiplying all the numbers of the said tuple: 27648

13.Write a Python program to calculate the average value of the numbers in a given tuple of tuples.

Original Tuple:

((10, 10, 10, 12), (30, 45, 56, 45), (81, 80, 39, 32), (1, 2, 3, 4))

Average value of the numbers of the said tuple of tuples:

[30.5, 34.25, 27.0, 23.25]

Original Tuple:

((1, 1, -5), (30, -15, 56), (81, -60, -39), (-10, 2, 3))

Average value of the numbers of the said tuple of tuples:

[25.5, -18.0, 3.75]

14.Write a Python program to check if a specified element presents in a tuple of tuples.

Original list:

(('Red', 'White', 'Blue'), ('Green', 'Pink', 'Purple'), ('Orange', 'Yellow', 'Lime'))

Check if White present in said tuple of tuples!

True

Check if White present in said tuple of tuples!

True

Check if Olive present in said tuple of tuples!

False

15.Write a Python program to convert a given list of tuples to a list of lists.

Original list of tuples: [(1, 2), (2, 3), (3, 4)]

Convert the said list of tuples to a list of lists: [[1, 2], [2, 3], [3, 4]]

Original list of tuples: [(1, 2), (2, 3, 5), (3, 4), (2, 3, 4, 2)]

Convert the said list of tuples to a list of lists: [[1, 2], [2, 3, 5], [3, 4], [2, 3, 4, 2]]

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

**(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Teams or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)**

|  |  |
| --- | --- |
| Roll No.C158 | Name: Mannan Sheth |
| Program:BTI CS | Division: D |
| Semester: 4 | Batch : D2 |
| Date of Experiment: 17-1-24 | Date of Submission: |
| Grade : |  |

B.1 Software Code written by student:

***(Paste your Python code completed during the 2 hours of practical in the lab here)***

**1.**

studentInfo = ("Mannan", "Aryan", "Rehaan", "Aryan", "Uday", "Umang", "Ishan", "Sohaan")

print(studentInfo[0])

print("Mannan" in studentInfo)

print(studentInfo.count("Aryan"))

listInfo = list(studentInfo)

listInfo.pop(1)

studentInfo = tuple(listInfo)

print(studentInfo)

print(studentInfo[:4])

print(studentInfo.index("Mannan"))

print(len(studentInfo))

listInfo[1] = "NA"

studentInfo = tuple(listInfo)

print(studentInfo)

**2.**

vegOptions = {1: {"Veg Burger": "120"},

              2: {"Veg pizza": "150"},

              3: {"Dosa": "50"}}

nonvegOptions = {1: {"non-Veg Burger": "120"},

              2: {"non-Veg pizza": "150"},

              3: {"Fish": "100"}}

order = []

while True:

    print("""Please select option:

                1-Veg menu

                2-Non-veg menu

                0-Exit""")

    choice = int(input("Enter your choice: "))

    if choice == 0:

        print("Exiting.")

        print(f"Your order is {order}")

        break

    if choice == 1:

        print("You have selected option 1 - Veg")

        print(f"""Select from the following:

            {vegOptions}""")

        choiceOfFood = int(input("Enter your choice: "))

        for i in vegOptions.keys():

            if i == choiceOfFood:

                order.append(vegOptions[i])

        while True:

            cont = input("Do you wish to continue [Y/N]: ")

            if cont == "Y":

                print(f"""Select from the following:

                {vegOptions}""")

                choiceOfFood = int(input("Enter your choice: "))

                for i in vegOptions.keys():

                    if i == choiceOfFood:

                        order.append(vegOptions[i])

            if cont == "N":

                print(f"Your order is {order}")

                break

        break

    if choice == 2:

        print("You have selected option 2- Non veg")

        print(f"""Select from the following:

            {nonvegOptions}""")

        choiceOfFood = int(input("Enter your choice: "))

        for i in nonvegOptions.keys():

            if i == choiceOfFood:

                order.append(nonvegOptions[i])

        while True:

            cont = input("Do you wish to continue [Y/N]: ")

            if cont == "Y":

                print(f"""Select from the following:

                {nonvegOptions}""")

                choiceOfFood = int(input("Enter your choice: "))

                for i in nonvegOptions.keys():

                    if i == choiceOfFood:

                        order.append(nonvegOptions[i])

            if cont == "N":

                print(f"Your order is {order}")

                break

        break

**3.**

dic1 = {1:10, 2:20}

dic2 = {3:30, 4:40}

dic3 = {5:50, 6:60}

dic1.update(dic2)

dic1.update(dic3)

print(dic1)

**4.**

d = {}

for i in range(1, 16):

    d[i] = i\*\*2

print(d)

**5.**

data = [{"V":"S001"}, {"V":"S002"}, {"VI":"S001"}, {"VI":"S005"}, {"VII":"S005"}, {"V":"S009"}, {"VIII":"S007"}]

values = []

for i in data:

    for j in i.keys():

        values.append(i[j])

print(set(values))

**6.**

data = [{"item": "item1", "amount": 400}, {"item": "item2", "amount": 300}, {"item": "item1", "amount": 750}]

newdata = {}

for i in data:

    item = i["item"]

    amount = i["amount"]

    if item in newdata:

        newdata[item] += amount

    else:

        newdata[item] = amount

print(newdata)

**7.**

data = {'item1': 45.50, 'item2':35, 'item3': 41.30, 'item4':55, 'item5': 24}

result = {}

values = list(data.values())

values.sort(reverse=True)

values = values[:3]

for i in values:

   for j in data.keys():

      if data[j] == i:

         result.update({j:i})

print(result)

**8.**

data = {'Cierra Vega': (6.2, 70), 'Alden Cantrell': (5.9, 65), 'Kierra Gentry': (6.0, 68), 'Pierre Cox': (5.8, 66)}

values = list(data.values())

print(values)

for i in values:

   if i[0] > 6 and i[1] >= 70:

      for j in data.keys():

         if data[j] == i:

            result = {j: i}

            print(result)

**9.**

data = [{'Math': 90, 'Science': 92}, {'Math': 89, 'Science': 94}, {'Math': 92, 'Science': 88}]

scienceMarks = []

for i in data:

   if "Science" in i.keys():

      scienceMarks.append(i["Science"])

print(f"Science Marks: {scienceMarks}")

mathMarks = []

for i in data:

   if "Math" in i.keys():

      mathMarks.append(i["Math"])

print(f"Math Marks: {mathMarks}")

**10.**

t = (100, 200, 300)

print("This is a tuple {}".format(t))

**11.**

data = [('item1', '12.20'), ('item2', '15.10'), ('item3', '24.5')]

sortedData = []

floats=[]

for i in data:

   floats.append(float(i[1]))

floats.sort(reverse=True)

for j in floats:

   for i in data:

      if float(i[1]) == j:

         sortedData.append(i)

print(sortedData)

**12.**

t = []

for i in range(int(input("Enter number of elements of tuple: "))):

   t.append(int(input()))

t = tuple(t)

product = 1

for i in t:

   product \*= i

print(f"Multiplying all numbers of said tuple: {product}")

**13.**

data = ((10, 10, 10, 12), (30, 45, 56, 45), (81, 80, 39, 32), (1, 2, 3, 4))

averages=[]

for j in range(len(data[0])):

   sums=0

   for i in data:

      sums += i[j]

   averages.append(sums/len(data))

print(averages)

**14.**

data = (('Red', 'White', 'Blue'), ('Green', 'Pink', 'Purple'), ('Orange', 'Yellow', 'Lime'))

check = input("Enter color to check: ")

flag = 0

for i in data:

   for j in i:

      if j == check:

         flag = 1

         break

if (flag):

   print(True)

else:

   print(False)

**15.**

data = [(1, 2), (2, 3), (3, 4)]

for i in range(len(data)):

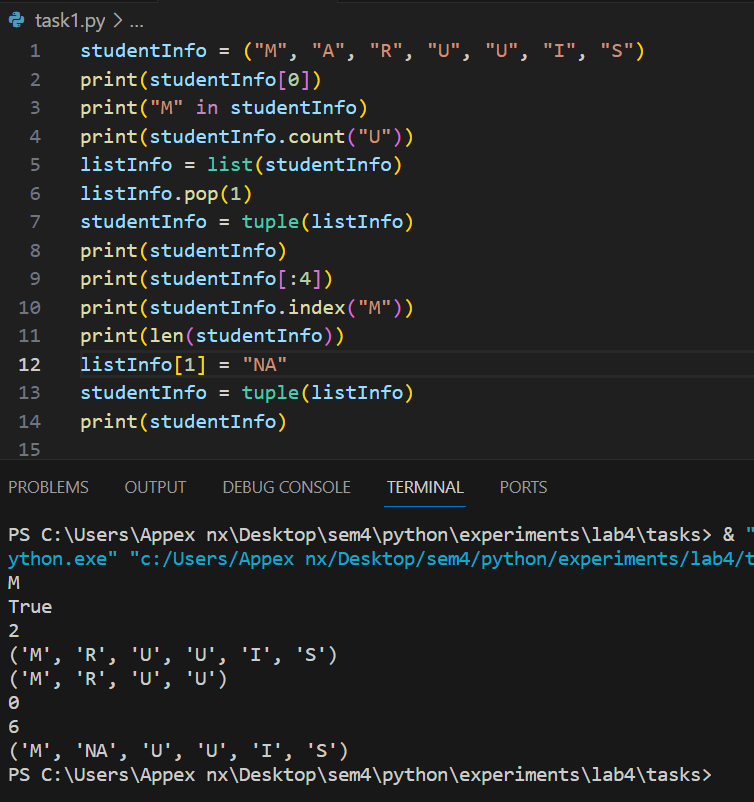
   data[i] = list(data[i])

print(data)

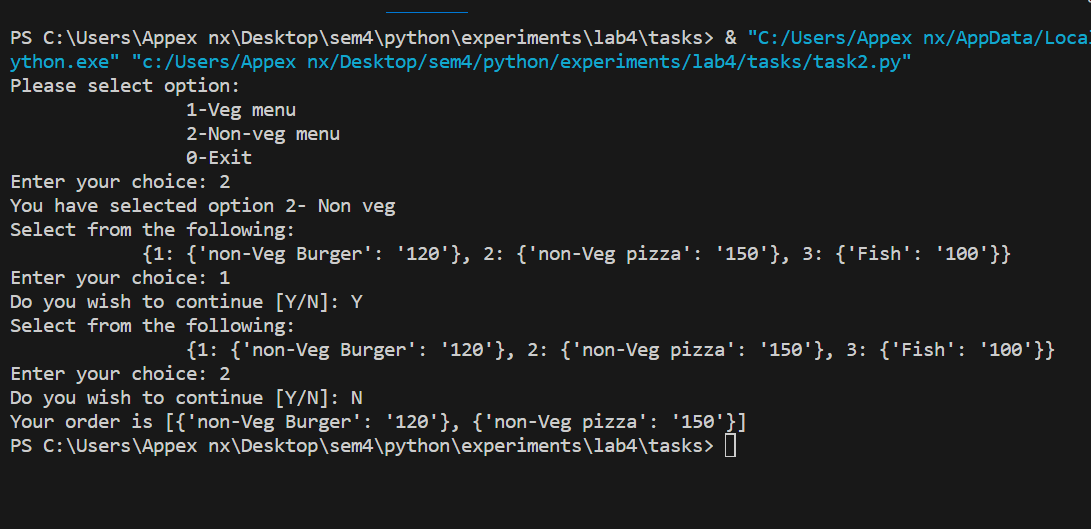
B.2 Input and Output:

***(Paste your program input and output in following format. If there is error then paste the specific error in the output part. In case of error with due permission of the faculty extension can be given to submit the error free code with output in due course of time. Students will be graded accordingly.)***

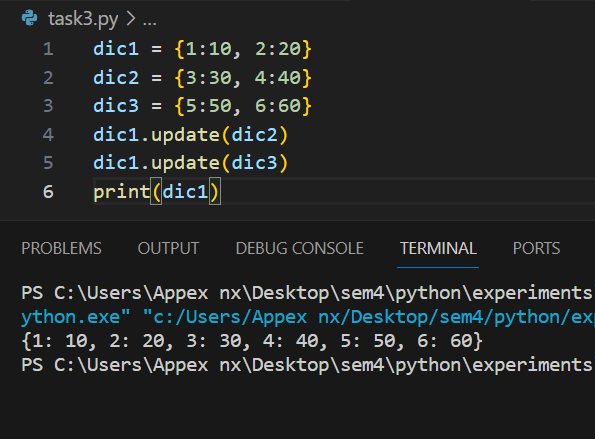
**1.**

****

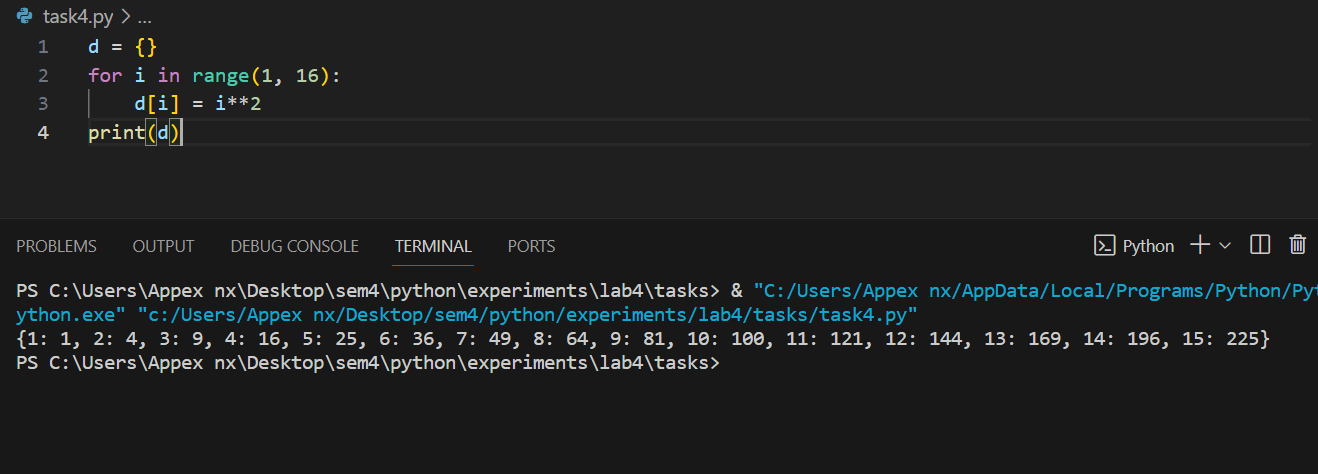
**2.**

****

**3.**

****

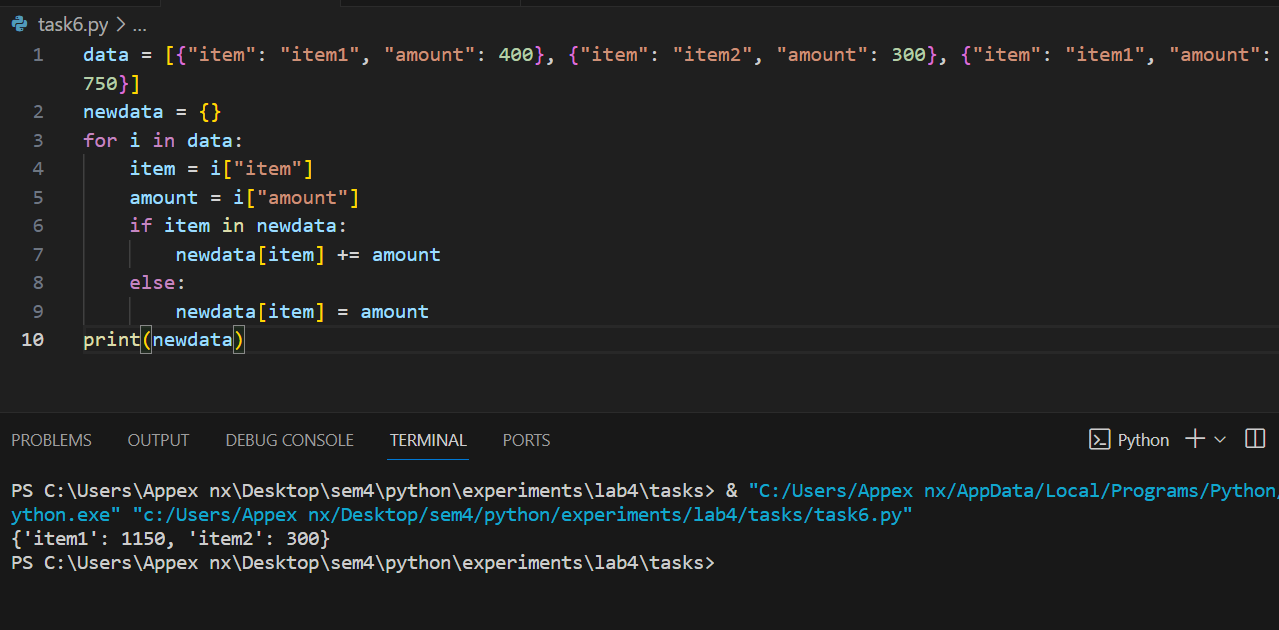
**4.**

****

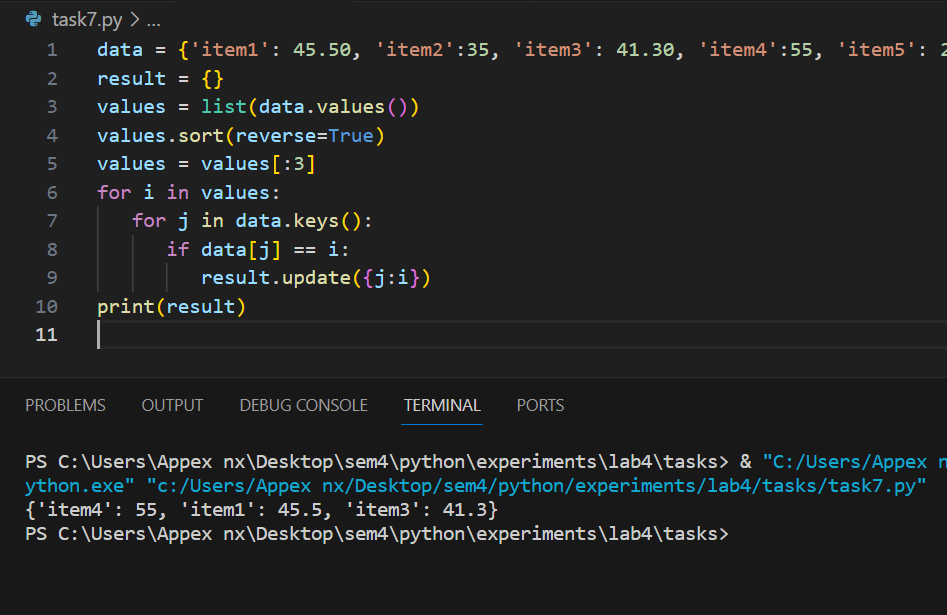
**5.**

****

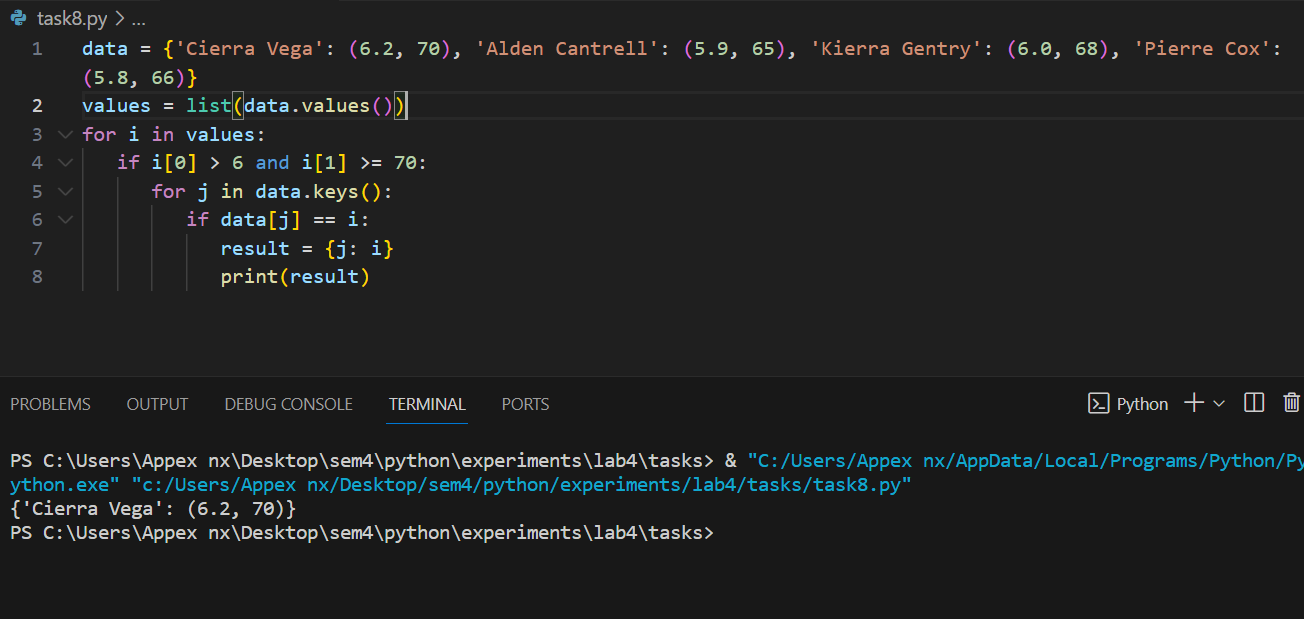
**6.**

****

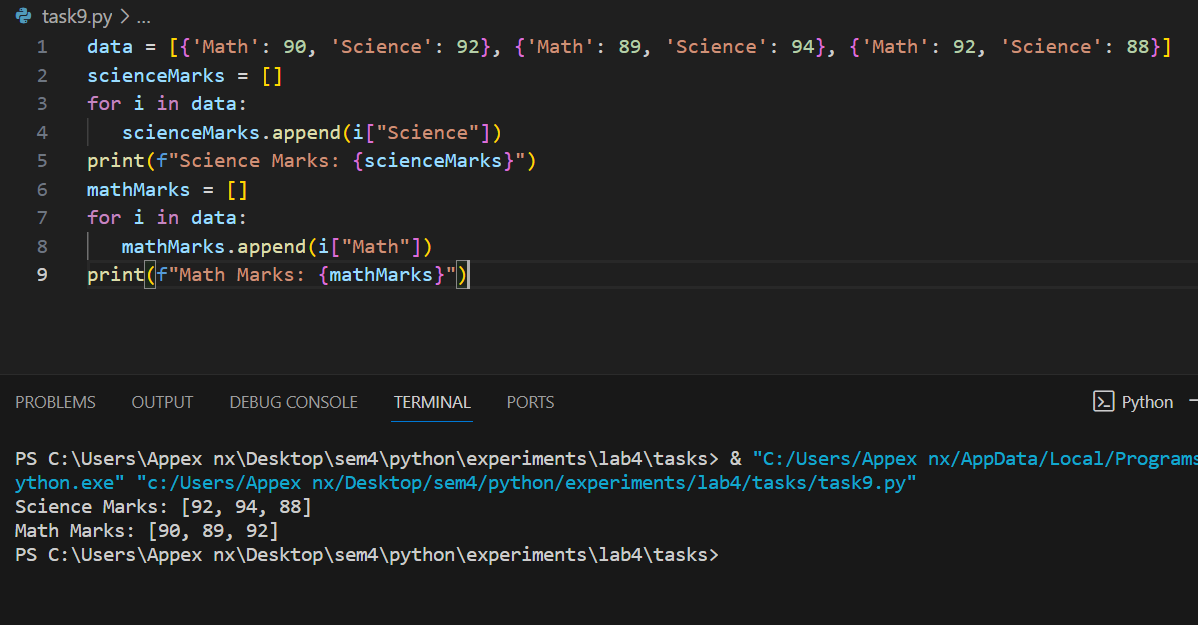
**7.**

****

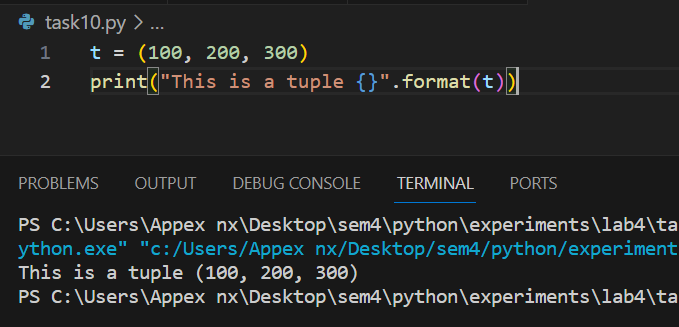
**8.**

****

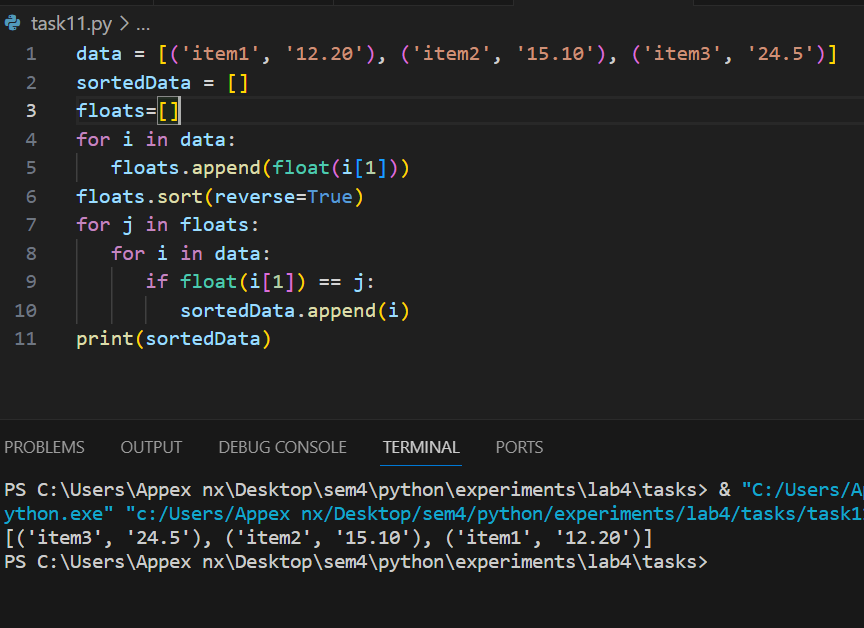
**9.**

****

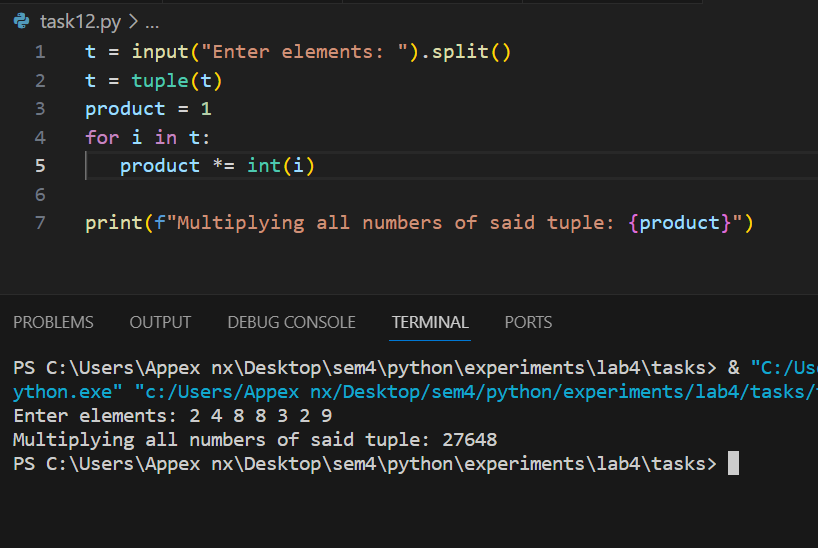
**10.**

****

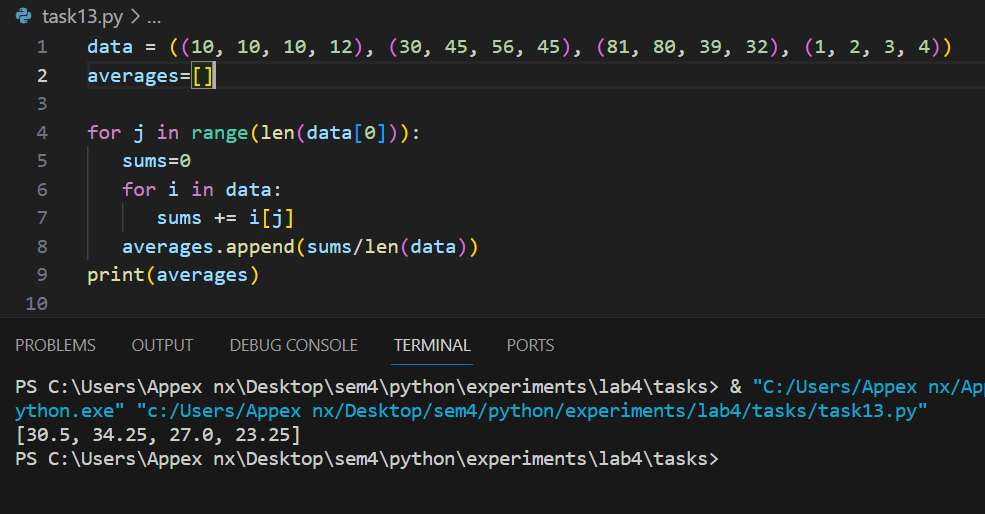
**11.**

****

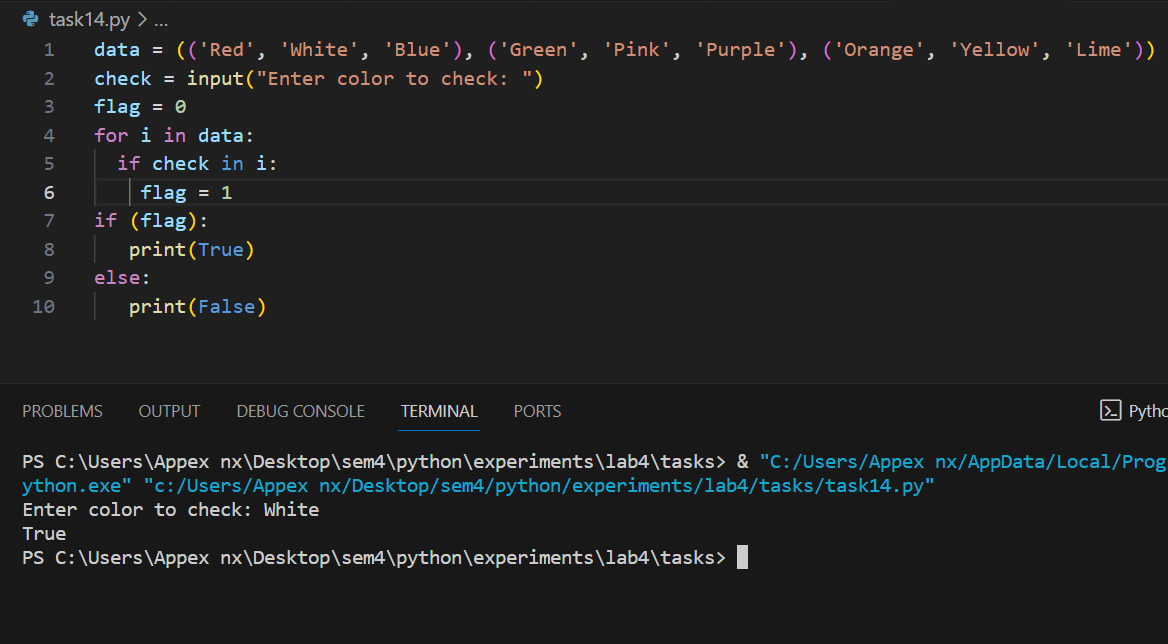
**12.**

****

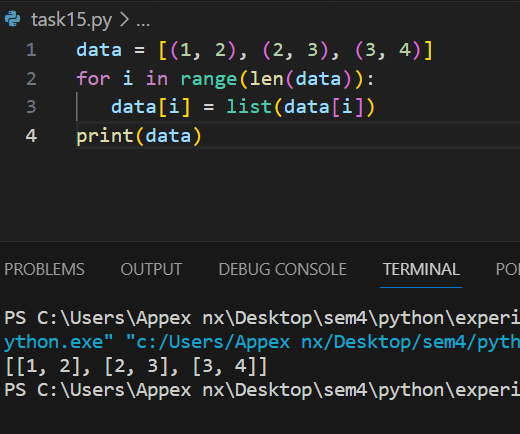
**13.**

****

**14.**

****

**15.**

****

B.3 Conclusion:

*(****Students must write the conclusion as per the attainment of individual outcome listed above and learning/observation noted in section B.1)***

After successful completion of this experiment, I am able to understand and implement Tuples and Dictionaries