A logo with blue and red text

Description automatically generated A logo with text on it

Description automatically generated

**Introduction to Programming**

**Lab Worksheet**

**Week 6**

**Python**

Name: Shivnath Mahato

Level 4 Section: A

British Id: 10012

LBU Id: c7576124

Level 4 BSc. Hons Computing

Subject: Fundamental of Computer Programming (FOCP)

The British College (TBC)

**Task:**

**#Week 6 Practical**

1. Write a for...in loop that iterates over all the elements of the squares list and prints the square root of each to the screen. *Hint*: you may want to import a function from the math module to help achieve this.

squares = [4, 9, 16, 25]

**Answer:**

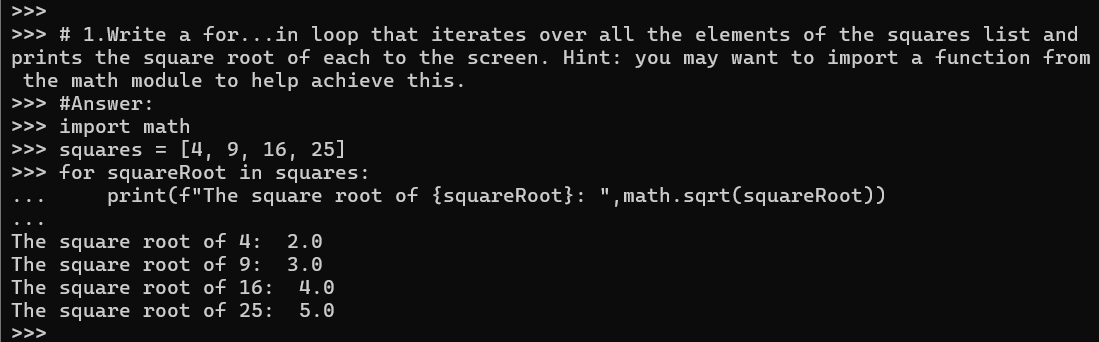
import math

squares = [4, 9, 16, 25]

for squareRoot in squares:

    print(f"The square root of {squareRoot}: ",math.sqrt(squareRoot))

**Output of Question No. 1:**



1. Write some code that uses the append() method to add the next three square values (49, 64, 81) to the end of the squares list.

**Answer:**

squares = [4, 9, 16, 25]

squares.append(36)

print(squares)

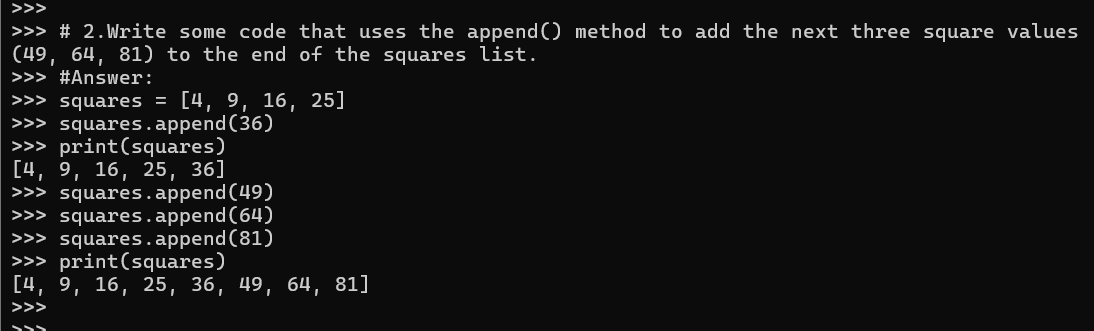
squares.append(49)

squares.append(64)

squares.append(81)

print(squares**)**

**Output of Question No. 2:**



1. Write some code that uses the extend() method to add the next three square values, starting at 121 (11 x 11), to the end of the squares list.

**Answer:**

squares = [4, 9, 16, 25, 36, 49, 64, 81]

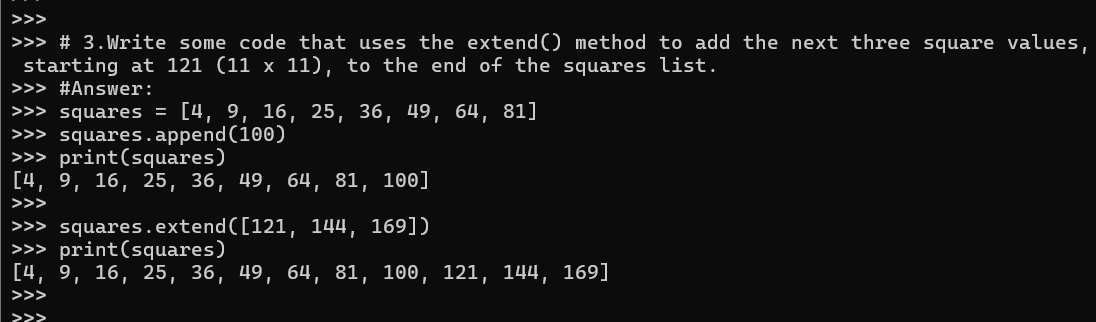
squares.append(100)

print(squares)

squares.extend([121, 144, 169])

print(squares)

**Output of Question No. 3:**



1. Write some code that uses the insert() method to insert the value 2, to the very beginning of the squares list.

**Answer:**

squares = [4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169]

squares.insert(0,2)

print(squares)

**Output of Question No. 4:**

A black screen with white text

Description automatically generated

1. Write some code that uses the remove() method to remove the value 49 from the squares list. Print the list afterwards to ensure the value has indeed been removed.

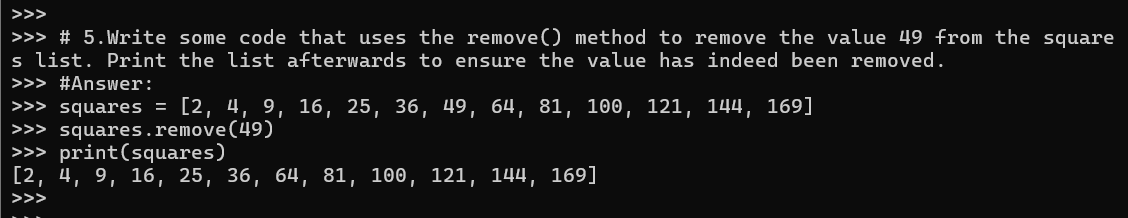
**Answer:**

squares = [2, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169]

squares.remove(49)

print(squares)

**Output of Question No. 5:**



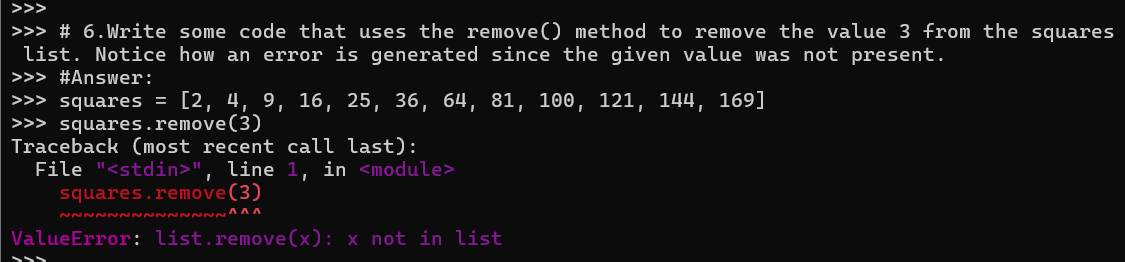
1. Write some code that uses the remove() method to remove the value 3 from the squares list. Notice how an error is generated since the given value was not present.

**Answer:**

squares = [2, 4, 9, 16, 25, 36, 64, 81, 100, 121, 144, 169]

squares.remove(3)

**Output of Question No. 6:**



1. Create a simple list that contains the values [1, 2, 3, 1, 2] and then use the remove () method to remove the value 2. Which value is removed?

**Answer:**

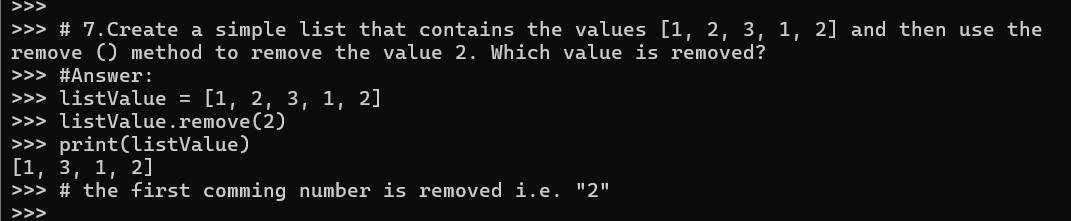
listValue = [1, 2, 3, 1, 2]

listValue.remove(2)

print(listValue)

# the first comming number is removed i.e. "2"

**Output of Question No. 7:**



1. Write some code that uses the pop() method to remove and display the last value of the squares list. Print the list afterwards to ensure the value displayed has been removed.

**Answer:**

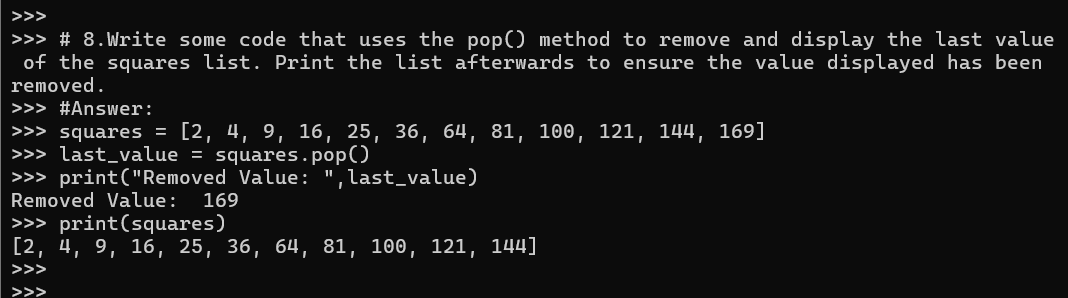
squares = [2, 4, 9, 16, 25, 36, 64, 81, 100, 121, 144, 169]

last\_value = squares.pop()

print("Removed Value: ",last\_value)

print(squares)

**Output of Question No. 8:**



1. Write some code that uses the pop() method to remove and display the first value of the squares list. Print the list afterwards to ensure the value has been removed.

**Answer:**

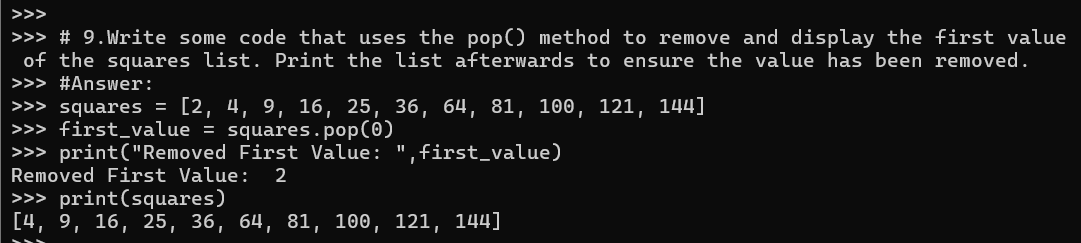
squares = [2, 4, 9, 16, 25, 36, 64, 81, 100, 121, 144]

first\_value = squares.pop(0)

print("Removed First Value: ",first\_value)

print(squares)

**Output of Question No. 9:**



1. Write some code that uses the sort() method with no arguments, to alphabetically sort the exact list of names shown below. Display the list after the sort has been called.

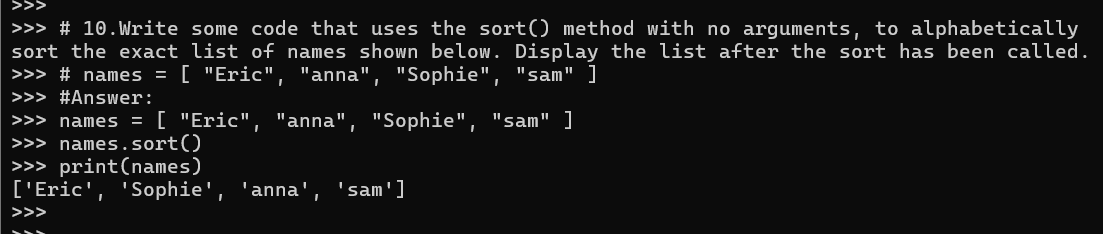
names = [ "Eric", "anna", "Sophie", "sam" ]

**Answer:**

names.sort()

print(names)

**Output of Question No. 10:**



1. Improve your previous solution so that the list is sorted correctly, ignoring the case used to write the names. To achieve this you will have to include a key argument in the form of a *lambda expression* that returns each string as uppercase letters only. Hint: you can use the str.upper() method to convert a name to uppercase letters.

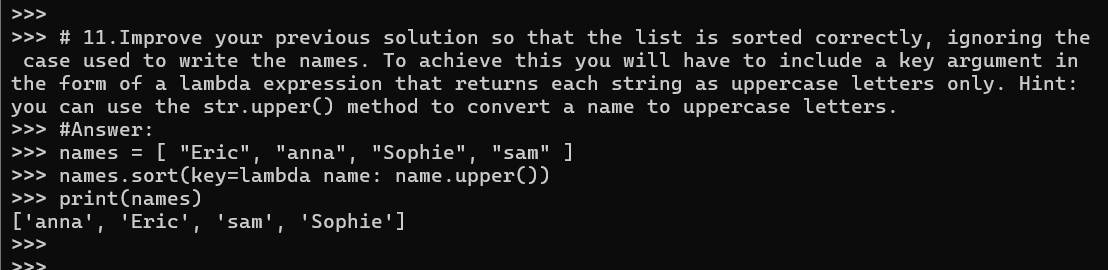
**Answer:**

names = [ "Eric", "anna", "Sophie", "sam" ]

names.sort(key=lambda name: name.upper())

print(names)

**Output of Question No. 11:**



1. Write some code that uses the reverse() method to reverse the values of the squares list. Print the list afterwards to ensure the values have been reversed.

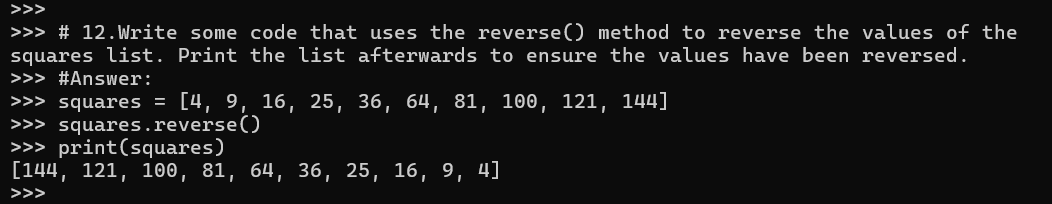
**Answer:**

squares = [4, 9, 16, 25, 36, 64, 81, 100, 121, 144]

squares.reverse()

print(squares)

**Output of Question No. 12:**



1. Write some code that finds the index of the colour blue.

colours = ["red", "green", "yellow", "blue", "red"]

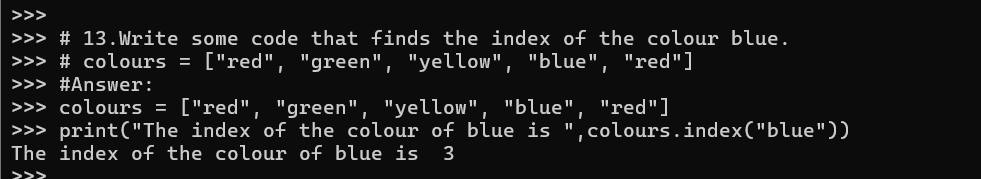
print(colours.index("blue"))

**Answer:**

colours = ["red", "green", "yellow", "blue", "red"]

print("The index of the colour of blue is ",colours.index("blue"))

**Output of Question No. 13:**



1. Write some code that makes a copy of the colours using the copy() method. Then make some changes to the original list. Print the contents of the copied list to ensure these changes have not affected the copy.

# colours = ["red", "green", "yellow", "blue", "red"]

**Answer:**

colours = ["red", "green", "yellow", "blue", "red"]

new\_colours = colours.copy()

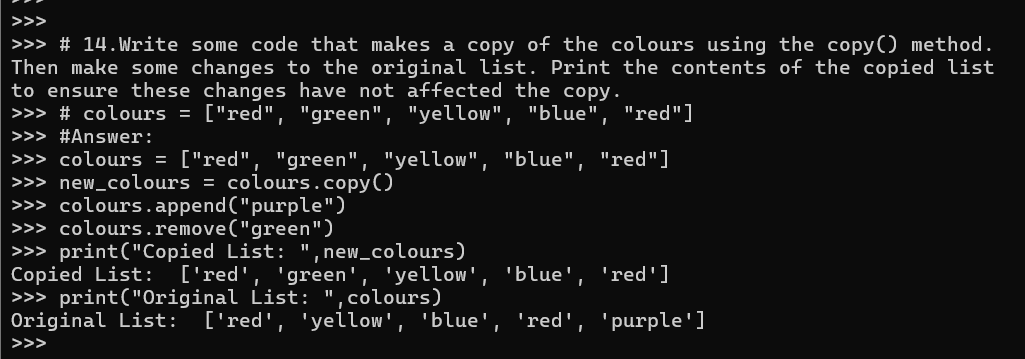
colours.append("purple")

colours.remove("green")

print("Copied List: ",new\_colours)

print("Original List: ",colours)

**Output of Question No. 14:**



1. Write some code that uses a list *comprehension* to create a list called cubes that contains the cubed values (x \* x \* x) of the numbers from 2 to 20 inclusive.

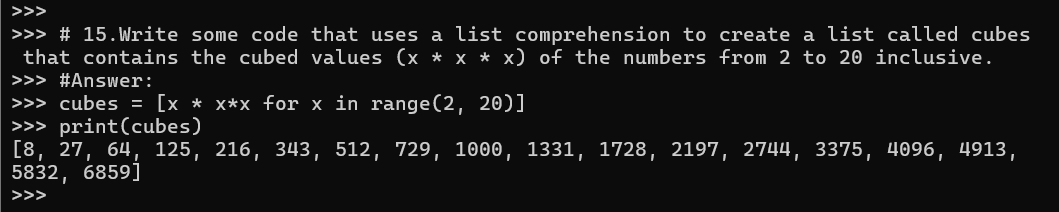
cubes = [x \* x\*x for x in range(2, 20)]

**Answer:**

cubes = [x \* x\*x for x in range(2, 20)]

print(cubes)

**Output of Question No. 15:**



1. Examine the below code and work out which user names will be placed in the some\_users list. What is the condition that has to be met for inclusion?

some\_users = [u for u in all\_users if len(u) < 8]

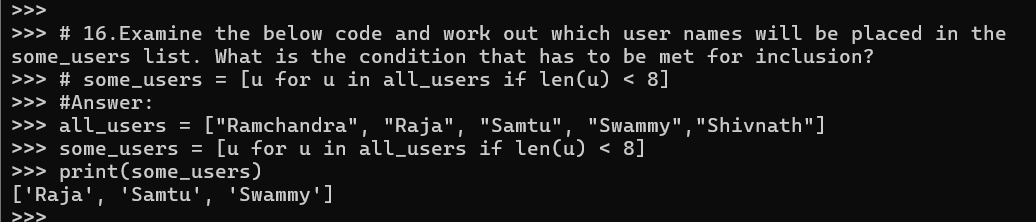
**Answer:**

all\_users = ["Ramchandra", "Raja", "Samtu", "Swammy","Shivnath"]

some\_users = [u for u in all\_users if len(u) < 8]

print(some\_users)

**Output of Question No. 16:**



1. Create a tuple called address that includes your own “house number”, “street” and, “postcode” as three different values.

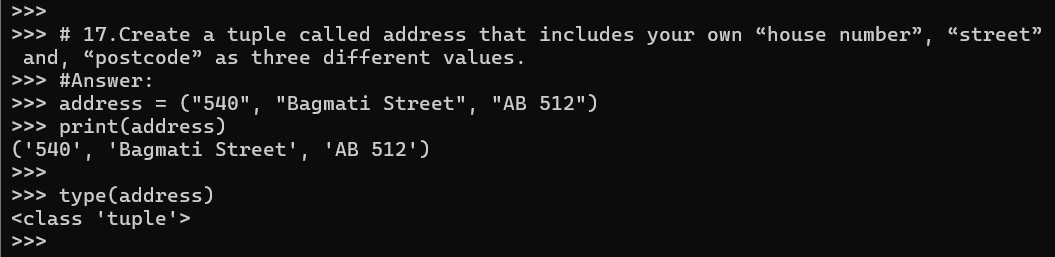
**Answer:**

address = ("540", "Bagmati Street", "AB 512")

print(address)

type(address)

**Output of Question No. 17:**



1. Try entering the below examples to create single element tuples. Then use the type() function to examine the data-type of the created variables.

empty = ()

the\_one = "Neo",

the\_one = ("Neo")

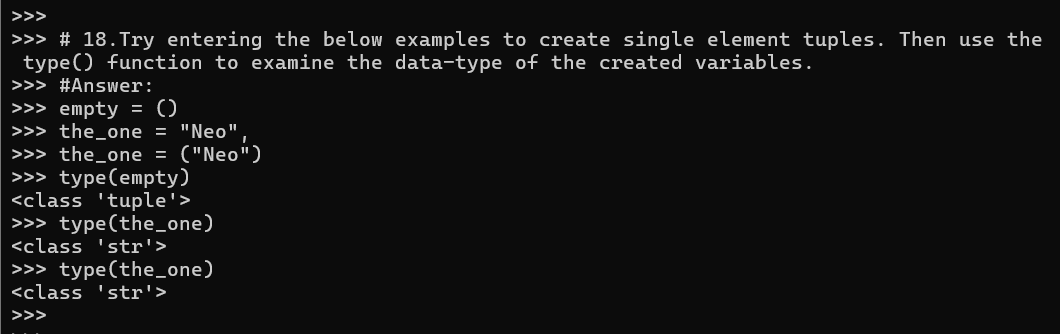
**Answer:**

type(empty)

type(the\_one)

type(the\_one)

**Output of Question No. 18:**

****

1. Use sequence unpacking to extract the values you stored within the address tuple earlier. Unpack the tuple into variables named house\_num, street and postcode.

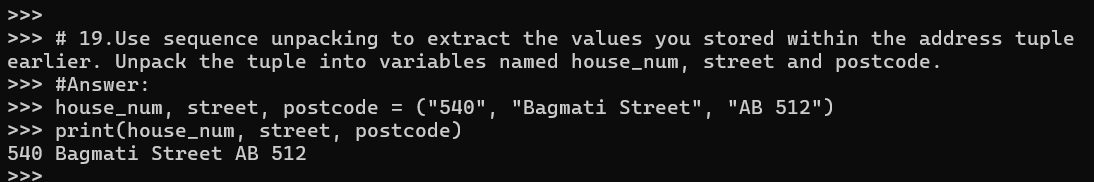
name, year, average\_grade = address

**Answer:**

house\_num, street, postcode = ("540", "Bagmati Street", "AB 512")

print(house\_num, street, postcode)

**Output of Question No. 19:**



1. Write some code that calls the print() function to output the contents of the address tuple you created earlier. Ensure you use the ‘\*’ prefix so that the elements are extracted before being passed to the function. Compare this with a version of the same code that calls the print() function without using the ‘\*’ prefix,

**Answer:**

house\_num = "540"

street = "Bagmati Street"

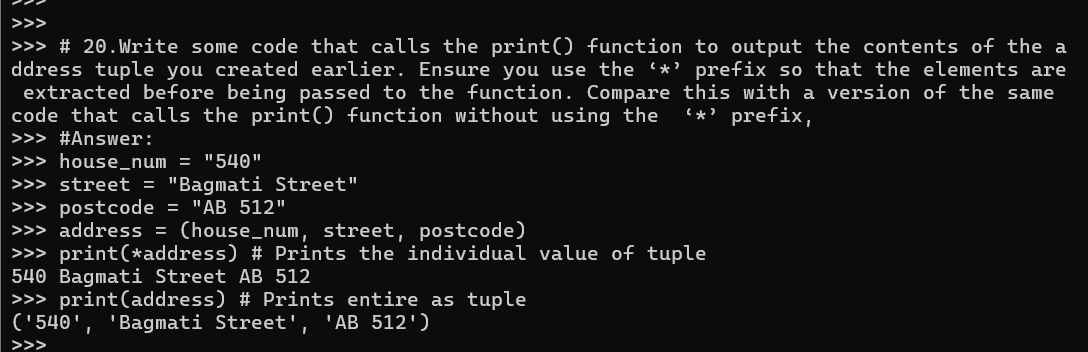
postcode = "AB 512"

address = (house\_num, street, postcode)

print(\*address) # Prints the individual value of tuple

print(address) # Prints entire as tuple

**Output of Question No. 20:**



1. Look at each of the phrases below and ensure you understand what each of these means. For any that you do not understand, do a little research to find a definition of each term. This research may involve looking back over these notes, or the associated lecture notes. It may also involve searching for these terms on the Internet.

* Method
* List Comprehension
* Tuple
* Tuple Packing
* Sequence Unpacking

**Answer:**

* **Method:** Methods are functions that are associated with an object and can manipulate its data or perform actions on it
* **List Comprehension:** List comprehension in Python is a concise way of creating lists from the ones that already exist.
* **Tuple:** Tuples are immutable, list like data structure and usually containing a sequence of heterogeneous elements that are accessed via unpacking or indexing.
* **Tuple Packing:** Tuples are immutable, list like data structure and usually containing a sequence of heterogeneous elements that are accessed via unpacking or indexing.
* **Sequence Unpacking:** Sequence Unpacking refers to the process of taking elements from iterable objects like tuples, lists, or dictionaries and assigning them to variables.