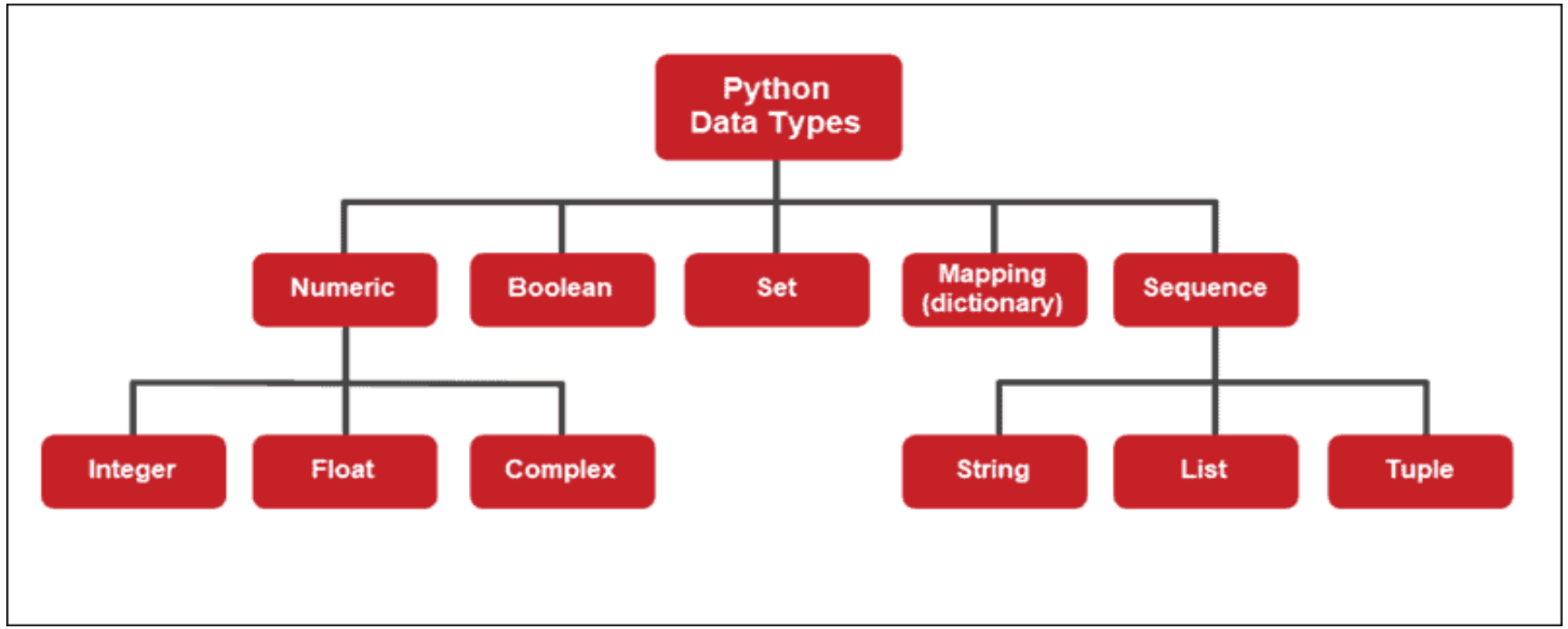
TUPLES in Python

Introduction

- So far, we have discussed about Strings and Lists in Sequence Data types, let's look at tuples now.



Tuples

- The data type tuple is an ordered sequence which is immutable and made up of one or more elements. Elements of a tuple are enclosed in parenthesis (round brackets) and are separated by comma.

- Unlike a string which consists of only characters, a tuple can have elements of different data types, such as integer, float, string, list or even another tuple.

  for e.g.)

  tuple1 = (1, 2, 3, 5, 7)

  tuple2 = (1, 2, [1,12,13], 5, 6)

Tuples

- If there is only a single element in a tuple then the element should be followed by a comma.

  for e.g.)

  tuple1 = (1,)

```
In [3]: tuple1 = (1,)

In [4]: type(tuple1)
Out[4]: tuple
```

Note:

If we assign the value without comma

it is treated as integer.

```
In [1]: tuple1 = (1)

In [2]: type(tuple1)
Out[2]: int
```

Tuples

Note:

It should be noted that a sequence without parenthesis is treated as tuple by default.

for e.g.)

```
In [7]: tuple1 = 1,2,3

In [8]: type(tuple1)
Out[8]: tuple

In [9]: tuple1
Out[9]: (1, 2, 3)
```

Accessing elements in a tuple

- Like list and string, elements of a tuple can be accessed using index values, starting from 0 and the last character is n-1 where n is the length of the tuple

  for e.g.)

  tuple1= (0, 2, 5, 10, 24)

Table: Indexing of elements in tuple

Positive Indices	0	1	2	3	4
tuple	0	2	5	10	24
Negative indices	-5	-4	-3	-2	-1

Accessing elements in a tuple

- The index specifies the character to be accessed in the tuple and is written in square brackets ([ ]).

for e.g.)

```
In [10]: tuple1 = (2, 6, "Hi", 5.6, 9+2j)

In [11]: tuple1[4]
Out[11]: (9+2j)
```

- The index can also be an expression including variables and operators but the expression must evaluate to an integer.

For e.g.)

```
In [12]: tuple1 = (2, 6, "Hi", 5.6, 9+2j)

In [13]: tuple1[5-3]
Out[13]: 'Hi'
```

Accessing elements in a tuple

- If we give index value out of this range then we get an *IndexError*.

  For e.g.)

```
In [14]: tuple1 = (2, 6, "Hi", 5.6, 9+2j)

In [15]: tuple1[8]
Traceback (most recent call last):

  File "<ipython-input-15-86446b69ce01>", line 1, in <module>
    tuple1[8]

IndexError: tuple index out of range
```

Tuple is immutable

- Tuple is an immutable data type. It means that the elements of a tuple cannot be changed after it has been created.

  Even if we try to make a change in the value, we will encounter an error

  For e.g.)

```
In [16]: tuple1 = (2, 6, "Hi", 5.6, 9+2j)

In [17]: tuple1[3] = "Ram"
Traceback (most recent call last):

  File "<ipython-input-17-7e9d2df3dc37>", line 1, in <module>
    tuple1[3] = "Ram"

TypeError: 'tuple' object does not support item assignment
```

Tuple is immutable

- However, if an element is of mutable data type, then that element can be modified.

  For e.g.)

```
In [18]: tuple1 = (1, 5, 3, [1,2,3], "Hi", 4.5)

In [19]: tuple1[3]
Out[19]: [1, 2, 3]

In [20]: tuple1[3][1] = "Ram"

In [21]: tuple1
Out[21]: (1, 5, 3, [1, 'Ram', 3], 'Hi', 4.5)
```

Operations on tuple: Concatenation

- To concatenate means to join. Python allows us to join two or more tuples using concatenation operator depicted by the symbol +.

For e.g.)

```
In [22]: tuple1 = (1, 5, 3, [1,2,3], "Hi", 4.5)

In [23]: tuple2 = ("Him", True, False, 0.6)

In [24]: tuple1 + tuple2
Out[24]: (1, 5, 3, [1, 2, 3], 'Hi', 4.5, 'Him', True, False, 0.6)
```

Note:

After the concatenation operation, there will be no change in the values of tuple1 and tuple2

```
In [25]: tuple1
Out[25]: (1, 5, 3, [1, 2, 3], 'Hi', 4.5)

In [26]: tuple2
Out[26]: ('Him', True, False, 0.6)
```

Operations on tuple: Concatenation

- Concatenation operator can also be used for extending an existing tuple. When we extend a tuple using concatenation a new tuple is created.

  For e.g.)

```
In [27]: tuple2 = ('Him', True, False, 0.6)

In [28]: tuple2 = tuple2 + (5,)

In [29]: tuple2
Out[29]: ('Him', True, False, 0.6, 5)

In [30]: tuple2 = tuple2 + (4,5,2)

In [31]: tuple2
Out[31]: ('Him', True, False, 0.6, 5, 4, 5, 2)
```

Operations on tuple: Concatenation

- The concatenation operator '+' requires that the operands should be of tuple type only. If we try to concatenate a tuple with elements of some other data type, TypeError occurs.

  For e.g.)

```
In [32]: tuple2 + "str"
Traceback (most recent call last):

  File "<ipython-input-32-483219fba9c9>", line 1, in <module>
    tuple2 + "str"

TypeError: can only concatenate tuple (not "str") to tuple
```

Operations on tuple: Repetition

- Python allows us to repeat the given tuple using repetition operator which is denoted by symbol *.

For e.g.)

```
In [43]: tuple1 = ('Him', True, False, 0.6, 5)

In [44]: tuple1*2
Out[44]: ('Him', True, False, 0.6, 5, 'Him', True, False, 0.6, 5)
```

Note:

After the repetition operation, there will be no change in the values of tuple1

```
In [43]: tuple1 = ('Him', True, False, 0.6, 5)

In [44]: tuple1*2
Out[44]: ('Him', True, False, 0.6, 5, 'Him', True, False, 0.6, 5)

In [45]: tuple1
Out[45]: ('Him', True, False, 0.6, 5)
```

Operations on tuple: Membership

- As we have already studied in Strings, Python has two membership operators:
  - 'in' and
  - 'not in'
- The 'in' operator checks if the element is present in the tuple, and returns True, else return False if element is not present in the tuple

For e.g.)

```
In [46]: tuple1 = ('Him', True, False, 0.6, 5)

In [47]: 5 in tuple1
Out[47]: True

In [48]: 6.7 in tuple1
Out[48]: False
```

Operations on tuple: Membership

- On the other hand, the 'not in' operator returns True if the element is not present in the tuple and returns False if element is present in the tuple

  For e.g.)

```
In [49]: tuple1 = ('Him', True, False, 0.6, 5)

In [50]: 6.7 not in tuple1
Out[50]: True

In [51]: 5 not in tuple1
Out[51]: False
```

Operations on tuple: Slicing

- Like string and lists, slicing operation can also be applied to tuples.
- Given a tuple t1, the slice operation t1[n:m] returns the part of the tuple t1 starting from index n (inclusive) and ending at m (exclusive).

  For e.g.)

```
In [52]: tuple1 = ('Him', True, False, 0.6, 5)

In [53]: tuple1[2:5]
Out[53]: (False, 0.6, 5)
```

- In other words, we can say that t1[n:m] returns all the elements from t1[n] till t1[m-1]

Operations on tuple: Slicing

Note:

The numbers of elements in the resulting tuple after slicing operation will always be equal to difference of two indices m and n, i.e., (m-n).

- If the first index is not mentioned, the slice starts from index 0.

  For e.g.)

```
In [56]: tuple1 = ('Him', True, 15, False, 0.6, 5)

In [57]: tuple1[:4]
Out[57]: ('Him', True, 15, False)
```

Operations on tuple: Slicing

- If the second index is not mentioned, the slicing is done till the length of the tuple.

  For e.g.)

```
In [58]: tuple1 = ('Him', True, 15, False, 0.6, 5)

In [59]: tuple1[1:]
Out[59]: (True, 15, False, 0.6, 5)

In [60]: tuple1[:]
Out[60]: ('Him', True, 15, False, 0.6, 5)

In [61]: tuple1[::-1]
Out[61]: (5, 0.6, False, 15, True, 'Him')
```

Operations on tuple: Slicing

- The slice operation can also take a third index that specifies the 'step size'. For example, t1[n:m:k], means every kth element has to be extracted from the tuple t1 starting from n and ending at m-1.

For e.g.)

```
In [62]: tuple1 = ('Him', True, 15, False, 0.6, 5)

In [63]: tuple1[1:5:2]
Out[63]: (True, False)
```

- By default, the step size is one and negative indexes can also be used for slicing.

For e.g.)

```
In [64]: tuple1 = ('Him', True, 15, False, 0.6, 5)

In [65]: tuple1[8:2:-2]
Out[65]: (5, False)
```

Traversing a tuple: using for loop

Code:

```
1  '''
2  Traversing a tuple using for loop
3  @author: Himanshu
4  '''
5  tuple1 = (1, 4, 1.2, "Ram", 7+2j)
6  for i in tuple1:
7      print(i, end=", ")
8
```

Output:

```
1, 4, 1.2, Ram, (7+2j),

...Program finished with exit code 0
Press ENTER to exit console.
```

Traversing a tuple: using for loop

- Another way of accessing the elements of the list is using range() and len() functions:

Code:

```
'''
Traversing a tuple using for loop
using range() and len() function
@author: Himanshu
'''
tuple1 = (1, 4, 1.2, "Ram", 7+2j)
l = len(tuple1)
for i in range(l):
    print(tuple1[i], end=", ")

```

Output:

```
1, 4, 1.2, Ram, (7+2j),

...Program finished with exit code 0
Press ENTER to exit console.
```

Traversing a tuple: using while loop

Code:

```
1  '''
2  Traversing a tuple using while loop
3  @author: Himanshu
4  '''
5  tuple1 = (1, 4, 1.2, "Ram", 7+2j)
6  l = len(tuple1)
7  i = 0
8  while i<l:
9      print(tuple1[i], end=", ")
10     i += 1
11
```

Output:

```
1, 4, 1.2, Ram, (7+2j),

...Program finished with exit code 0
Press ENTER to exit console.
```

Tuples methods and built-in functions

- **built-in functions:**

  len(), tuple(), count(), index(), sorted(), min(), max(), sum();

Method	Description	Example

len()	Returns the length or the number of elements of the tuple passed as the argument	>>> tuple1 = (10,20,30,40,50)
		>>> len(tuple1)
		5

tuple()	Creates an empty tuple if no argument is passed
	
	Creates a tuple if a sequence is passed as argument
		>>> tuple1 = tuple()
		>>> tuple1
		( )
		>>> tuple1 = tuple('aeiou')#string
		>>> tuple1
		('a', 'e', 'i', 'o', 'u')
		
		>>> tuple2 = tuple([1,2,3]) #list
		>>> tuple2
		(1, 2, 3)
		
		>>> tuple3 = tuple(range(5))
		>>> tuple3
		(0, 1, 2, 3, 4)

count()	Returns the number of times the given element appears in the tuple	>>> tuple1 = (10,20,30,10,40,10,50)
		>>> tuple1.count(10)
		3
		>>> tuple1.count(90)
		0

index()	Returns the index of the first occurrence of the element in the given tuple	>>> tuple1 = (10,20,30,40,50)
		>>> tuple1.index(30)
		2
		>>> tuple1.index(90)
		ValueError: tuple.index(x): x not in tuple

sorted()	Takes elements in the tuple and returns a new sorted list. It should be noted that, sorted() does not make any change to the original tuple	>>> tuple1 = ("Rama","Heena","Raj", "Mohsin","Aditya")
		>>> sorted(tuple1)
		['Aditya', 'Heena', 'Mohsin', 'Raj', 'Rama']

min()
 	
max()
 	
sum()
	Returns minimum or smallest element of the tuple
	
	Returns maximum or largest element of the tuple
	
	Returns sum of the elements of the tuple
		>>> tuple1 = (19,12,56,18,9,87,34)
		>>> min(tuple1)
		9
		>>> max(tuple1)
		87
		>>> sum(tuple1)
		235

Tuple Assignment

- Assignment of tuple is a useful feature in Python. It allows a tuple of variables on the left side of the assignment operator to be assigned respective values from a tuple on the right side.

  For e.g.)

```
In [108]: (a,b) = (10,20)

In [109]: print(a)
10

In [110]: print(b)
20
```

```
In [114]: tuple1 = (12, "Ram", 3.4)

In [115]: (rollNumber, name, marks) = tuple1

In [116]: rollNumber
Out[116]: 12

In [117]: name
Out[117]: 'Ram'

In [118]: marks
Out[118]: 3.4
```

Tuple Assignment

- The number of variables on the left should be same as the number of elements in the tuple.

  for e.g.)

```
In [119]: a, b, c = 12, "Ram", 4

In [120]: print(a,b,c)
12 Ram 4
```

- if variable and elements on both sides are not equal, then it would lead to an error

```
In [121]: a, b, c = 12, "Ram", 4, 56
Traceback (most recent call last):

  File "<ipython-input-121-fe7c7ca44411>", line 1, in <module>
    a, b, c = 12, "Ram", 4, 56

ValueError: too many values to unpack (expected 3)
```

Tuple Assignment

- If there is an expression on the right side then first that expression is evaluated and finally the result is assigned to the tuple.

  for e.g.)

```
In [122]: (num1, num2) = (10*5, 23+4)

In [123]: num1
Out[123]: 50

In [124]: num2
Out[124]: 27
```

Nested Tuples

- A tuple inside another tuple is called a nested tuple.

  for e.g.)

```
In [126]: tuple1 = (2, 6, 1, "Hi", (3, 6, 7), True, 4.5)

In [127]: tuple1[4]
Out[127]: (3, 6, 7)

In [128]: tuple1[4][1]
Out[128]: 6
```

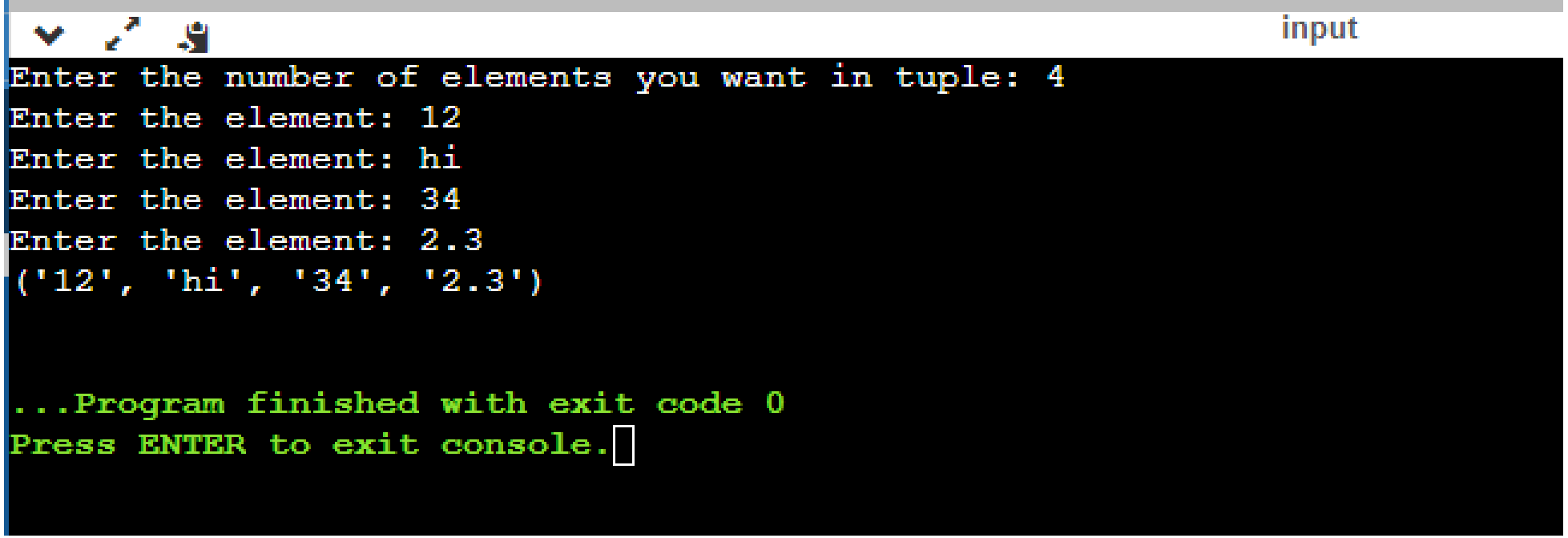
Practice problems on tuples

- Sample program: Input a tuple from user

Code:

```
'''
Input a tuple from user
@author: Himanshu
'''
n = int(input("Enter the number of elements you want in tuple: "))
list1 = list()
for i in range(n):
    ele = input("Enter the element: ")
    list1.append(ele)
tuple1 = tuple(list1)
print(tuple1)
```

Output:



Code:

```
'''
Input a tuple from user
@author: Himanshu
'''
n = int(input("Enter the number of elements you want in tuple: "))
tuple1 = tuple()
for i in range(n):
    ele = input("Enter the element: ")
    tuple1 = tuple1 + (ele,)
print(tuple1)
```

Output:

```
Enter the number of elements you want in tuple: 5
Enter the element: 15
Enter the element: Hi
Enter the element: ram
Enter the element: 34
Enter the element: 1.2
('15', 'Hi', 'ram', '34', '1.2')


...Program finished with exit code 0
Press ENTER to exit console.
```

Practice problems on tuple

- Sample program: swap 2 values without using a third temporary variable

```
1  '''
2  sample program: swap 2 values
3  @author: Himanshu
4  '''
5  a = 10
6  b = 40
7  (b,a) = (a,b)
8  print(a, b)
```

```
40 10


...Program finished with exit code 0
Press ENTER to exit console.
```

Practice problems on tuple

- Sample program: reverse a input tuple

Code:

```
'''
Reverse a tuple using slicing method
@author: Himanshu
'''
tuple1 = (10, 12, 45, 2, 1, 67)
tup_rev = tuple()
#using slicing method
tup_rev = tuple1[::-1]
print(tup_rev)

```

Output:

```
(67, 1, 2, 45, 12, 10)


...Program finished with exit code 0
Press ENTER to exit console.
```

Code:

```
'''
Reverse a tuple using concatenation
@author: Himanshu
'''
tuple1 = (10, 12, 45, 2, 1, 67)
tup_rev = tuple()
l = len(tuple1)
for i in range(l):
    tup_rev = (tuple1[i],) + tup_rev
print(tup_rev)

```

Output:

```
(67, 1, 2, 45, 12, 10)


...Program finished with exit code 0
Press ENTER to exit console.
```

Practice problems on tuples

- Program-17: Find the largest/smallest number in a tuple

Code:

```
'''
Program-17: Find the largest/smallest in a tuple
            using built-in function min() and max()
@author: Himanshu
'''
tuple1 = (10, 12, 45, 2, 1, 67, 23)
tup_min = min(tuple1)
tup_max = max(tuple1)
print("Minimum in tuple:",tup_min)
print("Maximum in tuple:",tup_max)

```

Output:

```
Minimum in tuple: 1
Maximum in tuple: 67


...Program finished with exit code 0
Press ENTER to exit console.
```

Code:

```
'''
Program-17: Find the largest/smallest in a tuple
@author: Himanshu
'''
tuple1 = (10, 12, 45, 2, 1, 67, 23)
tup_min = tuple1[0]
tup_max = tuple1[0]
for i in tuple1:
    if i<tup_min:
        tup_min = i
    if i>tup_max:
        tup_max = i
print("Minimum is:",tup_min)
print("Maximum is:",tup_max)

```

Output:

```
Minimum is: 1
Maximum is: 67


...Program finished with exit code 0
Press ENTER to exit console.
```

Practice problems on tuples

- Program-18: search for a given element in the tuple and return the index of that element

Code:

```
'''
Program-18: search for a given element in the tuple
            and return the index of that element
@author: Himanshu
'''
tuple1 = (10, 12, 45, 2, 1, 67, 23, 2)
ele = int(input("Enter the element you want to search: "))
if ele in tuple1:
    print(ele, "is present at index",tuple1.index(ele))
else:
    print(ele, "is not present in tuple")

```

Output:

```
Enter the element you want to search: 12
12 is present at index 1
```

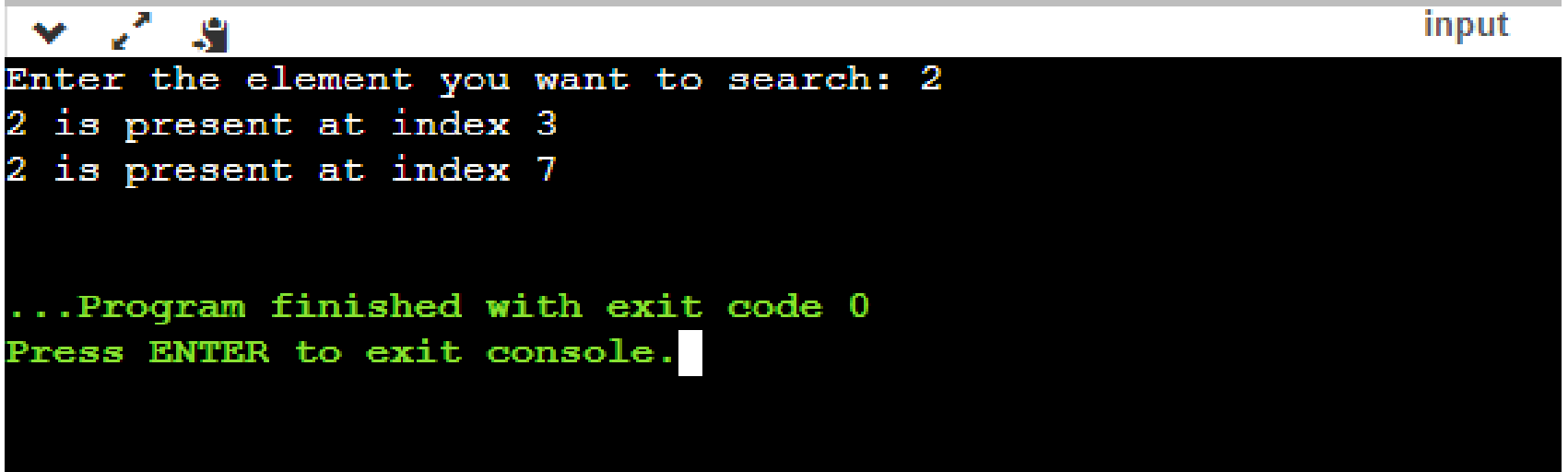
```
Enter the element you want to search: 121
121 is not present in tuple
```

```
Enter the element you want to search: 2
2 is present at index 3
```

Code:

```
'''
Program-18: search for a given element in the tuple
            and return the index of that element
@author: Himanshu
'''
tuple1 = (10, 12, 45, 2, 1, 67, 23, 2)
ele = int(input("Enter the element you want to search: "))
flag = 0
l = len(tuple1)
for i in range(l):
    if tuple1[i] == ele:
        print(ele, "is present at index",i)
        flag = 1
if flag == 0:
    print(ele, "is not present in tuple")
```

Output:



Summary

- Introduction to tuples
- Accessing elements in tuples (indexing)
- Operations on tuples:
  - Concatenation
  - Repetition
  - Membership
  - Slicing
- Traversing a tuple
- Programming problems on tuple

Assignment - 5

- Despite being a immutable data type, we can still modify the elements of tuple which is mutable data type (like lists). How is it possible?
- Read and note down the built-in functions in tuples (table: 10.1) page – 211
- Q1, 3, 5 and 7 from chapter- 10 "Tuples and Dictionaries" (NCERT, page-224)

Programming Assignment - 5

- Write a program to input names of n students and store them in a tuple. Also, input a name from the user and find if this student is present in the tuple or not.
- Write a program to input *n numbers* from the user. Store these numbers in a tuple. Print the maximum and minimum number from this tuple.
- Write a program to read a tuple of n integers and find their mean, median and mode.
- Write a program to read a tuple of elements. Modify this list so that it does not contain any duplicate elements, i.e., all elements occurring multiple times in the list should appear only once.