Computational Thinking and Programming - 2

Tuples

Definition

A tuple is a standard data type of Python that can store a sequence of elements belonging to any type. Python tuples are immutable, i.e. you cannot change the elements of a tuples in place.

Tuples in Python are values separated by commas in parentheses ( ).

For example :

```
a = 1, 2, 3        # a is the tuple (1, 2, 3)
a = (1, 2, 3)      # a is the tuple (1, 2, 3) are equivalent.
b=10,20,33           tuple
b=1                  integer
b=1,                 tuple  (1,)
```

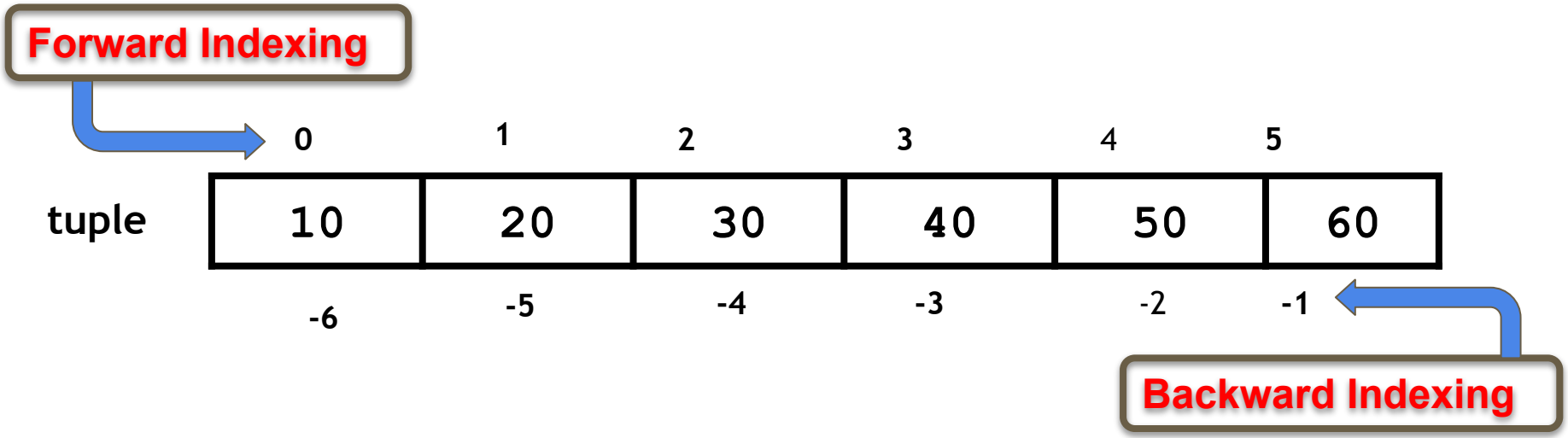
Examples of Tuples

Examples of Tuples are:

```
T = ()                                  # Empty Tuple
Number = (10,20,30,40)                  # Tuple of integer values
Values = (1.2,5.6,8,9,123.8)            # Tuple of float values
Names = ('Arun', 'Sheetal','Mohit')     # Tuple of strings


# tuple of different types of data
Student = (10,'Akshay Rawat',99.5)      # Tuple containing int, string and float values
```

Indexing



Indexing

The elements in a tuple are ordered (numbered), starting from 0.

It means that the first element of the tuple is numbered 0, the second element is numbered 1, third is numbered 2, and so on.

These numbers are called indices of tuple elements.

Python supports negative indices also.

In case of negative indices, -1 is the index for the last element, -2 for the second last element, and so on.

Creating tuples

To create a tuple, put a number of values in round brackets.

In general we can create a tuple in the form given below:

```
T = ( )                          # creating an empty tuple
T = (value1, value2,....)        #creating tuple with values
```

Example :

```
T1=(1,2,3,4,5)          # tuple containing elements of the same type
T2=(1,"RAHUL",80.0)     # tuple containing elements of mixed data types
```

Creating tuples (Contd.)

A tuple can be created from a sequence as per the syntax:

T = tuple(<sequence>)

where sequence can be any kind of sequence object including strings, tuples and lists.

Consider the following examples:

```
>>>T1 = tuple("PYTHON")
>>>print(T1)
('P','Y','T','H','O','N')
```

We can also create a tuple from single characters or single digits entered via the keyboard.

```
>>>T1 = tuple(input("Enter the values of the tuple:"))
Enter the values of the tuple:123456
>>>T1
('1','2','3','4','5','6')
>>> s1=eval(input("enter the values of a tuple"))
(10,20,30,40)
>>> s1
(10, 20, 30, 40)
```

Types of tuples

- **Empty tuple** :The empty tuple is a tuple containing no values.

  You can create an empty tuple as:

  L = tuple() or T = ( )

- **Single Element tuple**: If a single value is given in round brackets, python treats it as a value rather than a tuple. So in order to create a single element tuple, the value should be followed by a comma in round brackets.

```
>>> t=("python",)
>>> t
('python',)
```

Types of tuples

- **Long tuples** : If a list contains many elements, then to enter such long tuples, you can split them across several lines such as:

  tuple1 = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13)

- **Nested tuples**: A tuple can have an element in it, which itself is a tuple. Such a tuple is called a nested tuple, such as:

  tuple2 = (1, 2, (3, 6), 5)

Accessing elements of tuple:

- By using the tuple name and index numbers.

  For example:

  color = ('Red', 'Green', 'Pink', 'Black', 'Orange')

  print (color[2]) displays 'Pink'.

  print (color[-2]) displays 'Black'.

  print(color[6]) error

Note : If you give an index outside the legal indices (0 to length-1 or -length, -length+1), Python will raise IndexError

Accessing Elements of tuple:

- By using for loop where the range is specified by the tuple name.

For example:

```
color = ('Red', 'Green', 'Pink', 'Black', 'Orange')
for i in color:
   print(i, end=' ')
```

OR

```
color = ('Red', 'Green', 'Pink', 'Black', 'Orange')
for i in range ( 0, len(color)):
  print(color[i], end=' ')
```

Lists vs Tuples

Similarities

- Lists and Tuples are sequences.
- Lists and Tuples may contain heterogeneous elements.
- The concept of slicing is applicable to both Lists and Tuples.
- Both, Lists and Tuples support the concept of unpacking.

Differences

- A list is mutable, whereas a tuple is immutable. This means that a list's elements can be changed and deleted, but a tuple's elements cannot be.
- The concept of packing is applicable to tuples but not to lists.
- A list of one element can be created by putting the element in [ ] with or without any delimiter. To create a tuple of one element, the element must be followed by a comma.

Operations on tuples

OPERATIONS

1. Concatenation
2. Replication
3. Membership
4. Slicing

Concatenation

The + operator adds one tuple to the end of the other tuple. This operator requires both the operands to be of tuple type. You cannot add a number/ complex number / string.

Syntax : tuplename1+tuplename2

Example

```
t1=( 1, 2, 3 )
t2=( 5, 6, 7 )
>>> t1+t2
```

OUTPUT

```
(1, 2, 3, 5, 6, 7)
```

Replication

The * operator as a replication operator needs two operand i.e a tuple and an integer. Python replicates the tuple elements the given number of times.

Syntax : tuplename * integer data item

Example

```
>>> tuple1=(1,2,3)

>>> tuple1*3
```

OUTPUT

```
(1,2,3,1,2,3,1,2,3)
```

Comparison

All relational and comparison operators (==, >, <, <=, >=, !=) can be used to compare two tuples .

Python internally compares individual elements of the tuples in lexicographical order. This means that to check if the two tuples are equal, each corresponding element is compared and the two sequence must be of the same type.

Examples

Comparison	Result	Reason
(1,2,8,9)<(9,1)	True	Get result by comparing first element of both tuples **1<9 is True**
(1,2,8,9)<(1,2,9,1)	True	Get result by comparing third element of both tuples **8<9 is True**
(1,2,8,9)<(1,2,8,4)	False	Get result by comparing third element of both tuples **9<8 is False**

Membership

The operators in and not in are used to check whether element exists in the given tuple or not.

The in operator returns True if an element exists in the given tuple; False otherwise.

The not in operator returns True if an element does not exist in the given tuple; False otherwise.

Syntax : **element in <tuplename>**
element not in <tuplename>

Example

```
>>>tuple1=(1, 2, 12, 3)
>>>1 in tuple1
>>>tuple1=(1, 2, 12, 3)
>>>10 not in tuple1
>>>tuple1=("a", 12, "b")
>>>1 in tuple1
>>>tuple1=("a", "b", 83)
>>>"k" not in tuple1
```

OUTPUT

```
True
True
False
True
```

Tuple Slicing

Tuple slicing refers to retrieve few elements from the tuple which falls between two indexes.To extract the slice two indexes are given in square bracket separated by a colon (:) .

Syntax : tuplename [start : end : step]

where start, end and step are integers
start represents the starting index of tuple
end denotes the end index of tuple which is not inclusive
step denotes the distance between the elements to be sliced

Slicing of tuples

```
tuple1 = (1, 2, 3, 5, 6, 7, 10, 11,12)
tuple2=tuple1[2:6]
print("tuple after slicing :")
print(tuple2)
```

OUTPUT

tuple after slicing :
(3,5,6,7)

tuples also supports slice steps.

```
tuple1 = (1, 2, 3, 5, 6, 7, 10, 11, 12)
tuple2=tuple1[ 1 : 8 : 2 ]
print("tuple after slicing with step :")
print(tuple2)
```

OUTPUT

tuple after slicing with step:
(2,5,7,11)

Packing and Unpacking

Grouping of elements into a tuple is called Packing the elements in a tuple. Creating a tuple from a set of values is called packing.

Example:

```
t=("one","two","three")
```

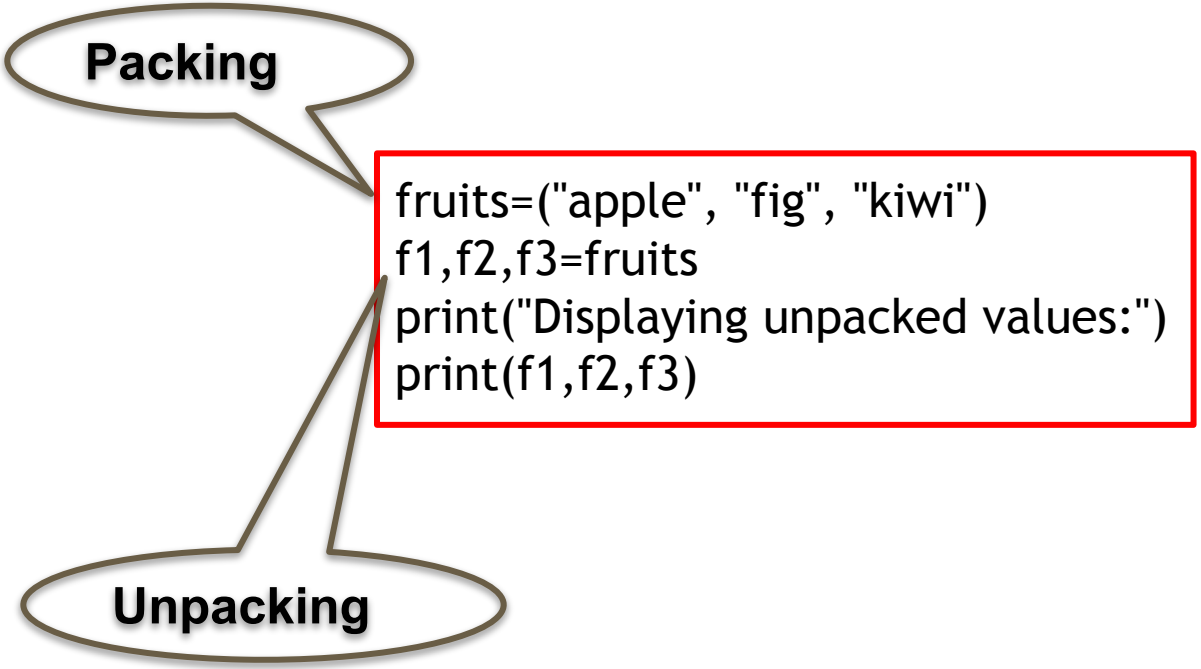
On the other hand, creating individual elements from a tuple is called unpacking. Unpacking is done as per the syntax:

```
<variable1>,<variable2>,<variable3>,..... = t
```

Example:

```
>>>x,y,z=t          # Unpacks the tuple elements into individual variables
>>>print(x)         # prints the value "one"
>>>print(z)         # prints the value "three"
```

Packing and Unpacking



OUTPUT

```
Displaying Unpacked values:
apple fig kiwi
```

Exercise :

Find the output of the following statements :
T1=(1,2,3,4,5,6,7,8)

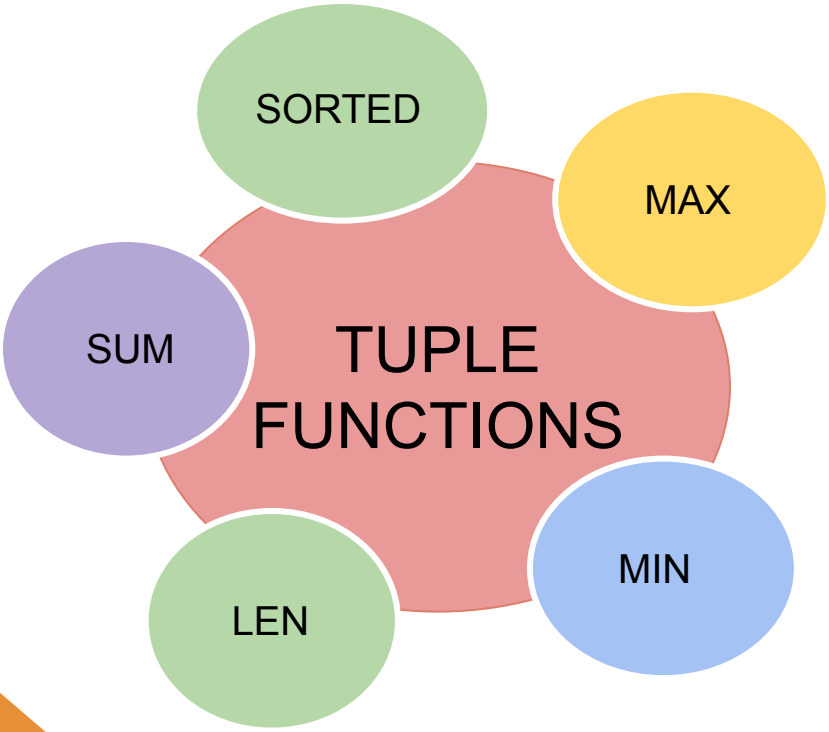
```
print(T1[ : ])
print(T1[3 : ])
print(T1[ : 4])
print(T1[-2 : -5 ])
```

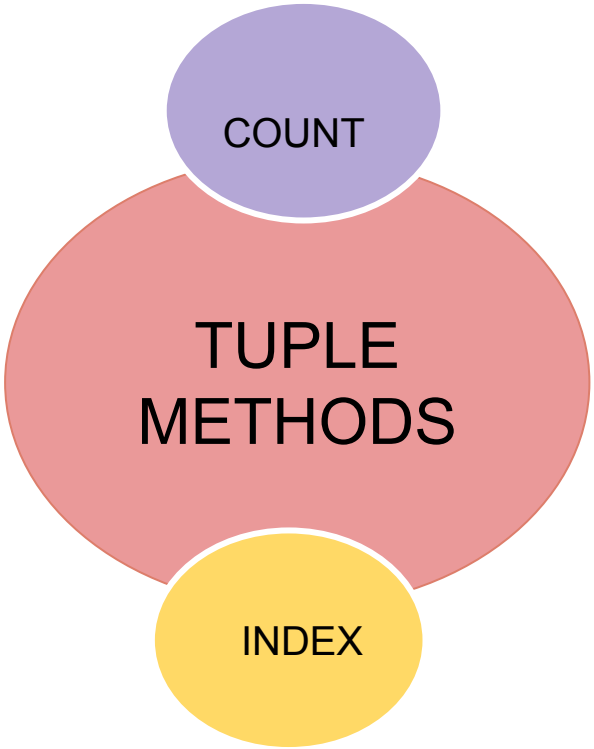
```
print(T1[1 : : 2])
print(T1[-1 : -5 : -2])
print(T1[: : -1])
print(T1[ : 7 : 2])
```

SOLUTION

(1, 2, 3, 4, 5, 6, 7, 8)	(2, 4, 6, 8)
(4, 5, 6, 7, 8)	(8, 6)
(1, 2, 3, 4)	(8, 7, 6, 5, 4, 3, 2, 1)
( )	(1, 3, 5, 7)

Tuple Functions and Methods





Tuple Functions and Methods

1. **len()**

Returns the length of the tuple or the number of elements in the tuple

Syntax : `len(tuplename)`

Example

```
tuple1=(10,20,30,40,50)
print("Length:",len(tuple1))
```

OUTPUT

```
Length : 5
```

Tuple Functions and Methods

2. max()

Returns the largest element of the tuple. All the elements must be of the same data type.

Syntax : `max(tuplename)`

Example

```
tuple1=(1,45,34,11,78)
print("Maximum:",max(tuple1))
```

OUTPUT

```
Maximum: 78
```

Tuple Functions and Methods

3. min()

Returns the smallest element of the tuple. All the elements must be of the same data type.

Syntax : min(tuplename)

Example

```
tuple1=(100,45,34,11,78)
print("Minimum:",min(tuple1))
```

OUTPUT

Minimum: 11

Tuple Functions and Methods

4. **sorted()**

This function returns the sorted values in ascending order (by default) or descending order in the tuple but the tuple remains unaffected. All the values in the tuple must be of the same data type.

Syntax : sorted(<tuple>,<reverse = True/False (default)>)

Example

```
>>> t1=(11,15,13,78,13,56,13)
>>> t2=sorted(t1)
>>> t2
```

OUTPUT

(11, 13, 13, 13, 15, 56, 78)

Tuple Functions and Methods

5. sum()

This function returns the sum of all elements in the tuple.

Syntax : sum(<tuplename>)

Example

```
tuple1=(10,20,30,40,50)
print("Sum:")
print(sum(tuple1))
```

OUTPUT

```
Sum:
150
```

Tuple Functions and Methods

6. count()

It returns the count of the item that has been passed as an argument. If the item is not present in the tuple, it returns zero.

Syntax : `tuplename.count(value)`

Example

```
t1=(11,22,33,44,33)
print("tuple:",11)
x=t1.count(33)
print("After counting:")
print(x)
```

OUTPUT

```
tuple:(11,22,33,44,33)
After counting :
2
```

Tuple Functions and Methods

7. index()

The index() method finds the first occurrence of the specified value.
The index() method raises an exception if the value is not found.

Syntax : `tuplename.index(value)`

Example

```
t1=(11,22,33,44,33)
print("tuple:",11)
x=t1.index(33)
print("Indexof 33:",x)
```

OUTPUT

```
tuple:(11,22,33,44,33)
Index of 33: 2
```

del statement

The del statement is used to delete elements and objects, but since tuples are immutable, which means that the individual elements cannot be deleted.So a complete tuple can be deleted as shown below.

Syntax : **del tuplename**

Example

```
>>>t1=(10,20,30,40,50,60,70)
>>>print(t1)
>>>del t1
>>>t1
```

OUTPUT

```
(10, 20, 30, 40, 50, 60, 70)
NameError: name 't1' is not
defined
>
```

Exercise :

Find the output of the following statements :

```
T1 = (23, 32, 4, 5, 2, 12, 23, 7, 9, 10, 23)
print(len(T1) + T1[-1])
print(T1[T1.count(23) + len(T1) -5])
print(T1.count(T1[6]))
print(T1.count(max(T1)))
```

Solution

```
34
10
3
1
```

Program #1

Program to calculate the average of the tuple of values entered by the user

```
t=eval(input("Enter tuple:"))
length=len(t)
m=s=0
for  i  in  range(length):
    s+=t[i]
m=s/length
print("Given tuple is:",t)
print("Sum of elements is : ",s)
print("Average of tuple is : ",m)
```

Output :

```
Enter tuple:(3,5,1,6,3)
Given tuple is: (3, 5, 1, 6, 3)
Sum of elements is :  18
Average of tuple is :  3.6
```

Program #2

Program to find the second maximum element from a tuple of numbers entered by the user along with the index of the largest element

```
t=eval(input("Enter the tuple elements :"))
sl= t[0]
l = t[0]
for i in range(len(t)):
    if t[i] > l:
        l = t[i]
for i in range(len(t)):
    if t[i]>sl and t[i]!=l :
        sl = t[i]
        ind=i
print("Second Maximum element is ",sl," at index ",ind)
```

Output :

Enter tuple:(3,5,1,6,3)
Given tuple is: (3, 5, 1, 6, 3)
Second Maximum element is 5 at index 1

Program #3

Program to enter a tuple of values and a number x to be searched. Search for x in the tuple. If present, display the position where found else display the message "element is not present in the tuple"

```
t=eval(input("Enter elements of the tuple:"))
length=len(t)
x=int(input("Enter the number"))
for  i  in  range(length):
     if x==t[i]:
          print("Element found at ",i,"position")
          break
else:
     print("Element is not present in the tuple")
```

Output :

Enter elements of the tuple:

(1,4,5,26,33)

Enter element to be searched:

26

Element found at position 4

Program #4

Program to input marks of five subjects out of 100 one by one from the user in a tuple and find aggregate and percentage obtained. After that display marks greater than 90%.

```
t=()
s=0
for i in range(5):
    mk=int(input("Enter the marks of subject :"))
    t=t+(mk,)
    s=s+mk
per=(s/500)*100
print("Total Marks :",s)
print("Percentage :",per)
print("Marks greater than 90%:")
for i in t:
    if i>90:
        print(i)
```

Output :

```
Enter the marks of subject : 77
Enter the marks of subject : 80
Enter the marks of subject : 66
Enter the marks of subject : 97
Enter the marks of subject : 56
Total Marks : 376
Percentage : 75.2
Marks greater than 90%:
97
```

Program #5

Program to count the frequency of a given element in a tuple of values.

```
t=eval(input("Enter the tuple:"))
element=int(input("Enter element to be searched :"))
c=0
for  i  in range(len(t)):
    if element==t[i]:
       c+=1
if c==0:
   print(element," not found in given tuple")
else:
   print(element," has frequency of ",c," in given
   tuple")
```

Output :

Enter the tuple:(1,2,3,4,4,5,4)
Enter element to be searched :4
4 has frequency of 3 in given tuple

Program #7

Write a program to input an integer n and create a tuple with n terms of the Fibonacci series.

```
n=int(input("Enter the number of terms :"))
first=0
second=1
t=(first,second)
for i in range(2,n):
    third=first+second
    first,second=second,third
    t=t+(third,)
print("Tuple created :", t)
```

```
Enter the number of terms :10

Tuple created : (0, 1, 1, 2, 3, 5, 8, 13, 21, 34)
```

Program #8

Write a program which inputs an integer x and an integer n and creates a tuple containing x, x^2, x^3,x^4,x^5..... x^n Also display the tuple so created.

```
n=int(input("Enter the number of terms :"))
x=int(input("Enter the value of x :"))
t=()
for i in range(1,n+1):
    t=t+(x**i,)
print("Tuple created :", t)
```

OUTPUT :

Enter the number of terms :5

Enter the value of x :2

Tuple created : (2, 4, 8, 16, 32)

Program #9

Write a program which creates a tuple containing the squares of all integers from 1 through N using a for loop.

```
N=int(input("Enter the number of terms :"))
t=()
for i in range(1,n+1):
    s=i*i
    t=t+(s,)
print("Tuple created :", t)
```

OUTPUT :

Enter the number of terms :5

Tuple created : (1, 4, 9, 16, 25)

Program #10

Write a program which creates a tuple (‘a’,’bb’,’ccc’,’dddd’,....) that ends with 26 copies of the letter a to z using for loop

```
t=()
for i in range(1,27):
    s=""
    for j in range(1,i+1):
        c=chr(96+i)
        s=s+c
    t=t+(s,)
print("Tuple created :", t)
```

OUTPUT :

Enter the number of terms :5

Tuple created :
(“a”,”bb”,”ccc”.....)

Program #10

Write a program to read email ids of n number of students and store them in a tuple. Now create two tuples to store username and domain name from those email ids.

```
n=int(input("Enter the number of students :"))
t=()
user=()
domain=()
for i in range(n):
    email=input("Enter the email ids :")
    t=t+(email,)
for i in t:
    s=i.split("@")
    user=user+(s[0],)
    domain=domain+(s[1],)
print("Email ids entered :", t)
print("Usernames :",user)
print("Domains :",domain)
```

OUTPUT :

Enter the number of students :3

Enter the email ids :sarab@gmail.com

Enter the email ids :monika@rediffmail.com

Enter the email ids :gagan@dpsmathuraroad.org

Email ids entered : ('sarab@gmail.com', 'monika@rediffmail.com', 'gagan@dpsmathuraroad.org')

Usernames : ('sarab', monika, 'gagan')

Domains : ('gmail.com', 'rediffmail.com', 'dpsmathuraroad.org')

Output Questions

Find the output of the following code :

```
Tuple = (3, 4.5, [4, 5, 6])
print("Original Tuple is:", Tuple)
Tuple[2][0] = 2
print("Updated Tuple is:", Tuple)
```

Solution

```
Original Tuple is: (3, 4.5, [4, 5, 6])
Updated Tuple is: (3, 4.5, [2, 5, 6])
```

Output Questions

Find the output of the following code :

```
Tuple = ("John", 23567, "Software Engineer")
eName, eID, eTitle = Tuple
print("Packed tuple is:", Tuple)
print("Employee name is:", eName)
print("Employee ID is:", eID)
print("Employee Title is:", eTitle)
```

Solutions

```
Packed tuple is: ("John", 23567, "Software Engineer")
Employee name is: John
Employee ID is: 23567
Employee Title is: Software Engineer
```

Output Questions

Find the output of the following code :

```
L= [2e-04, 'a', False, 87]
T = (6.22, "BOY", True, 554)
for i in range(len(L)):
    if L[i]:
        L[i] = L[i] + T[i]
    else:
        L[i] = L[i] * T[i]
        break
print(L)
```

Solution

```
[6.2202, 'aBOY', 0, 87]
```

Output Questions

Find the output of the following code :

```
A=(2,3,4,5,6)
B=-2,-1,0,1,2
for i in range(len(A)):
    print(A[i]+B[i])
B=B[:2]+(3,)+B[2:]
for i in B:
    if i in A:
        print(i)
print(A,B)
x=A[1]
y=B[-1]
print(x,y)
print(A,B)
```

Solution

```
0
2
4
6
8
3
2
(2, 3, 4, 5, 6) (-2, -1, 3, 0, 1, 2)
3 2
(2, 3, 4, 5, 6) (-2, -1, 3, 0, 1, 2)
```

Practice Programs

Q1. Write a program to input two different tuples and then create a tuple that all the common elements
from the two tuples. For example :
If tuple1=(11, 22, 44, 55, 99, 66) and tuple2=(44, 22, 55, 12, 56) then

tuple3 is to be created as tuple3 = (22, 44, 55)

Q2. Write a program to input a tuple and then double the even elements of the tuple and triple the odd elements of the tuple.

Q3. Write a program to input a tuple containing names of cities and then display the names of all those cities that start with the alphabet 'A'.
For example if the tuple contains ("AHMEDABAD", CHENNAI", "NEW DELHI", "AMRITSAR"," AGRA"), then the program should print AHMEDABAD, AMRITSAR and AGRA.

Q4. Write a program to enter a tuple and then push all the zeros in the tuple to the end of the tuple.

For example: If the tuple contains (0, 2, 3, 4, 6, 7, 0, 1), then the program should re-tange the elements as (2, 3, 4, 6, 7,1, 0, 0)