Nabarun Kar

CSE 2nd Year 3rd Sem

Roll:33

<u>OBSERVATIONS</u>

A)

```
>>> tup=(1,2,3)   #Declaration
>>> print(tup)    #print
(1, 2, 3)
```

Observation:

Tuples are easy to form, here we are printing a tuple after declaration

B)

```
>>> tup=(1,2,3)  #Declaration of first tuple
>>> next=(10,20,30) #Declaration of second tuple
>>> new=tup+next
>>> print(new)
(1, 2, 3, 10, 20, 30)
```

Observation:

Joining two tuples and storing it in a new one. Concatenation is done easily using ‘+’

C)

```
>>> tup=(5,)  #Declaration
>>> tup=tup*5   #Multiplying the element with 5 so that it gets printed 5 times
>>> print(tup)
(5, 5, 5, 5, 5)
```

OBSERVATION- the data inside the tuple is getting printed 5 times.

D)

```
>>> tup=(1,2,3,10,20,30)  #Declaration
>>> print(tup[2:4])
(3, 10)
```

Observation:

Specifying positions of the elements in the tuples that need to be displayed

E)

CONVERT THE LIST IN A TUPLE

```
>>> tup=(1,2,3)  #Declaration
>>> new=list(tup)  #Converting the tuple into a list by using the function
>>> print(new)
[1, 2, 3]
```

F)

FIND THE MAXIMUM AND MINIMUM ITEM IN A TUPLE AND ALSO FIND THE LENGTH OF THE TUPLE

```
>>> tup=(1,2,3,4,5) #Declaration
>>> max(tup)        #Using max function to find teh maximum value of an element in the tuple
5
>>> min(tup)        #using min function to find the minimum value of an element in the tuple
1
>>> len(tup)       #using len function to find the length of teh tuple
5
>>>
```

1)

WRITE A PROGRAM TO REVERSE A TUPLE

```
>>> x=("W3RESOURSE")  #Declaration
>>> y=reversed(x)     #Reverse the tuple
>>> print(tuple(y))
('E', 'S', 'R', 'U', 'O', 'S', 'E', 'R', '3', 'W')
>>> x=(1,2,3,4,5)      #Declaration
>>> y=reversed(x)      #Reversing the tuple
>>> print(tuple(y))    #Displaying
(5, 4, 3, 2, 1)
>>>
```

2) WRITE A PROGRAM TO COUNT THE ELEMENTS IN A LIST UNTIL THE ELEMENT IS A TUPLE.

```
num = [10,20,30,(10,20),40]  #Declaration of a list and one of the element inside that list will be a tuple present in some position
ctr = 0                  #ctr is declared. It will be incremented inside the loop and will ultimately give us the result
for n in num:
    if isinstance(n, tuple):  #Checking whether the element is tuple or not
        break               #If it is tuple then break and get out of the loop
    ctr += 1                #If it is not tuple then the ctr will be incremented
print(ctr)
```

OUTPUT

```
3
```

3)

Write a Python program to find the index of an item of a tuple. Convert a string to a tuple. Check it for

all possible parameters of index function. Check it for an item which is not present.

```
tuplex = tuple("index tuple")  #Declaration
print(tuplex)
index = tuplex.index("p")      #get index of the first item whose value is passed as parameter
print(index)
index = tuplex.index("p", 5)   #define the index from which you want to search
print(index)
index = tuplex.index("e", 3, 6)  #define the segment of the tuple to be searched
print(index)
```

```
index = tuplex.index("y")    #if item not exists in the tuple return ValueError Exception
```

OUTPUT

```
('i', 'n', 'd', 'e', 'x', ' ', 't', 'u', 'p', 'l', 'e')
8
8
3
Traceback (most recent call last):
  File "D:\PYTHON\college\tuple.py", line 9, in <module>
    index = tuplex.index("y")
ValueError: tuple.index(x): x not in tuple
```

Observation:

The index function is used to search for an element in a list or a tuple. It returns the index at which the search element is present. If the element is not present it returns a ValueError which can be avoided using a try & error block.

4)

Write a program in Python to do slicing in all possible ways with all possible parameters, providing

positive and negative values for step. Also, perform slicing from start and end both.

```
#create a tuple
tuplex = (2, 4, 3, 5, 4, 6, 7, 8, 6, 1)
#used tuple[start:stop] the start index is inclusive and the stop index
slice = tuplex[3:5]
#is exclusive
print(slice)
#if the start index isn't defined, is taken from the beg inning of the tuple
slice = tuplex[:6]
print(slice)
```

```
#if the end index isn't defined, is taken until the end of the tuple

slice = tuplex[5:]

print(slice)

#if neither is defined, returns the full tuple

slice = tuplex[:]

print(slice)

#The indexes can be defined with negative values

slice = tuplex[-8:-4]

print(slice)

#create another tuple

tuplex = tuple("HELLO WORLD")

print(tuplex)

#step specify an increment between the elements to cut of the tuple

#tuple[start:stop:step]

slice = tuplex[2:9:2]

print(slice)

#returns a tuple with a jump every 3 items

slice = tuplex[::4]

print(slice)

#when step is negative the jump is made back

slice = tuplex[9:2:-4]

print(slice)
```

OUTPUT

```
(5, 4)

(2, 4, 3, 5, 4, 6)

(6, 7, 8, 6, 1)

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

(3, 5, 4, 6)

('H', 'E', 'L', 'L', 'O', ' ', 'W', 'O', 'R', 'L', 'D')
```

('L', 'O', 'W', 'R')

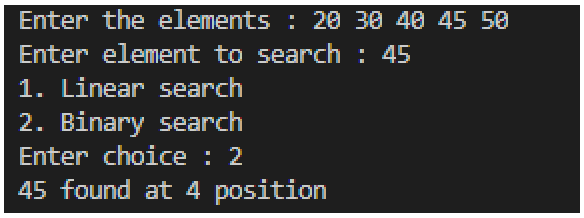
('H', 'O', 'R')

('L', ' ')

Observation:

Like lists, tuples can also be sliced using slicing method in Python.

OUTPUT



```
Enter the elements : 20 30 40 45 50
Enter element to search : 45
1. Linear search
2. Binary search
Enter choice : 2
45 found at 4 position
```