Topics:

- **UNIT – II [14 Lectures]**
- **Data Structures:** Sequence, Lists, Tuple, Sets, Dictionaries
- **Strings and its operations:** Concatenating, Appending, Multiplying strings, Built-in String methods and functions, Slice Operation, Iterating String, String Module
- **Modules:** Importing module, The from..import statement, Name of Module, Making your own modules, The dir()function, The Python Module,Math module, OS Module, Sys Module, Random module
- **Introduction to Functions:** Declaration and Definition, Variable Scope and Lifetime, Return Statements, Types of Arguments, Lambda function, Recursion
- **Functional Programming:** filter() function, map()function, reduce()function

Tuple

- An ordered group of sequence separated by symbol comma (,) and enclosed inside the parenthesis
- A tuple is a sequence of immutable objects, therefore tuple cannot be changed
- In Python, tuple is similar to a list. Only the difference is that list is enclosed between square bracket, tuple between parenthesis and List have mutable objects whereas Tuple have immutable objects.
- The main difference between lists and tuples is that you can change the values in a list but not in a tuple.
- This means that while tuple is a read only data type, the list is not.

- **Creating a Tuple**
- A tuple is created by placing all the items (elements) inside a parentheses ( ), separated by comma. The parentheses are optional but is a good practice to write it. A tuple can have any number of items and they may be of different types (integer, float, list, string etc.).
- **Syntax:** tuple1= ( ) # creation of empty tuple

```
tuple2 = ('Sequence 1', )
# , symbol is mandatory  with out which it becomes just a string
                    assignment operator
tuple3= ('Sequence1','Sequence2')
print(tuple1,tuple2,tuple3)
```

- **NOTE:**

If Parenthesis is not given with a sequence, it is by default treated as Tuple.

```
tup1=()#creates an empty tuple
tup2=(10,)#creates a tuple with single element
tup3=(10,20,30)# creates a tuple with integer values
tup4=('p','y','t','h','o','n')#creates a tuple of characters
tup5=("Iam","Good","at","python","programming")#creates a tuple of strings
tup6=(10.1,20.03,30.4)#creates a tuple of foating point numbers
tup7=(100,"Raj",76.72,'A')#creates a tuple of mixed type
tup8=(1,3.2,"mouse", [8, 4, 6], (1, 2, 3))#creates a tuple of characters
print(tup1)
print(tup2)
print(tup3)
print(tup4)
print(tup5)
print(tup6)
print(tup7)
print(tup8)
```

```
#Program to illustrate the use of divmod( )
q,r=divmod(100,3)
print("Quotient=",q)
print("Reminder=",r)


   #Tuple with out Parentheses
   a='C',"PERL",10,30.12
   print('C',"PERL",10,30.12)
   print(type(a))


#Necessity of having a comma in the Tuple

tup=(10,)#comma after first element
print(type(tup))

tup1=(10)#comma missing
print(type(tup1))
```

Note: If you want to create a tuple with a single element, then you must add a comma after the element. In the absence of a comma, Python treats the element as an ordinary data type

- **Accessing Elements in a Tuple**
- 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 index from 0 to 5. Trying to access an element other that (6, 7,…) will raise an IndexError.
- The index must be an integer, so we cannot use float or other types. This will result into TypeError. Likewise, nested tuple are accessed using nested indexing as shown in the example below.
- Python allows negative indexing for its sequences.
- **Example**

```
tup=(1,2,3,4,5,6,7,8,9,10)
print("tup[3:6]=",tup[3:6])
print("tup[:8]=",tup[:4])
print("tup[4:]=",tup[4:])
print("tup[:]=",tup[:])
```

Tuple Operations : Slicing

- We can access a range of items in a tuple by using the slicing operator - colon "**:**".
- A subpart of a tuple can be retrieved on the basis of index. This subpart is known as tuple slice.
- If the index provided in the Tuple slice is outside the list, then it raises an IndexError exception

Example

```
my_tuple = ('p','r','o','g','r','a','m')
print(my_tuple[1:4])
print(my_tuple[:-5])
print(my_tuple[5:])
print(my_tuple[:])
print(my_tuple[-1])
print(my_tuple)
```

```
# Tuple Programs
tup=()
tup1=('a','bc',55,4.32)
tup2=('d',34)
print(tup)
print(tup1)
print(tup1[0])#Prints first element of the Tuple
print(tup1[1:3])#Prints eles starting from 2nd till 3rd
print(tup1[2:])#Prints eles starting from 3rd element
print(tup1*2)#Repeats the Tuple
print(tup1+tup2)#Concatenation of two tuples
```

```
n_tuple = (1,"mouse", [8, 4, 6], (1, 2, 3))
print(n_tuple[1][3])
print(n_tuple[2][1])
print(n_tuple[0])
print(n_tuple[-2])
print(n_tuple[-2][-1])
print(n_tuple[-1][-2])
...
```

- **Updating a Tuple**
- Unlike lists, tuples are immutable. This means that elements of a tuple cannot be changed once it has been assigned.
- But, if the element is itself a mutable datatype like list, its nested items can be changed. We can also assign a tuple to different values (reassignment).
- **Example**

```
my_tuple = (4, 2, 3, [6, 5])
my_tuple[3][0] = 9
print(my_tuple)#(4, 2, 3, [9, 5])
my_tuple[1]=40
#TypeError:
#'tuple' object does not support item assignment
print(my_tuple)
```

We can only extract values from a tuple to form another tuple

```
#prm to extract values from a tuple
tup1=(1,2,3,4,5)
tup2=(6,7,8,9,10)
tup3=tup1+tup2
print("tup3:",tup3)
```

- We can use + operator to combine two tuples. This is also called **concatenation**.
- We can also **repeat** the elements in a tuple for a given number of times using the * operator.
- Both + and * operations result into a new tuple.
- **Example**
- # Concatenation

```
print((1, 2, 3) + (4, 5, 6))
```

- # Repeat

```
print(("Repeat",) * 3)
```

- **Deleting element in a Tuple**
- we cannot delete or remove items or elements from a tuple.
- But deleting a tuple entirely is possible using del statement.
- **Example**

```
#Deleting ele in tuple
tup1=(1,2,3,4,5)
del tup1[3]
print(tup1)
```

```
    del tup1[3]
TypeError: 'tuple' object doesn't support item deletion
```

```
my_tuple=('p','r','o','g','r','a','m')
del my_tuple
# will delete the tuple data
print(my_tuple)
# will show an error sice tuple data is already deleted
```

Basic Tuple Operations

Operation	Expression	Output
Length	Len(1,2,3,4,5,6)	6
Concatenation	(1+2+3)+(4+5+6)	(1,2,3,4,5,6)
Repetition	('Python')*3	'PythonPythonPython
Membership	6 in (1,2,3,4,5,6,7,8,9)	True
Iteration	for i in(1,2,3,4,5,6,7,8,9): print(i, end=' ')	1,2,3,4,5,6,7,8,9
Comparison (Use >,<,==)	Tup1=(1,2,3,4,5) Tup2=(1,2,3,,4,5) print(Tup1>Tup2)	False
Maximum	max(1,0,3,8,2,9)	9
Minimum	min(1,0,3,8,2,9)	0
Convert to tuple(converts a sequence into a tuple)	Tuple("Hello") Tuple([1,2,3,4,5])	('H','E','L','L','O') (1,2,3,4,5)

- **Tuple Methods**
- Methods that add items or remove items are not available with tuple. Only the following two methods are available.
- count(x) Return the number of items that is equal to x
  - **Sysntax: tup.count(obj)**
- index(x) Return index of first item that is equal to x
  - **Syntax:Tup.index(obj)**
- **Example**

```
my_tuple=('a','p','p','l','e',)
print("p appears ", my_tuple.count('p'),"times in",my_tuple)
print("Length of my_tuple is ", my_tuple.index('l'))
```

Tuple functions

- all (T) - Return True if all elements of the tuple are true (or if the tuple is empty).
- any(T) - Return True if any element of the tuple is true. If the tuple is empty, return False.
- enumerate(T) - Return an enumerate object. It contains the index and value of all the items of tuple as pairs.
- sorted(T) - Take elements in the tuple and return a new sorted list (does not sort the tuple itself).
- sum(T) - Retrun the sum of all elements in the tuple.
- tuple(T) - Convert an iterable (list, string, set, dictionary) to a tuple.
- cmp(t1,t2) – To compare the two given tuples

- **cmp(tuple1,tuple2)**
- **Explanation:**If elements are of the same type, perform the comparison and return the result. If elements are different types, check whether they are numbers.
- If numbers, perform comparison.
- If either element is a number, then the other element is returned.
- Otherwise, types are sorted alphabetically .
- If we reached the end of one of the lists, the longer list is "larger." If both list are same it returns 0.

- **Ex:**

```
data1=(10,20,'rahul',40.6,'z')
data2=(20,30,'sachin',50.2)
print(cmp(data1,data2))
print(cmp(data2,data1))
data3=(20,30,'sachin',50.2)
print(cmp(data2,data3))
```

- **Example**

```
pyTuple=(20,55,43,22,67,90,0)
print(all(pyTuple))
print(any(pyTuple))
print(len(pyTuple))
print(max(pyTuple))
print(min(pyTuple))
print(sum(pyTuple))
print(sorted(pyTuple))
a=enumerate(pyTuple)
print(tuple(a))
for item in enumerate(pyTuple):
    print(item)

# data=(10,20,'Ravi',40.6,'z')
        #Try out
```

Tuple Assignment

- Tuple assignment is a vey powerful feature in python
- It allows a tuple of variables on the left side of the assignment operator to be assigned values from a tuple given on the right side of the assignment operator, where each value is assigned to its respective variable
- In case of an expression is specified on the right side of the assignment operator, first that expression is evaluated and then assignment is done

Program to show the diff. ways of tuple assignment

```
#an unnamed tuple of values assigned to values of another unnamed tuple
(val1,val2,val3)=(10,20,30)
print(val1,val2,val3)

tup1=(100,200,300)
(v1,v2,v3)=tup1
print(v1,v2,v3)

#Expressions are evaluated before assignment
(val1,val2,val3)=(2+4,5/3+4,9%6)
print(val1,val2,val3)
```

Note: Ensure that number of values on the both sides of assignment operator must be same

```
(val1,val2,val3)=(1,2)
print(val1,val2,val3)
  (val1,val2,val3)=(1,2)
ValueError: not enough values to unpack (expected 3, got 2)
```

Nested Tuples

- Python allows to define a tuple inside another tuple.
- This is called as Nested Tuple

```
#Nested Tuples

Students_Details=(("Ashwin","B.Tech",92.03),
("Jagruti","M.Tech",97.67),("RamNayak","B.com",83.87))
for i in Students_Details:
    print(i)
```

For Details Contact Me @ :

9247448766

ravikanth27787@gmail.com

python

Programming Language