SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

19AMB302-FULL STACK AI

M.POORNIMA DEVI,AP/AIML

LISTS,TUPLES,INDEXING AND SLICING

Tuples in Python

•Python Tuple is a collection of objects separated by commas.

• In some ways, a tuple is similar to a Python list in terms of indexing, nested objects, and repetition but the main difference between both is Python tuple is immutable, unlike the Python list which is mutable.

Creating Python Tuples

There are various ways by which you can create a tuple in Python. They are as follows:

1.Using round brackets

2.With one item

3.Tuple Constructor

Create Tuples using Round Brackets ()

To create a tuple we will use () operators.

EXAMPLE:

```
var = (“ONE", “TWO", “THREE")
print(var)
```

OUTPUT:('Geeks', 'for', 'Geeks')

Create a Tuple With One Item

Python 3.11 provides us with another way to create a Tuple.

EXAMPLE:

```
values : tuple[int | str, ...] = (1,2,4,"Geek")
print(values)
```

OUTPUT:

```
(1, 2, 4, 'Geek')
```

Tuple Constructor in Python

To create a tuple with a Tuple constructor, we will pass the elements as its parameters.

```
tuple_constructor = tuple(("dsa", "developement", "deep
learning"))
print(tuple_constructor)
```

EXAMPLE:

```
tuple_constructor = tuple(("dsa", "developement", "deep
learning"))
print(tuple_constructor)
```

OUTPUT:

```
('dsa', 'developement', 'deep learning')
```

What is Immutable in Tuples?

•Tuples in Python are similar to Python lists but not entirely. Tuples are immutable and ordered and allow duplicate values. Some Characteristics of Tuples in Python.

•We can find items in a tuple since finding any item does not make changes in the tuple.

•One cannot add items to a tuple once it is created.

•Tuples cannot be appended or extended.

•We cannot remove items from a tuple once it is created.

Accessing Values in Python Tuples

Tuples in Python provide two ways by which we can access the elements of a tuple.

Using a positive index

Using a negative index

Python Access Tuple using a Positive Index

Using square brackets we can get the values from tuples in Python.

EXAMPLE:

```
var = (“HI", “HOW", “ARE")
print("Value in Var[0] = ", var[0])
print("Value in Var[1] = ", var[1])
print("Value in Var[2] = ", var[2])
```

OUTPUT:

```
Value in Var[0] = HI
Value in Var[1] = HOW
Value in Var[2] = ARE
```

Access Tuple using Negative Index

In the above methods, we use the positive index to access the value in Python, and here we will use the negative index within [].

EXAMPLE:

```
var = (1, 2, 3)
print("Value in Var[-1] = ", var[-1])
print("Value in Var[-2] = ", var[-2])
print("Value in Var[-3] = ", var[-3])
```

OUTPUT:

```
Value in Var[-1] = 3 Value in Var[-2] = 2 Value in Var[-3] = 1
```

Different Operations Related to Tuples

- Below are the different operations related to tuples in Python:

1. Concatenation
2. Nesting
3. Repetition
4. Slicing
5. Deleting
6. Finding the length
7. Multiple Data Types with tuples
8. Conversion of lists to tuples
9. Tuples in a Loop

1.Concatenation

To [Concatenation]() of Python Tuples, we will use plus operators(+).

EXAMPLE:

```
# Code for concatenating 2 tuples
tuple1 = (0, 1, 2, 3)
tuple2 = ('python', 'geek')
# Concatenating above two
print(tuple1 + tuple2)
```

Output:

```
(0, 1, 2, 3, 'python', 'geek')
```

2.Nesting of Python Tuples

A nested tuple in Python means a tuple inside another tuple.

EXAMPLE:

```
# Code for creating nested tuples
tuple1 = (0, 1, 2, 3)
tuple2 = ('python', 'geek')
tuple3 = (tuple1, tuple2)
print(tuple3)
```

Output :

```
((0, 1, 2, 3), ('python', 'geek'))
```

3. Repetition Python Tuples

We can create a tuple of multiple same elements from a single element in that tuple.

EXAMPLE:

```
# Code to create a tuple with repetition
tuple3 = ('python',)*3
print(tuple3)
```

Output:

```
('python', 'python', 'python')
```

4.Slicing Tuples in Python

Slicing a Python tuple means dividing a tuple into small tuples using the indexing method.

EXAMPLE:

```
# code to test slicing
tuple1 = (0 ,1, 2, 3)
print(tuple1[1:])
print(tuple1[::-1])
print(tuple1[2:4])
```

OUTPUT:

```
(1, 2, 3) (3, 2, 1, 0) (2, 3)
```

5. Deleting a Tuple in Python

In this example, we are deleting a tuple using ‘del’ keyword. The output will be in the form of error because after deleting the tuple, it will give a NameError.

6. Finding the Length of a Python Tuple

To find the length of a tuple, we can use Python’s len() function and pass the tuple as the parameter.

EXAMPLE:

```
# Code for printing the length of a tuple
tuple2 = ('python', 'geek') print(len(tuple2))
```

OUTPUT:2

7.Multiple Data Types With Tuple

Tuples in Python are heterogeneous in nature. This means tuples support elements with multiple datatypes.

EXAMPLE:

```
# tuple with different datatypes
tuple_obj = ("immutable",True,23)
print(tuple_obj)
```

OUTPUT:

```
('immutable', True, 23)
```

8.Converting a List to a Tuple

We can convert a list in Python to a tuple by using the tuple() constructor and passing the list as its parameters.

EXAMPLE:

```
# Code for converting a list and a string into a tuple
list1 = [0, 1, 2]
print(tuple(list1))
# string 'python'
print(tuple('python'))
```

Output:

Tuples take a single parameter which may be a list, string, set, or even a dictionary(only keys are taken as elements), and converts them to a tuple.

```
(0, 1 , 2)
(‘P’,’Y’,’T’,’H’,’O’,’N’)
```

9.Tuples in a Loop

We can also create a tuple with a single element in it using loops.

EXAMPLE:

```
# python code for creating tuples in a loop
tup = (‘SNS',)
# Number of time loop runs
n = 5
for i in range(int(n)):
        tup = (tup,)
        print(tup)
```

OUTPUT:?

THANKYOU

SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

19AMB302-FULL STACK AI

M.POORNIMA DEVI,AP/AIML

INDEXING AND SLICING IN PYTHON

1.Slicing and indexing are two fundamental concepts in Python.
2.They help you access specific elements in a sequence, such as a list, tuple or string.
3.By using these techniques, you can extract substrings from strings, filter lists, and extract columns from 2D lists, among other things.

Indexing in Python

1.Indexing is the process of accessing an element in a sequence using its position in the sequence (its index).
2.In Python, indexing starts from 0, which means the first element in a sequence is at position 0, the second element is at position 1, and so on.
3.To access an element in a sequence, you can use square brackets [] with the index of the element you want to access.

Let's consider the following example:

```
my_list = ['apple', 'banana', 'cherry', 'date']
 print(my_list[0]) # output: 'apple'
print(my_list[1])
# output: 'banana‘
```

In the above code, we have created a list called my_list and then used indexing to access the first and second elements in the list using their respective indices.

2.Slicing in Python

Slicing is the process of accessing a sub-sequence of a sequence by specifying a starting and ending index. In Python, you perform slicing using the colon : operator. The syntax for slicing is as follows:

EXAMPLE:

```
sequence[start_index:end_index]
```

where start_index is the index of the first element in the sub-sequence and end_index is the index of the last element in the sub-sequence (excluding the element at the end_index). To slice a sequence, you can use square brackets [] with the start and end indices separated by a colon.

For example:
my_list = ['apple', 'banana', 'cherry', 'date'] print(my_list[1:3]) # output: ['banana', 'cherry']
In the above code, we have used slicing to access a sub-sequence of my_list containing the second and third elements.
You can also omit either the start_index or the end_index in a slice to get all the elements from the beginning or end of the sequence.

For example:
my_list = ['apple', 'banana', 'cherry', 'date'] print(my_list[:2]) # output: ['apple', 'banana'] print(my_list[2:]) # output: ['cherry', 'date']
In the first line of the above code, we have used slicing to get all the elements from the beginning of my_list up to (but not including) the element at index 2. In the second line, we have used slicing to get all the elements from index 2 to the end of my_list.

Example 1: How to Extract Substrings from a String
Suppose we have a string representing a sentence, and we want to extract the first word from the sentence.
We can do this using indexing as follows:
sentence = "The quick brown fox jumps over the lazy dog"
first_word = sentence[:3] print(first_word) # output: "The“

In the above code, we have used indexing to extract the first three characters from the sentence string, which correspond to the first word. The [:3] syntax means that we are selecting all characters from the beginning of the string up to (but not including) the character at index 3.

We could also extract the second and third words from the sentence using slicing as follows:
second_word = sentence[4:9] third_word = sentence[10:15] print(second_word) # output: "quick" print(third_word) # output: "brown“

Example 2: How to Filter a List

Suppose we have a list of numbers and we want to extract all the odd numbers from the list. We can do this using slicing as follows:

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9] odd_numbers = numbers[::2] print(odd_numbers) # output: [1, 3, 5, 7, 9]

In the above code, we have used slicing to extract all the odd numbers from the numbers list. The ::2 slice means that we are selecting every other element starting from the first element, which correspond to the odd numbers in the list. Since we only want the odd numbers, we start with the first element (index 0) and then select every other element after that.

Example 3: How to Extract Columns from a 2D List
Suppose we have a 2D list representing a table of data, and we want to extract a particular column from the table.

We can do this using list comprehension and indexing as follows:

```
data = [[1, 2, 3], [4, 5, 6], [7, 8, 9]] column = [row[1] for row in
data] print(column) # output: [2, 5, 8]
column_0 = [row[0] for row in data] column_2 = [row[2] for row
in data] print(column_0) # output: [1, 4, 7] print(column_2) #
output: [3, 6, 9]
```

Example 4: How to Modify Parts of a List
Suppose we have a list of numbers and we want to modify the values of some of the elements in the list. We can do this using slicing as follows:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9] numbers[1:4] = [10, 20, 30]
print(numbers) # output: [1, 10, 20, 30, 5, 6, 7, 8, 9]
```

Example 4: How to Modify Parts of a List

Suppose we have a list of numbers and we want to modify the values of some of the elements in the list. We can do this using slicing as follows:

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9] numbers[1:4] = [10, 20, 30] print(numbers) # output: [1, 10, 20, 30, 5, 6, 7, 8, 9]

In the above code, we have used slicing to select a range of elements from the numbers list (indices 1 to 3), and then replaced these elements with a new list [10, 20, 30]. The result is that the elements at indices 1 to 3 in the numbers list have been replaced with the new values.

We could also insert new elements into the list using slicing as follows:

numbers[4:4] = [40, 50] print(numbers) # output: [1, 10, 20, 30, 40, 50, 5, 6, 7, 8, 9]

THANKYOU