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

A.Catherine,AP/AIML

PYTHON LIST SLICING

•In Python, list slicing is a common practice and it is the most used technique for programmers to solve efficient problems.

•Consider a Python list, in order to access a range of elements in a list, you need to slice a list. One way to do this is to use the simple slicing operator i.e. colon(:). With this operator, one can specify where to start the slicing, where to end, and specify the step.

• List slicing returns a new list from the existing list.

1.Python List Slicing Syntax

The format for list slicing is of Python List Slicing is as follows:

Lst[Initial : End : IndexJump] If *Lst* is a list, then the above expression returns the portion of the list from index *Initial* to index *End*, at a step size *IndexJump*.

2.Indexing in Python List

•Indexing is a technique for accessing the elements of a Python List. There are various ways by which we can access an element of a list.

Positive Indexes

In the case of Positive Indexing, the first element of the list has the index number 0, and the last element of the list has the index number N-1, where N is the total number of elements in the list (size of the list).

Index	0	1	2	3	4	5	6
Elements	50	70	30	20	90	10	50

Lst
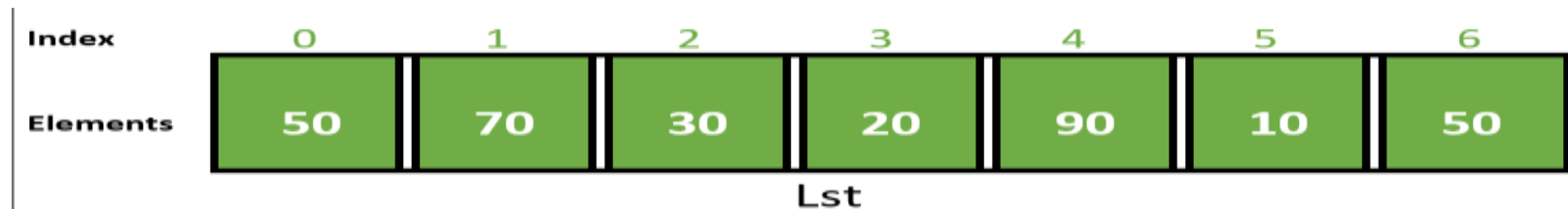
Example:
In this example, we will display a whole list using positive index slicing.
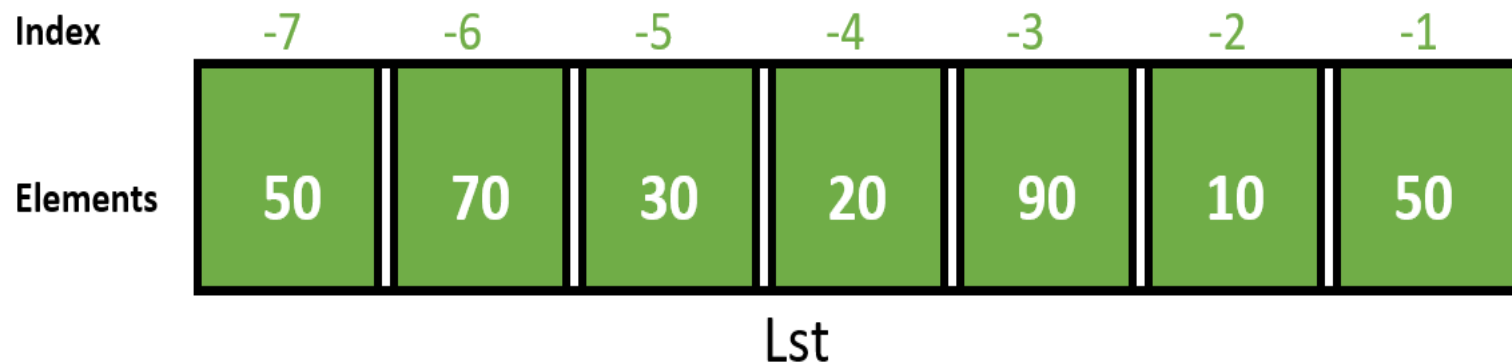# Initialize list
Lst = [50, 70, 30, 20, 90, 10, 50]
# Display list
print(Lst[::])
Output:
[50, 70, 30, 20, 90, 10, 50]

Negative Indexes
The below diagram illustrates a list along with its negative indexes. Index -1 represents the last element and -N represents the first element of the list, where N is the length of the list.

Example:

In this example, we will access the elements of a list using negative indexing.

\# Initialize list

Lst = [50, 70, 30, 20, 90, 10, 50]

\# Display list
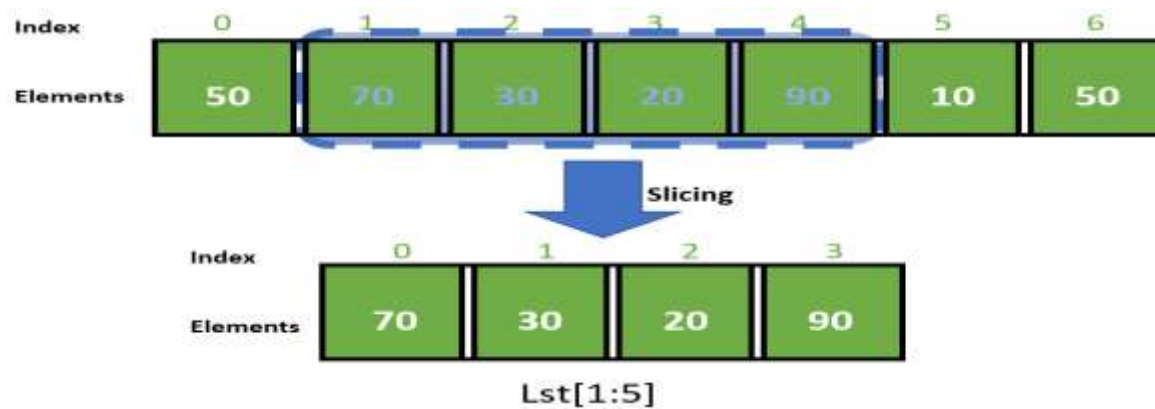
print(Lst[-7::1])

Output:

[50, 70, 30, 20, 90, 10, 50]

Slicing

As mentioned earlier list slicing in Python is a common practice and can be used both with positive indexes as well as negative indexes. The below diagram illustrates the technique of list slicing:

Example:

In this example, we will transform the above illustration into Python code.

```python
# Initialize list
Lst = [50, 70, 30, 20, 90, 10, 50]


# Display list
print(Lst[1:5])
```

Output:

[70, 30, 20, 90]

Examples of List Slicing in Python

Let us see some examples which depict the use of list slicing in Python.

Example 1: Leaving any argument like Initial, End, or IndexJump blank will lead to the use of default values i.e. 0 as Initial, length of the list as End, and 1 as IndexJump.

```python
# Initialize list
List = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# Show original list
print("Original List:\n", List)

print("\nSliced Lists: ")

# Display sliced list
print(List[3:9:2])

# Display sliced list
print(List[::2])

# Display sliced list
print(List[::])
```

Output:
Original List: [1, 2, 3, 4, 5, 6, 7, 8, 9] Sliced Lists: [4, 6, 8] [1, 3, 5, 7, 9] [1, 2, 3, 4, 5, 6, 7, 8, 9]

Example 2: A reversed list can be generated by using a negative integer as the IndexJump argument. Leaving the Initial and End as blank. We need to choose the Initial and End values according to a reversed list if the IndexJump value is negative.

```
# Initialize list
List = ['Geeks', 4, 'geeks !']

# Show original list
print("Original List:\n", List)

print("\nSliced Lists: ")

# Display sliced list
print(List[::-1])

# Display sliced list
print(List[::-3])

# Display sliced list
print(List[:1:-2])
```

Example 3: If some slicing expressions are made that do not make sense or are incomputable then empty lists are generated.

```
# Initialize list
List = [-999, 'G4G', 1706256, '^_^', 3.1496]
# Show original list
print("Original List:\n", List)
print("\nSliced Lists: ")
# Display sliced list
print(List[10::2])
# Display sliced list
print(List[1:1:1])
# Display sliced list
print(List[-1:-1:-1])
# Display sliced list
print(List[:0:])
```

Output:
Original List: [-999, 'G4G', 1706256, '^_^', 3.1496] Sliced Lists: [] [] [] []

Example 4: List slicing can be used to modify lists or even delete elements from a list.

```
# Initialize list
List = [-999, 'G4G', 1706256, 3.1496, '^_^']
# Show original list
print("Original List:\n", List)
print("\nSliced Lists: ")
# Modified List
List[2:4] = ['Geeks', 'for', 'Geeks', '!']
# Display sliced list
print(List)
# Modified List
List[:6] = []
# Display sliced list
print(List)
```

Output:
Original List: [-999, 'G4G', 1706256, 3.1496, '^_^'] Sliced Lists:
[-999, 'G4G', 'Geeks', 'for', 'Geeks', '!', '^_^'] ['^_^']

Example 5: By concatenating sliced lists, a new list can be created or even a pre-existing list can be modified.

```
# Initialize list
List = [1, 2, 3, 4, 5, 6, 7, 8, 9]
# Show original list
print("Original List:\n", List)
print("\nSliced Lists: ")
# Creating new List
newList = List[:3]+List[7:]
# Display sliced list
print(newList)
# Changing existing List
List = List[::2]+List[1::2]
# Display sliced list
print(List)
```

Output:
Original List: [1, 2, 3, 4, 5, 6, 7, 8, 9] Sliced Lists: [1, 2, 3, 8, 9] [1, 3, 5, 7, 9, 2, 4, 6, 8]

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

SLICING PANDAS DATAFRAME

•Slicing Pandas DataFrames is a powerful technique, allowing extraction of specific data subsets based on integer positions.

•In this article, let's understand examples showcasing row and column slicing, cell selection, and boolean conditions.

Slicing Pandas Dataframe

•With the help of Pandas, we can perform slicing in Dataframe.

•Slicing in pandas dataframes using iloc[] is a powerful technique in Python for extracting specific subsets of data.

• The iloc[] method allows you to locate and extract rows and columns based on their integer positions.

•To perform slicing with iloc[], you specify the row and column indices you want to include in your sliced dataframe.

• The syntax is similar to traditional array slicing, making it intuitive for Python users. For example, df.iloc[1:5, 2:4] extracts rows 2 to 5 and columns 3 to 4 from the dataframe.

Slicing a DataFrame in Pandas includes the following steps:
1.Create a DataFrame
2.Slice the DataFrame
Example:
import pandas as pd

# Initializing the nested list with Data set
player_list = [['M.S.Dhoni', 36, 75, 5428000],

['A.B.D Villers', 38, 74,

3428000],

['V.Kohli', 31, 70, 8428000],
['S.Smith', 34, 80, 4428000],
['C.Gayle', 40, 100, 4528000],
['J.Root', 33, 72, 7028000],
['K.Peterson', 42, 85, 2528000]]

# creating a pandas dataframe
df = pd.DataFrame(player_list, columns=['Name', 'Age',
'Weight', 'Salary'])
df # data frame before slicing

1. Slicing Using iloc

A. Slicing Rows in dataframe in python

B. # Slicing rows in data frame

C. df1 = df.iloc[0:4]

D.   # data frame after slicing

A. Df1

Output:

Name Age Weight Salary 0 M.S.Dhoni 36 75 5428000 1
A.B.D Villers 38 74 3428000 2 V.Kohli 31 70 8428000 3
S.Smith 34 80 4428000

B. Slicing Columns in dataframe in python

Slicing columnss in data frame

df1 = df.iloc[:, 0:2]

data frame after slicing

Df1

Output:

Name Age 0 M.S.Dhoni 36 1 A.B.D Villers 38 2 V.Kohli 31
3 S.Smith 34 4 C.Gayle 40 5 J.Root 33 6 K.Peterson 42

C. Selecting a Specific Cell in Dataframe in Python

```
specific_cell_value = df.iloc[2, 3] # Row 2, Column 3 (Salary)
print("Specific Cell Value:", specific_cell_value)
```

Output:

Specific Cell Value: 8428000

D. Using Boolean Conditions in Dataframe in Python

```
filtered_data = df[df['Age'] > 35].iloc[:, :] # Select rows where
Age is greater than 35
print("\nFiltered Data based on Age > 35:\n", filtered_data)
```

Output:

Filtered Data based on Age > 35: Name Age Weight Salary 0
M.S.Dhoni 36 75 5428000 1 A.B.D Villers 38 74 3428000 4
C.Gayle 40 100 4528000 6 K.Peterson 42 85 2528000

2. Slicing Using loc[]

We can also, implement slicing through loc there are some limitations:

•loc relies on labels, and if your DataFrame has custom labels, you need to be careful with how you specify them.

•If labels are integers, there might be confusion between using integer positions and actual labels.

For this, we need to set index as labels manually with following code:

df_custom = df.set_index('Name')
df_custom

Output:

Age Weight Salary Name M.S.Dhoni 36 75 5428000 A.B.D Villers 38 74 3428000 V.Kohli 31 70 8428000 S.Smith 34 80 4428000 C.Gayle 40 100 4528000 J.Root 33 72 7028000 K.Peterson 42 85 2528000

A. Slicing Rows in Dataframe in Python

sliced_rows_custom = df_custom.loc['A.B.D
    Villers':'S.Smith']

sliced_rows_custom

Output:

Age Weight Salary Name A.B.D Villers 38 74 3428000
V.Kohli 31 70 8428000 S.Smith 34 80 4428000

B. Selecting Specified cell in Dataframe in Python

specific_cell_value = df_custom.loc['V.Kohli', 'Salary']
print("\nValue of the Specific Cell (V.Kohli, Salary):",
specific_cell_value)

Output:

Value of the Specific Cell (V.Kohli, Salary): 8428000

Conclusion

In summary, both iloc[] and loc[]provide versatile slicing capabilities in Pandas. While iloc[] is integer-based, loc[] relies on labels, requiring careful consideration when working with custom indices or mixed data types.

THANKYOU