30-09-2024

Training Day- 10

INDEXING AND SLICING: Python provides powerful tools for accessing and manipulating elements of sequences like strings, lists, tuples, and more. These tools include **indexing** and **slicing**.

1. Indexing

Indexing is used to access individual elements of a sequence by their position. Indexing: Selecting a single element from an iterator

Key Points:

- Indexes start at 0 for the first element.
- Negative indexes count from the end of the sequence.
- Syntax: sequence[index]

Examples:

python
Copy code

```
# Indexing a string
my_string = "Python"
print(my_string[0])  # Output: P (first character)
print(my_string[-1]) # Output: n (last character)

# Indexing a list
my_list = [10, 20, 30, 40, 50]
print(my_list[2])    # Output: 30
print(my_list[-3])   # Output: 30
```

.

2. Slicing

Slicing extracts a portion (a subset) of a sequence using a range of indexes.

Syntax:

python
Copy code

```
sequence[start:stop:step]
```

- **start**: The starting index (inclusive, defaults to 0 if omitted).
- **stop**: The stopping index (exclusive).
- **step**: The step size (defaults to 1 if omitted).

LIST: A Collection of heterogeneous data type List is mutable We can change the elements in list:
Syntax:

[Comma separated elements]
L=[10,"CETPA",True,[2,3],2.5]

TUPLE: A Collection of heterogeneous data type tuple is immutable we can't change the elements
in tuple
Syntax:
(Comma separated elements)
t=(10,"CETPA",True,[2,3],2.5)

…….New Program…….
```
# L=[10,True,5.5]
# print(L)
# print(type(L))
```
…….New Program…….
```
# t=(10,True,5.5)
# print(t)
# print(type(t))
```

Indexing: Accessing Single Elements Of An Iterator

s="CETPA" #No of elements: 5
Indexing: 0 to len-1
indices for "CETPA": 0 to 4
Syntax: For str, list, tuple:
var_name[index]
…….New Program…….
```
# s="CETPA"
# print(s[0])
# print(s[1])
# print(s[4])
```
…….New Program…….
```
# s="Welcome"     #index: 0 to 6:
# print(s[10])       #IndexError: string index out of range
```

…….New Program…….
```
# L=[10,20,30,40,50]
# print(L[0])
# print(L[1])
# print(L[4])
```
…….New Program…….
```
# L=[10,20,30,40,50]
# print(L[0],L[1],L[4])
```
…….New Program…….
```
# t=(10,20,30,40,50)
# print(t[0])
# print(t[1])
# print(t[4])
```

IN CASE OF INDEXING, IF INDEX GOES OUT OF RANGE THEN WE GET? INDEXERROR

+ve indexing: 0 to len-1 or 0 to n-1 where n is the no of elements in the iterator

Python supports -ve indexing also

-ve indexing: -n to -1 or -length to -1 where n is the no of elements in the iterator

-ve indexing travels from right to left ie -1 to -n
+ve indexing travels from left to right ie 0 to n-1

…….New Program…….

```
# s="Welcome"     #length=7, index -7 to -1
# print(s[-1])
# print(s[-2])
# print(s[-7])
```

SLICING: When we want to access multiple elements from an iterator in some particular sequence then we use slicing

Syntax for Slicing:
var_name[lower_bound:upper_bound:step_size]
Result: Lower bound to upper bound (discarding upper bound) and adding step size in lower bound

Upper bound is discarded in python at most of the places.
Upper bound is discarded in slicing also

"""

…….New Program…….

```
# s="Welcome"     #index 0 to 6
# print(s[1:6:2])     #s[1] s[3] s[5]
```

…….New Program…….

```
# s="Welcome"     #index 0 to 6
# print(s[1:5:2])     #s[1] s[3]
```

…….New Program…….

```
# s="Welcome to CETPA"    #0 to 15
# print(s[0:10:3])        #s[0] s[3] s[6] s[9]
```

…….New Program…….

```
# s="Welcome to CETPA"    #0 to 15
# print(s[2:10:4])        #s[2] s[6]
```

…….New Program…….

```
# s="Welcome to CETPA"    #0 to 15
# print(s[3:10:2])        #s[3] s[5] s[7] s[9]
```

"""

You have a list of 8 customers, find the customers 1st, 3rd and 5th customer

…….New Program…….

```
# L=["Aditi","Umesh","Aman","Kajal","Roshni","Manpreet","Hemant","Binod"]
# print(L[0:5:2])     #L[0]  L[2]  L[4]
```

"""

Default Step Size: Always 1
Step size is optional to provide
Lower bound, upper bound and step size should be integers.
Step size can't be 0

…….New Program…….

```
# s="Welcome"
# print(s[2:5:1])         #s[2] s[3] s[4]
# print(s[2:5])       #s[2:5:1]       s[2] s[3] s[4]
```

…….New Program…….

```
# s="Welcome"
# print(s[2:5:0])   #ValueError: slice step cannot be zero
```

…….New Program…….

```
# s="Welcome"     #index 0 to 6
# print(s[10])       #IndexError
```

"""

In case of indexing, if index is out of range then there will be error.
But in case of slicing if bounds are out of range then there won't be any error

"""

…….New Program…….

```
# s="Welcome"     #index 0 to 6
# print(s[1:10:2])    #s[1] s[3] s[5]
```

In case of + step size:
Default lower bound: 0
Default upper bound: n ie len
"""

```
# #New Program
# s="Welcome"
# print(s[:5:2])      #s[0:5:2]    s[0] s[2] s[4]

# #New Program
# s="Welcome"   #Index 0 to 6
# print(s[2::2])      #s[2:7:2]    s[2] s[4] s[6]

# #New Program
# s="Welcome"   #Index 0 to 6
# print(s[::2])      #s[0:7:2]   s[0] s[2] s[4] s[6]
```

For +ve Step Size: We will travel from left to right
For +ve step size: Upper bound should be bigger than lower

bound to get the elements otherwise we will get empty value

…….New Program…….

```
# s="Welcome"
# print(s[1:5:2])     #s[1] s[3]
# print(s[5:1:2])     #Empty string: We are trying to travel
#                     # right to left with +ve step size: no output
```

…….New Program…….

```
# L=[10,20,30,40,50,60,70]    #index 0 to 6
# print(L[1:5:2])     #L[1] L[3]
# print(L[5:1:2])     #Empty list
```

…….New Program…….

```
# L=[10,20,30,40,50,60,70]    #index 0 to 6
# print(L[5:5:2])     #Empty list
# print(L[5:6:2])     #L[5]
```

…….New Program…….

```
# s="Welcome"
# print(s[20:40:2])       #Empty String
# print(s[0:40:2])        #s[0] s[2] s[4] s[6]
```

"""
For +ve Step Size: We will travel from left to right
Default lower bound:0
Default upper bound: n
If our bounds are traveling from right to left then empty value
at the output

For -ve Step Size: We will travel from right to left
Default lower bound:0
Default upper bound: n
If our bounds are traveling from left to right then empty value
at the output
Upper bound should be smaller than lower bound

"""

```
# #New Program
# s="Welcome"
# print(s[5:1:-1])    #s[5] s[4] s[3] s[2]
# print(s[1:5:-1])    #Empty string


# #New Program
# s="Welcome"         #0 to 6
# print(s[15:1:-1])    #  s[6] s[5] s[4] s[3] s[2]

# #New Program
# s="Welcome"         #0 to 6
# print(s[15:10:-1])    # Empty String
```

```
# #New Program
# print()        #print(end)

# #New Program
# s="Welcome"         #0 to 6
# print(s[-2:-5:-1])  #s[-2] s[-3] s[-4]

# #New Program
# s="Welcome"         #0 to 6
# print(s[-5:-2:-1])  #Empty String

# #New Program
# s="Welcome"         #0 to 6
# print(s[-5:-5:-1])  #Empty String

# #New Program
# s="Welcome"         #0 to 6
# print(s[-5:-5:-1])  #Empty String


"""
For +ve step size:
Default Lower Bound = 0
Default Upper Bound = n

For -step size
Default Lower Bound = -1
Default upper bound = (-n-1)
"Welcome" indexes -7 to -1
"""

# #New Program
# s="Welcome"         #0 to 6, -7 to -1
# print(s[-2::-1])  #s[-2:-8:-1]  s[-2] s[-3] s[-4] s[-5] s[-6] s[-7]

# #New Program
# s="Welcome"         #0 to 6, -7 to -1
# print(s[-2::-2])  #s[-2:-8:-2]  s[-2] s[-4] s[-6]

"""Print the alternate elements of a string in reverse direction"""
# #New Program
# s="Welcome"         #0 to 6, -7 to -1
# print(s[::-2])  #s[-1:-8:-2]  s[-1] s[-3] s[-5] s[-7]

# """Reverse  list: Was asked in Google Written Exam"""
# L=[10,20,30,40,50]
# print(L[::-1])
```

Default step size is always 1

```
# #New Program
# s="Welcome"
# print(s[-2:-4])     #s[-2:-4:1]

"""
If bounds are of different signs then firstly convert them
into common sign either +ve or -ve for better understanding
the output
"""
# #New Program
# s="Welcome"
# print(s[-7:4:1])    #s[0:4:1]   #s[-7:-3:1]
```