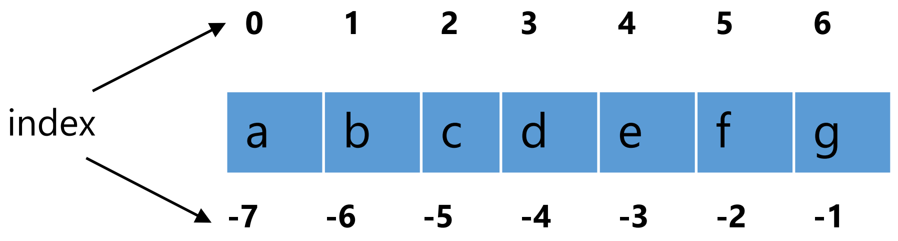
IT5001 Software Development Fundamentals

Simulation of Python's String Slicing

string[start : stop : step]

Rajendra Prasad Sirigina

String Slicing

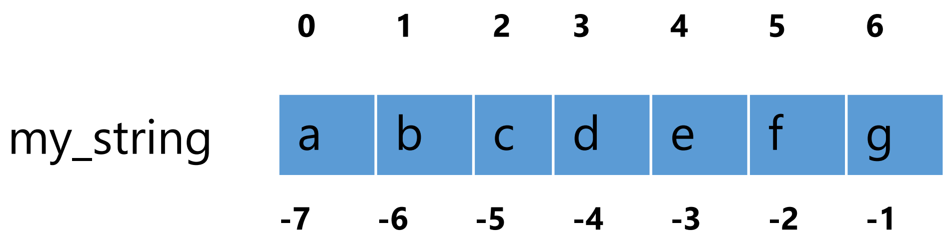


- my_str[$start$: $stop$: $step$]
  - ➢ Characters at the following indices are selected
    - ○ $start, start + step, start + 2 * step, \ldots.$
    - ○ Number of characters in the output $= max\left(0, \left\lceil \frac{stop - start}{step} \right\rceil\right)$
    - ○ The output includes start index and excludes end index
  - ➢ This formula is valid for both positive and negative step values
  - ➢ If $start == stop$, output is an empty string

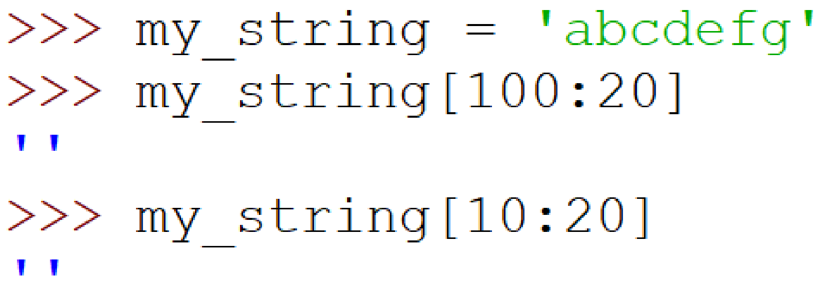
- Default value for step is 1

```
if step is None:
    step = 1
```

String Slicing (step > 0)



- Python's string slicing accepts start and end values beyond the length of the string.
  - ➢ my_string[100:20] won't produce *IndexError*

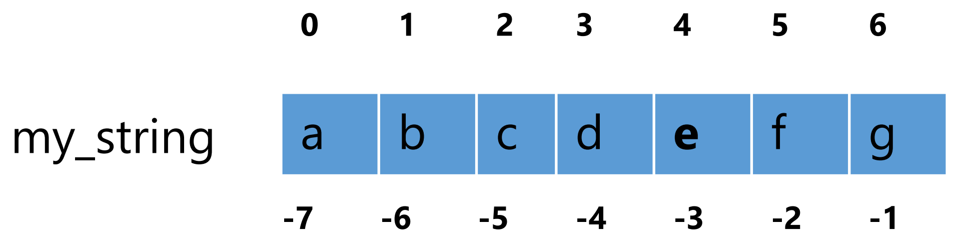


```
>>> my_string = 'abcdefg'
>>> my_string[100:20]
''
>>> my_string[10:20]
''
```

  - ➢ It is managed by clamping start and end values

start = max(0, min(start, length of string))
end = max(0, min(end, length of string))

String Slicing: step > 0



- How are negative values for start, end, and step lengths handled?
    - ➢ Negative indices are converted to positive indices

```
>>> my_string = 'abcdefg'
>>> my_string[-4:-1:1]
'def'
```

start = -4+7

end = -6+7

```
>>> my_string[3:6:1]
'def'
```

Length of string = 7

Pseudocode: step > 0

```
if step > 0:
    if start is None:  # default value for start
        start = 0
    else if start < 0: # negative indices to positive indices
        start = start  + length of string

    if end is None: # default value for end
        end = length of string
    else if end < 0: # negative indices to positive indices
        end = end + length of string

    # start and end clamped between 0 and length of string (to avoid $IndexError$)
    start = max(0, min(start, length of string))
    end = max(0, min(end, length of string))
```

String Slicing: step < 0

0	**1**	**2**	**3**	**4**	**5**	**6**
a	b	c	d	e	f	g
-7	**-6**	**-5**	**-4**	**-3**	**-2**	**-1**

- Negative step value helps in reversing the string

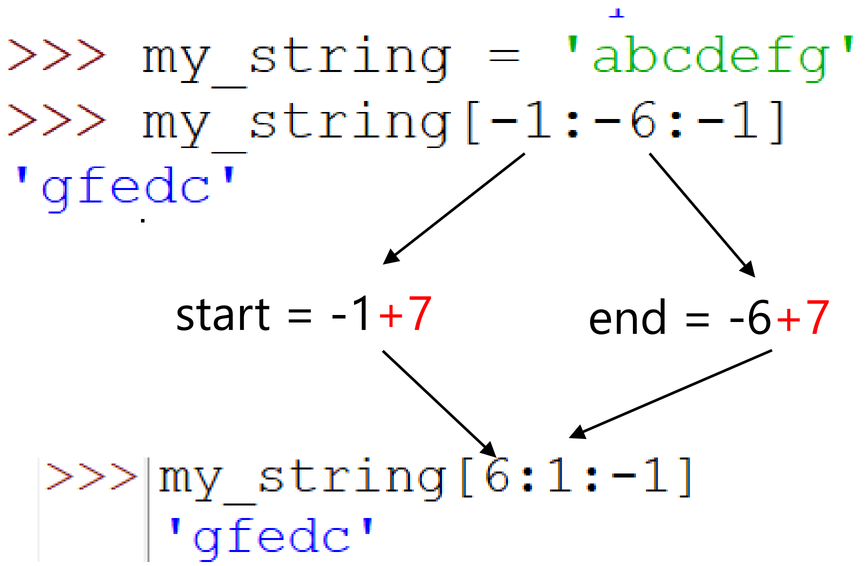
```
>>> my_string = 'abcdefg'
>>> my_string[::-1]
'gfedcba'
```

- If start and end are *None*, their default values are set as
  - start= Length of string-1 = 6
  - end = - (Length of string+1) = -8 #It will be clamped to legal range (see slide 8)

String Slicing: step < 0

0	1	2	3	4	5	6
a	b	c	d	e	f	g
-7	-6	-5	-4	-3	-2	-1

- How are negative start and end values handled?
  - ➢ add length of string



```
>>> my_string = 'abcdefg'
>>> my_string[-1:-6:-1]
'gfedc'
```

start = -1+7

end = -6+7

```
>>> my_string[6:1:-1]
'gfedc'
```

Length of string = 7

Pseudocode: step < 0

```
if step < 0:
    if start is None: # default value for start
        start = length of string – 1
    else if start < 0: # negative indices to positive indices
        start = length of string + start
    if end is None:  #default index for end
        end = - (length of string  +1)
    else if end < 0: #negative to positive indices
        end = length of string + end
    #Clamping the start and end values if they are out of range
    start = max(-1, min(start, length of string -1))
    end = max(-1, min(end, length of string-1))
```

Finally String Slicing

# After start, step and end values are revised as shown in slide 2, 5 and slide 8, perform string slicing

```
sliced_seq = ''   #initialize an empty string

for i in range(start, end, step):
        sliced_seq = sliced_seq + my_string[i]
```