TECHNO INDIA
SILIGURI INSTITUTE
OF TECHNOLOGY
A Satyam Roychowdhury Initiative

Strings

Sequence Datatype

Python String

- Python string—an ordered collection of characters used to store and represent text-based information.
- From a functional perspective, strings can be used to represent just about anything that can be encoded as text: symbols and words (e.g., your name), contents of text files loaded into memory, Internet addresses, Python programs, and so on.
- They can also be used to hold the absolute binary values of bytes, and multibyte Unicode text used in internationalized programs.
- Python strings are categorized as immutable sequences
- strings support expression operations such as concatenation (combining strings), slicing (extracting sections), indexing (fetching by offset), and so on.
- Python also provides a set of string methods that implement common string-specific tasks, as well as modules for more advanced text-processing tasks such as pattern matching.

String Literals

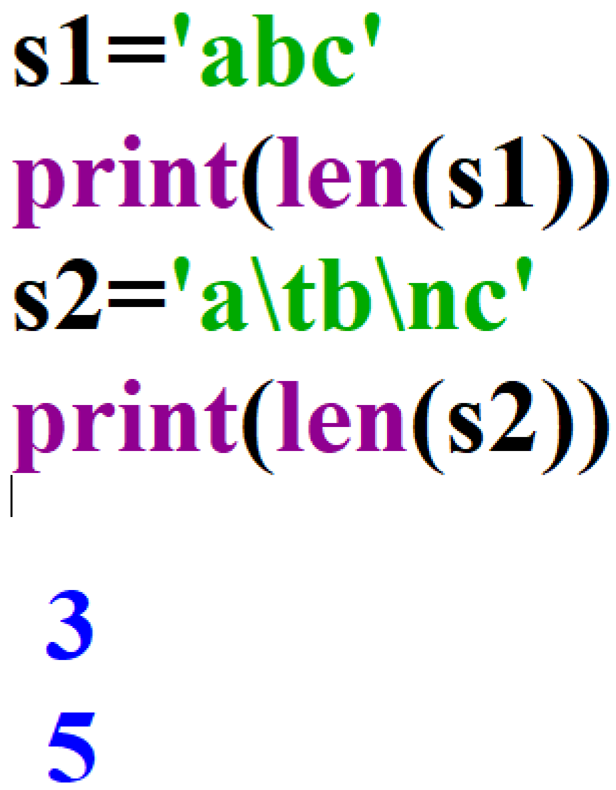
- Many ways to write string in Python code

```
s1='MCA'
print(s1)
s2="MCA"
print(s2)
s3="Student's"
print(s3)
s4='student"s'
print(s4)
```

```
s5='''this is
for multiline
string'''
print(s5)
s6='''this is\nfor multiline\nstring'''
print(s6)
s='a\tb\nc'
print(s)
s7=r'c:\my drive\test'
print(s7)
```

String Literals

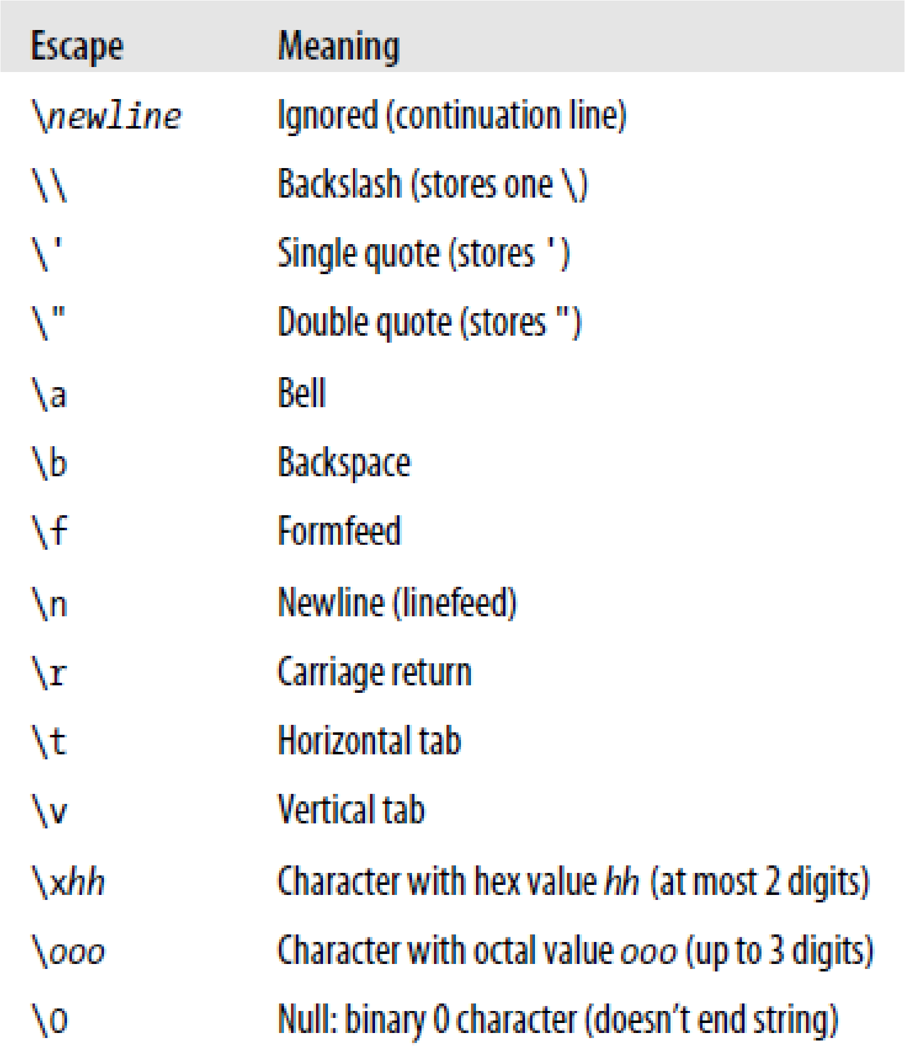
- Single and double quoted strings are the same.
- Escape sequences represent special bytes.
- backslashes are used to introduce special byte coding known as escape sequences.
- Escape sequences let us embed byte codes in strings that cannot easily be typed on a keyboard.



```
s1='abc'
print(len(s1))
s2='a\tb\nc'
print(len(s2))
```

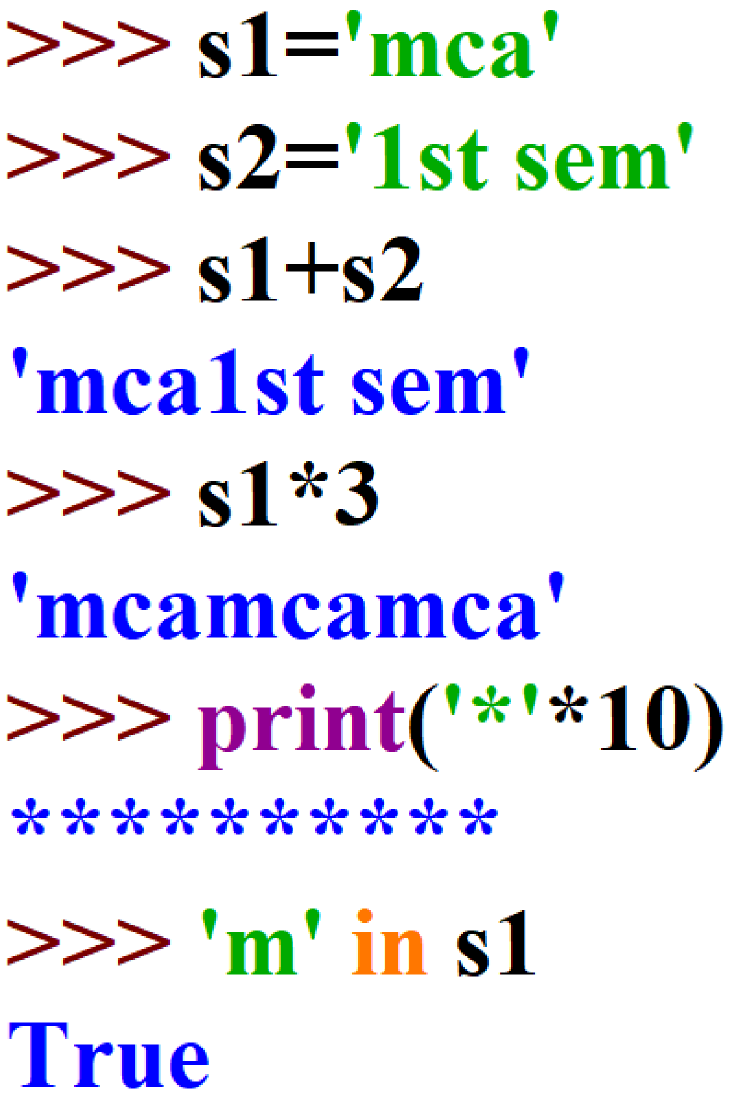
```
3
5
```

String Literals



Escape	Meaning
\newline	Ignored (continuation line)
\\	Backslash (stores one \)
\'	Single quote (stores ')
\"	Double quote (stores ")
\a	Bell
\b	Backspace
\f	Formfeed
\n	Newline (linefeed)
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\xhh	Character with hex value hh (at most 2 digits)
\ooo	Character with octal value ooo (up to 3 digits)
\0	Null: binary 0 character (doesn't end string)

Basic Operation

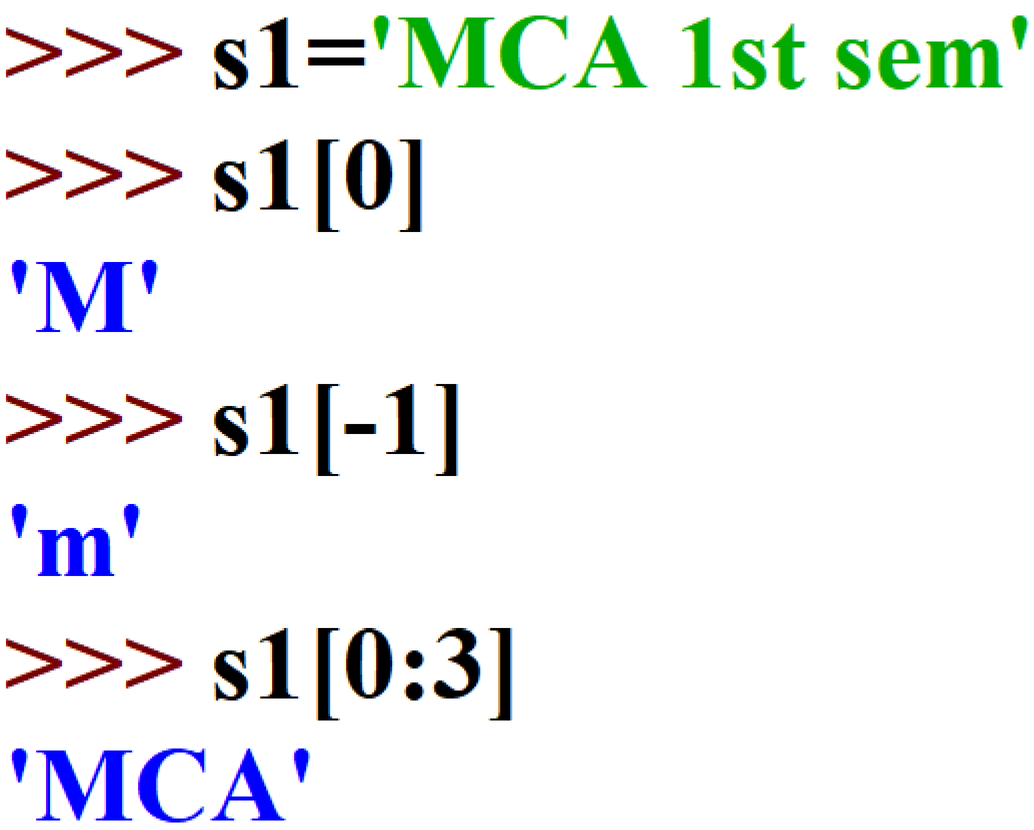


```
>>> s1='mca'
>>> s2='1st sem'
>>> s1+s2
'mca1st sem'
>>> s1*3
'mcamcamca'
>>> print('*'*10)
**********

>>> 'm' in s1
True
```

Indexing and Slicing

- In Python, characters in a string are fetched by indexing—providing the numeric offset of the desired component in square brackets after the string.
- You get back the one-character string at the specified position.
- a negative offset is added to the length of a string to derive a positive offset. You can also think of negative offsets as counting backward from the end.



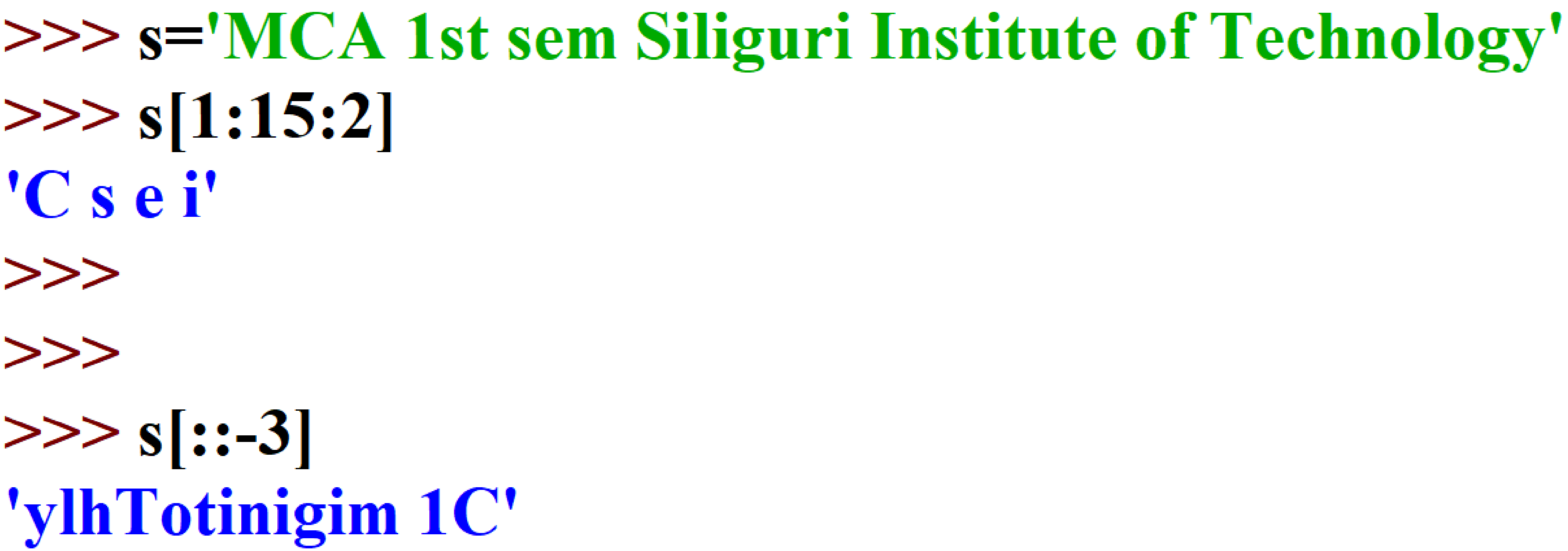
```
>>> s1='MCA 1st sem'
>>> s1[0]
'M'
>>> s1[-1]
'm'
>>> s1[0:3]
'MCA'
```

Indexing and Slicing

- Indexing (S[i]) fetches components at offsets:
  - The first item is at offset 0.
  - Negative indexes mean to count backward from the end or right.
  - S[0] fetches the first item.
  - S[−2] fetches the second item from the end (like S[len(S)−2]).
- Slicing (S[i:j]) extracts contiguous sections of sequences:
  - The upper bound is noninclusive.
  - Slice boundaries default to 0 and the sequence length, if omitted.
  - S[1:3] fetches items at offsets 1 up to but not including 3.
  - S[1:] fetches items at offset 1 through the end (the sequence length).

Extended Slicing

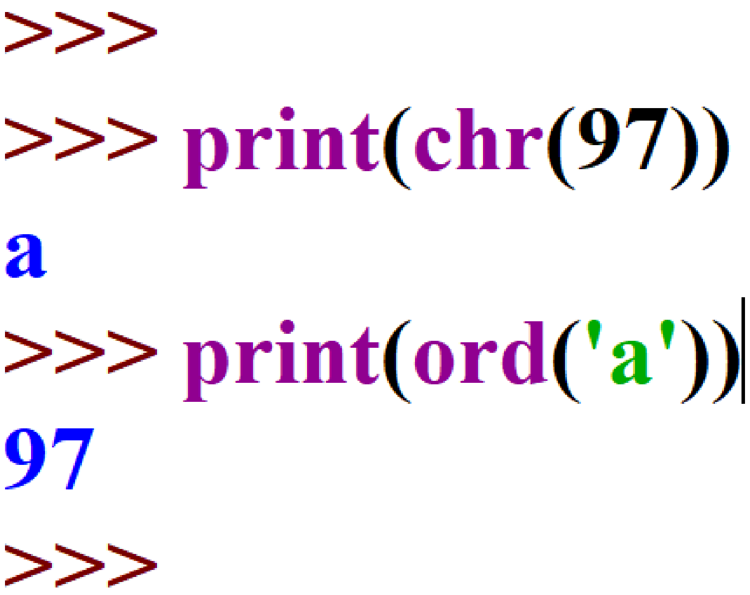
- The full-blown form of a slice is now $X[i:j:k]$, which means “extract all the items in X, from offset i through $j-1$, by k.” The third limit, k, defaults to 1



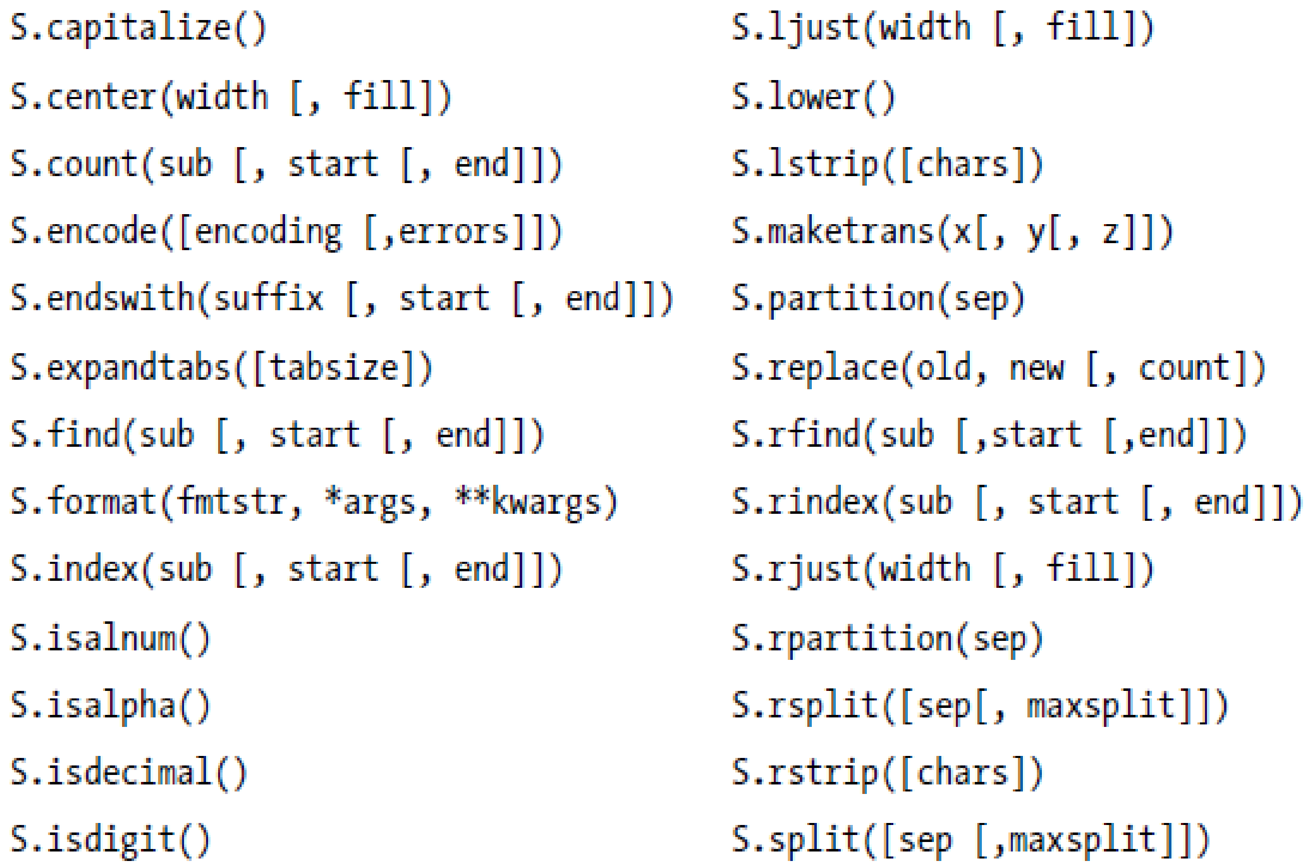
```
>>> s='MCA 1st sem Siliguri Institute of Technology'
>>> s[1:15:2]
'C s e i'
>>>
>>>
>>> s[::-3]
'ylhTotinigim 1C'
```

Character Code Conversion

- Convert a single character to its underlying ASCII integer code by passing it to the built-in ord function—this returns the actual binary value of the corresponding byte in memory.
- The chr function performs the inverse operation, taking an ASCII integer code and converting it to the corresponding character:



```
>>>
>>> print(chr(97))
a
>>> print(ord('a'))
97
>>>
```



```
S.capitalize()
S.center(width [, fill])
S.count(sub [, start [, end]])
S.encode([encoding [,errors]])
S.endswith(suffix [, start [, end]])
S.expandtabs([tabsize])
S.find(sub [, start [, end]])
S.format(fmtstr, *args, **kwargs)
S.index(sub [, start [, end]])
S.isalnum()
S.isalpha()
S.isdecimal()
S.isdigit()
S.ljust(width [, fill])
S.lower()
S.lstrip([chars])
S.maketrans(x[, y[, z]])
S.partition(sep)
S.replace(old, new [, count])
S.rfind(sub [,start [,end]])
S.rindex(sub [, start [, end]])
S.rjust(width [, fill])
S.rpartition(sep)
S.rsplit([sep[, maxsplit]])
S.rstrip([chars])
S.split([sep [,maxsplit]])
```