PYTHON PROGRAMMING
(20MCA21C)

UNIT - III

Strings and Lists

FACULTY:

Dr. R. A. Roseline, M.Sc., M.Phil., Ph.D.,

Associate Professor and Head,
Post Graduate and Research Department of Computer Applications,
Government Arts College (Autonomous), Coimbatore - 641 018.

STRINGS

STRINGS

- Strings are amongst the most popular types in Python. We can create them simply by enclosing characters in quotes. Python treats single quotes the same as double quotes. Creating strings is as simple as assigning a value to a variable. For example −

```
var1 = 'MCA,GAC!'

var2 = "Python Programming"
```

- Accessing Values in Strings.:Python does not support a character type; these are treated as strings of length one, thus also considered a substring.To access substrings, use the square brackets for slicing along with the index or indices to obtain your substring. For example −

```
print "var1[0]: ", var1[0]

print "var2[1:5]: ", var2[1:5]
```

- When the above code is executed, it produces the following result −

```
var1[0]:  M

var2[0:6]:  Python
```

Escape Characters

Following table is a list of escape or non-printable characters that can be represented with backslash notation. An escape character gets interpreted; in a single quoted as wellas double quoted strings.

Backslash notation	Hexadecimal character	Description
\a	0x07	Bell or alert
\b	0x08	Backspace
\cx		Control-x
\C-x		Control-x
\e	0x1b	Escape
\f	0x0c	Formfeed
\M-\C-x		Meta-Control-x
\n	0x0a	Newline
\nnn		Octal notation, where n is in the range 0.7
\r	0x0d	Carriage return
\s	0x20	Space
\t	0x09	Tab
\v	0x0b	Vertical tab
\x		Character x
\xnn		Hexadecimal notation, where n is in the range 0.9, a.f, or A.F

String Special Operators

Assume string variable a holds 'Hello' and variable b holds 'Python', then

Operator	Description	Example
+	Concatenation - Adds values on either side of the operator	a + b will give HelloPython
	Repetition - Creates new strings, concatenating multiple copies of the same string	a*2 will give - HelloHello
[]	Slice - Gives the character from the given index	a[1] will give e
[:]	Range Slice - Gives the characters from the given range	a[1:4] will give ell
in	Membership - Returns true if a character exists in the given string	H in a will give 1
not in	Membership - Returns true if a character does not exist in the given string	M not in a will give 1
r/R	Raw String - Suppresses actual meaning of Escape characters. The syntax for raw strings is exactly the same as for normal strings with the exception of the raw string operator, the letter "r," which precedes the quotation marks. The "r" can be lowercase (r) or uppercase (R) and must be placed immediately preceding the first quote mark.	print r'\n' prints \n and print R'\n'prints \n
%	Format - Performs String formatting	See at next section

Triple Quotes

- Python's triple quotes comes to the rescue by allowing strings to span multiple lines, including verbatim NEWLINEs, TABs, and any other special characters.The syntax for triple quotes consists of three consecutive single or double quotes.

new_str = "“”WELCOME TO

DEPT of COMPUTER APPLICATIONS

GAC,CBE-18”””

Output

>>>print new_str

WELCOME TO

DEPT of COMPUTER APPLICATIONS

GAC,CBE-18

Built-in String Methods

- Python includes the following built-in methods to manipulate strings

S. No	Methods with Description
1	capitalize() Capitalizes first letter of string
2	center(width, fillchar) Returns a space-padded string with the original string centered to a total of width columns.
3	count(str, beg= 0,end=len(string)) Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given.
4	decode(encoding='UTF-8',errors='strict') Decodes the string using the codec registered for encoding. encoding defaults to the default string encoding.
5	encode(encoding='UTF-8',errors='strict') Returns encoded string version of string; on error, default is to raise a ValueError unless errors is given with 'ignore' or 'replace'.
6	endswith(suffix, beg=0, end=len(string)) Determines if string or a substring of string (if starting index beg and ending index end are given) ends with suffix; returns true if so and false otherwise.
7	expandtabs(tabsize=8) Expands tabs in string to multiple spaces; defaults to 8 spaces per tab if tabsize not provided.
8	find(str, beg=0 end=len(string)) Determine if str occurs in string or in a substring of string if starting index beg and ending index end are given returns index if found and -1 otherwise.
9	index(str, beg=0, end=len(string)) Same as find(), but raises an exception if str not found.
10	isalnum() Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise.

11	isalpha() Returns true if string has at least 1 character and all characters are alphabetic and false otherwise.
12	isdigit() Returns true if string contains only digits and false otherwise.
13	islower() Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise.
14	isnumeric() Returns true if a unicode string contains only numeric characters and false otherwise.
15	isspace() Returns true if string contains only whitespace characters and false otherwise.

16	istitle() Returns true if string is properly "titlecased" and false otherwise.
17	isupper() Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise.
18	join(seq) Merges (concatenates) the string representations of elements in sequence seq into a string, with separator string.
19	len(string) Returns the length of the string
20	ljust(width[, fillchar]) Returns a space-padded string with the original string left-justified to a total of width columns.

21	lower() Converts all uppercase letters in string to lowercase.
22	lstrip() Removes all leading whitespace in string.
23	maketrans() Returns a translation table to be used in translate function.
24	max(str) Returns the max alphabetical character from the string str.
25	min(str) Returns the min alphabetical character from the string str.

26	replace(old, new [, max]) Replaces all occurrences of old in string with new or at most max occurrences if max given.
27	rfind(str, beg=0,end=len(string)) Same as find(), but search backwards in string.
28	rindex(str, beg=0, end=len(string)) Same as index(), but search backwards in string.
29	rjust(width,[, fillchar]) Returns a space-padded string with the original string right-justified to a total of width columns.
30	rstrip() Removes all trailing whitespace of string.

31	split(str="", num=string.count(str)) Splits string according to delimiter str (space if not provided) and returns list of substrings; split into at most num substrings if given.
32	splitlines(num=string.count('\n')) Splits string at all (or num) NEWLINEs and returns a list of each line with NEWLINEs removed.
33	startswith(str, beg=0,end=len(string)) Determines if string or a substring of string (if starting index beg and ending index end are given) starts with substring str; returns true if so and false otherwise.
34	strip([chars]) Performs both lstrip() and rstrip() on string.
35	swapcase() Inverts case for all letters in string.

36	title() Returns "titlecased" version of string, that is, all words begin with uppercase and the rest are lowercase.
37	translate(table, deletechars="") Translates string according to translation table str(256 chars), removing those in the del string.
38	upper() Converts lowercase letters in string to uppercase.
39	zfill (width) Returns original string leftpadded with zeros to a total of width characters; intended for numbers, zfill() retains any sign given (less one zero).
40	isdecimal() Returns true if a unicode string contains only decimal characters and false otherwise.

String Slicing in Python

Python slicing is about obtaining a sub-string from the given string by slicing it respectively from start to end.
Python slicing can be done in two ways.
•slice() Constructor
•Extending Indexing

slice() Constructor
The `slice()` constructor creates a slice object representing the set of indices specified by range(start, stop, step).
Syntax:
•slice(stop)
•slice(start, stop, step)
Parameters:
start: Starting index where the slicing of object starts.
stop: Ending index where the slicing of object stops.

step: *It is an optional argument that determines the increment between each index for slicing.*

Return Type: *Returns a sliced object containing elements in the given range only.*

Python program to demonstrate
String slicing

```
String ='MCA GACCBE'
# Using slice constructor
s1 = slice(3)
s2 = slice(1, 5, 2)
s3 = slice(-1, -5, -2)
print("String slicing")
print(String[s1])
print(String[s2])
print(String[s3])
```

output

String slicing
MCA
C
EC

0	1	2	3	4	5	6	7	8	9
M	C	A		G	A	C	C	B	E
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

LISTS

LISTS

- List is a data type that is mutable.
- Mutability is the ability for certain types of data to be changed without entirely recreating it. This is important for Python to run programs both quickly and efficiently.
- Many types in Python are immutable. Integers, floats, strings, and (as you'll learn later in this course) tuples are all immutable.
- Once one of these objects is created, it can't be modified, unless you reassign the object to a new value.

Create a List in Python

- Lets see how to create a list in Python. To create a list all you have to do is to place the items inside a square bracket [] separated by comma ,.
- # list of floats
- numlist = [10.2, 99.9,12.0]
- # list of int, float and strings
- mylist = [10.17, 78, “PYTHON", 100,]
- # an empty list
- nlist = []
-

List Operations

Python Expression	Results	Description
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership
for x in [1, 2, 3]: print x,	1 2 3	Iteration

Index in Lists

```
# A list of numbers
numbers = [1, 2, 3, 100, 200, 300]
print(numbers[0])
print(numbers[5])
```

Output

1

300

- **Negative Index to access the list items from the end**
- Python allows you to use negative indexes. The idea behind this to allow you to
- access the list elements starting from the end.
- For example an index of -1 would access the last element of the list, -2 second last, -3 third last and so on.
- 3.1 Example of Negative indexes in Python

```
# a list of strings
my_list = [“GACa, “CBEa, “MCAa]
print(my_list[-1])
print(my_list[-3])
```

- Output:
- MCA
- GAC

Built in List Functions

Sr. No	Function with Description
1	cmp(list1, list2)Compares elements of both lists.
2	len(list)Gives the total length of the list.
3	max(list)Returns item from the list with max value.
4	min(list)Returns item from the list with min value.
5	list(seq)Converts a tuple into list.

LIST METHODS

S.No	Methods with Description
1	list.append(obj)Appends object obj to list
2	list.count(obj)Returns count of how many times obj occurs in list
3	list.extend(seq)Appends the contents of seq to list
4	list.index(obj)Returns the lowest index in list that obj appears
5	list.insert(index, obj)Inserts object obj into list at offset index
6	list.pop(obj=list[-1])Removes and returns last object or obj from list
7	list.remove(obj)Removes object obj from list
8	list.reverse()Reverses objects of list in place
9	list.sort([func])Sorts objects of list, use compare func if given

Deleting Elements from Lists

Deleting elements using remove(), pop() and clear() methods
remove(item): Removes specified item from list.
pop(index): Removes the element from the given index.
pop(): Removes the last element.
clear(): Removes all the elements from the list.

```
# list of char
ch_list = ['A', ‘E', ‘I', ‘O', ‘U']
# Deleting the element with value 'B
'ch_list.remove(‘O')

print(ch_list)

ch_list.pop(1)

print(ch_list)

# Deleting all the elements
ch_list.clear()

print(ch_list)
```

Thank you

The Content in this Material are from the Textbooks and Reference books given in the Syllabus