SCMA 249

Computer Programming in Actuarial Science

1/2022

LAB 4
String, List

LAB 4
String, List

Strings are a data type in Python for dealing with text. Python has a number of powerful features for manipulating strings.

4.1 String

4.1.1 Basics

4.1.1 Create a string

Aforementioned in some previous sections, a string is created by enclosing some specific text in quotes. Either single quotes, ', or double quotes, ", can be used to create a string. If we want to create a multi-line text, we can use a triple-quote, """.

Example 4.1

4.1.1 `txt1 = 'Actuarial Science'`

4.1.2 `txt2 = "Computer Programming"`

4.1.3 `txt3 = """I wanna create a very long string in Python by enclosing it in triple-quote. It will be spread across multiple lines."""`

4.1.4 `txt4 = 'I\'m a student in an actuarial science program.'`

4.1.2 Length

The **`len()`** function is used to get the length of a string.

Example 4.2

Find the length of strings given in Example 4.1.

4.2.1 **`len(txt1)`**	Answer 17
4.2.2 **`len(txt2)`**	Answer 19
4.2.3 **`len(txt3)`**	Answer 117
4.2.4 **`len(txt4)`**	Answer 46

4.1.2 Concatenation and repetition

To concatenate strings, we use the operator +, called **concatenation**. If we want to repeat a string a certain number of times, we can use the operator *.

Example 4.3

4.3.1 `'SCMA' + ' 249'` Answer 'SCMA249'

4.3.2 `'SCMA' + ' 249' + ' Semester 1'` Answer 'SCMA249Semester1'

4.3.3 `'A' * 10` Answer 'AAAAAAAAAA'

4.3.4 `'B' + 'K'*2` Answer 'BKK'

Example 4.4

The **+** operator can be used to build up a string, piece by piece, analogously to the way we built up counts and sums. Here is an example that repeatedly asks the user to enter ID card number and builds up a string consisting of their ID card number.

```
print('Please enter your ID card number.')
#create an empty string
id_no = ''
for i in range(13):
        txt = 'Digit' + str(i+1) + ':'
        d = input(txt)
        id_no += d
print(id_no)
```

4.1.3 Indexing

Since strings in Python are arrays of bytes representing unicode characters, we can select particular character from a string by using index. In Python, we use square brackets, [ ] enclosing with an specific index number.

Each of a string's characters also correspond to an index number. Any symbol or punctuation mark, such as *#$&.;?, including a space is also a character and would be associated with its own index number. When we straightforward index the characters in a string, the index number is started from 0 to **`len(s)-1`**. We can also count the index number backward from the end of the string, in this case, the index number is the negative number starting from -1 to **`-len(s)`**. Table 4.1 shows the example of indexing.

Table 4.1 Example of string indexing in Python

	S	C	M	A		2	4	9
Positive Index	0	1	2	3	4	5	6	7
Negative Index	-8	-7	-6	-5	-4	-3	-2	-1

Example 4.5

Suppose we have a string **s** = 'EXAMPLE 4.5'.

4.5.1 `s[0]` Answer 'E'

4.5.2 `s[-11]` Answer 'E'

4.5.3 `s[-len(s)]` Answer 'E'

4.5.4 To get the character M in **s**, the code is s[3] , s[-8].

4.5.5 To get the character 4 in **s**, the code is s[8] , s[-3].

4.1.4 Slices

We can select pick out part of a string (more than one character) by giving a range of index, called **slice**. The basic syntax is

```
string_name[strating location:ending location + 1]
```

Note that the **slice** has the same property as **range** function in the sense that it doesn't include the ending location.

Example 4.6

Suppose we have a string **s1** = '0123456789'.

Code	Result	Description
`s1[2:6]`	'2345'	pick the characters at indices 2 to 5
`s1[:3]`	'012'	pick the first three characters
`s1[3:]`	'3456789'	pick characters from index 3 to the end
`s1[-3:]`	'789'	pick the last three characters
`s1[:]`	'0123456789'	
`s1[0:]`	'0123456789'	
`s1[:len(s1)]`	'0123456789'	
s1[5:13]	'56789'	pick the characters at indices 5 to 12
s1[-7:-3]	'3456'	slice by using the negative indices

There is an **optional third argument**, just like in the **range** statement, that can specify the step. The basic syntax is

```
string_name[strating location:ending location + 1:step size]
```

Example 4.7

Code	Result	Description
`s1[2:10:2]`	'2468'	pick the characters at indices 2 to 9 by two
`s1[::2]`	'02468'	pick the characters with the **even** indices
s1[1::2]	'13579'	pick the characters with the **odd** indices
`s1[::-1]`	'9876543210'	negative step reverses the string
`s1[::-2]`	'97531'	negative reverses the string by two

Tips

Changing individual characters of a string

Suppose we have a string **`name`** = `'Suntree'`. We want to change the character at index 4 of **`name`** to `'a'`. The code is shown below.

```
name = name[:4] + 'a' + name[5:]
```

Combine string with another data type

Another note is that Python does not do implicit string conversion. If you try to concatenate a string with a non-string type, Python will raise a TypeError:

```
>>> 'SCMA' + 249
Traceback (most recent call last):
  File "<pyshell#73>", line 1, in <module>
    'SCMA' + 249
TypeError: Can't convert 'int' object to str implicitly
>>> 'SCMA' + str(249)
'SCMA249'
```

4.1.5 String methods

Strings come with a ton of methods, functions that return information about the string or return a new string that is a modified version of the original. Example 4.8 provides some of the most useful ones.

Example 4.8

Let **s** be a string `'   I love SCMA 249.   '`.

4.8.1 **`lower()`**: returns a string with every letter of the original in **lowercase**.

Code: **`s1 = s.lower()`**

Result: ' i love scma 249. '

4.8.2 **`upper()`**: returns a string with every letter of the original in **uppercase**.

Code: **`s2 = s.upper()`**

Result: ' I LOVE SCMA 249. '

delete the unneccessary blanks

4.8.3 **`strip()`**: returns a copy of the string in which all chars have been stripped from the beginning and the end of the string (default whitespace characters).

Code: **`s3 = s.strip()`**

Result: 'I love SCMA 249.'

4.8.4 Code: **`s4 = s.strip().upper()`**

Result: 'I LOVE SCMA 249.'

4.8.5 **`split()`**: automatically cuts out leading and trailing whitespace, as well as consecutive whitespace.

Code: **`s.split()`**

Result: ['I', 'love', 'SCMA', '249']

4.8.6 **`count(x)`**: counts the number of occurrences of **x** in the string.

Code: **`s.count(' ')`**

Result: 13

4.8.7 **`index(x)`**: returns the location of the first occurrence of **x** in the string.

Code: **`s.index(' ')`**

Result: 0

isupper()
islower()

4.8.8 **`isalpha()`**: returns True if every character of the string is a letter.

Code: **`s.isalpha()`**

Result: False

4.8.9 Code: **`s[-4:-2].isalpha()`**

Result: False

4.8.10 Code: **`s[-9:-6].isalpha()`**

Result: True

Example 4.9

You can use **isalpha** in if conditions.

```
if s[0].isalpha():
     print('Your string starts with a letter')
if not s.isalpha():
     print('Your string contains a non-letter.')
```

Note If you try to find the index of something that is not in a string, Python will raise an error. It is recommended to check first. See the example below.

```
>>> s5='SCMA'
>>> s5.index('I')
Traceback (most recent call last):
  File "<pyshell#54>", line 1, in <module>
    s5.index('I')
ValueError: substring not found
>>> if 'I' in s5:
        location = s5.index('I')
```

4.1.6 Escape characters

Sometimes, we have to include some special character, called escape characters, in a string. The followings are some useful examples of escape characters.

Example 4.10

- `\n`   used to advance to the next line

  Code: `print('SCMA\n249')`

  Result: SCMA
  249

- `\'`   insert apostrophes into strings enclosing with single quotes

  Code: `'I don\'t want to miss any class of SCMA 249'`

  Result: **"I don't want to miss any class of SCMA 249"**

- `\"`   analogous to `\'`
- `\t`   the tab character   8 char per tab

  Code: `print('SCMA\t\t\t249')`

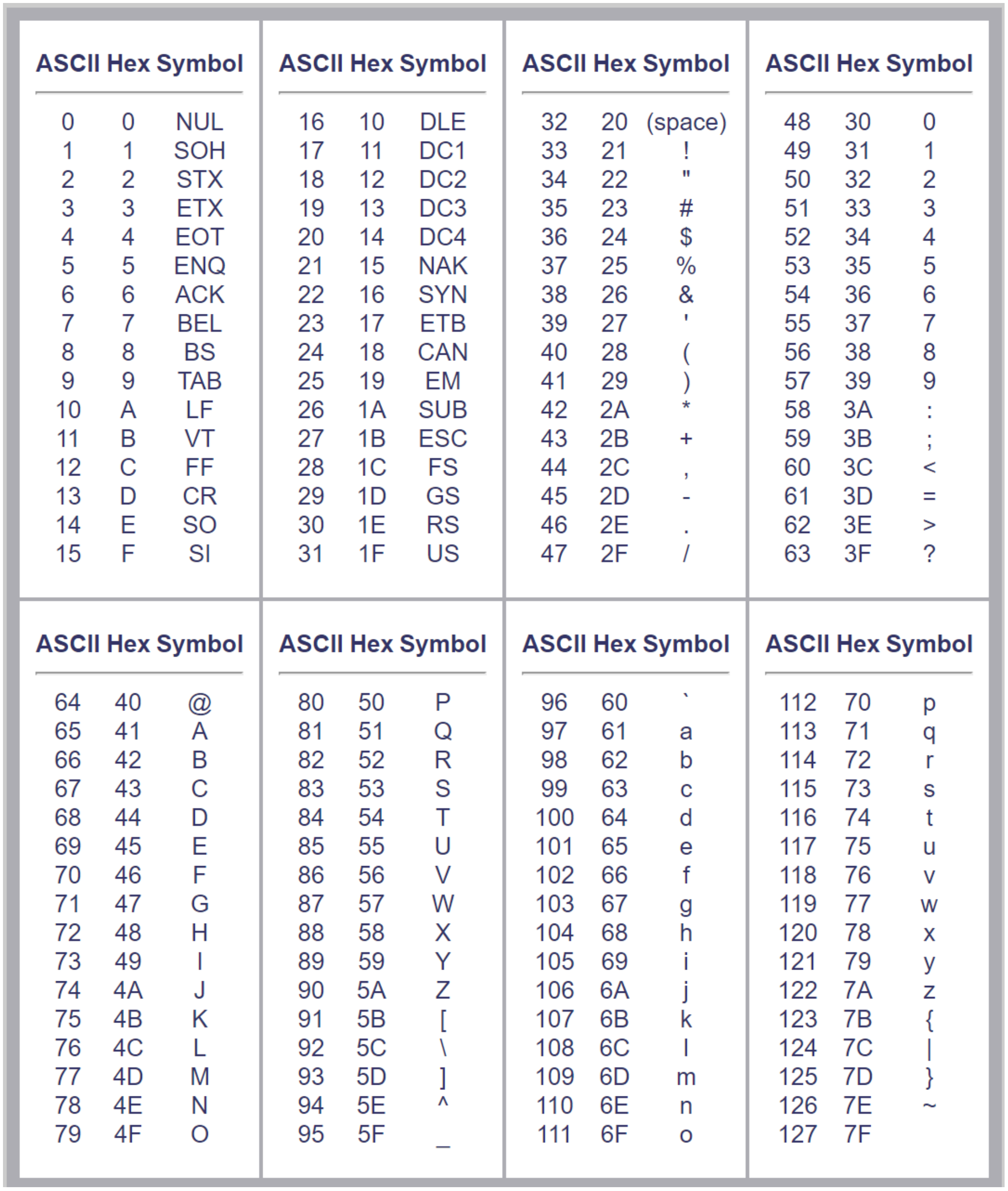
  Result: SCMA   249

4.1.7 ASCII

ASCII stands for **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange. Computers can only understand numbers, henceforth ASCII code is the numerical representation of a character such as 'a' or an action of some sort. ASCII encodes 128 specified characters into

7-bit integers. The ASCII character set contains uppercase and lowercase alphabets A-Za-z, numbers from 0-9, some special characters like {'{}[]@$!'}<>, and some non-printable control characters like newline (\n), backspace (\b), horizontal tab (\t), vertical tab (\v) etc. Table 4.2 is ASCII table.

Table 4.2 ASCII table



ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
0	0	NUL	16	10	DLE	32	20	(space)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	

In Python, we can use a function **ord** to convert any single text to ASCII code. Conversely, a function **chr** is used to convert ASCII code to a text.

Example 4.11

character

`ord('A')` Answer 65

`chr(125)` Answer '}'

`ord('EXAMPLE')` Answer ERROR (string)

4.2 Lists []

List, like an array in other language, is a collection which is ordered and changeable. It does not need to be homogeneous in terms of data type in a list. In other words, a single list can contain several data types like integers, strings, and objects.

4.2.1 Creating a list

To create a list in Python, we can either put the sequence inside square brackets [ ] separate each element by using comma ',' or use a **list** function, i.e., **list()**.

Example 4.12

	Code	Results
4.12.1	`x1 = [ ]`	[]
4.12.2	`x2 = list()`	[]
4.12.3	`x3 = [2,4,9]`	[2,4,9]
4.12.4	`x4 = [249]*10`	[249,249,249,249,249,249,249,249,249,249]
4.12.5	`x5 = list('SCMA 249')`	['S', 'C', 'M', 'A', ' ', '2', '4', '9']
4.12.6	`x6 = list(range(1,10,2))`	[1, 3, 5, 7, 9]
4.12.7	`x7 = ['SCMA', 249, [1,2019]]`	['SCMA', 249, [1, 2019]]
4.12.8	`x8 = [1,2] + [0,1]`	[1, 2, 0, 1]
4.12.9	`x9 = 2*([1,2] + 2*[0,1])`	[1, 2, 0, 1, 0, 1, 1, 2, 0, 1, 0, 1]

4.3.2 Useful functions and methods for list

Function/Method	Description
`len`	returns the number of items in the list
`sum`	returns the sum of all items in the list
`min`	returns the minimum of the items in the list
`max`	returns the maximum of the items in the list
`append(x)`	add **x** to the end of the list
`sorted(x)`	sorts the list
`count(x)`	returns the number of times **x** occurs in the list
`index(x)`	returns the location of the first occurrence of **x**
`remove(x)`	removes first occurrence of **x** from the list
`pop(ind)`	removes the item at index **ind** and returns its value
`insert(ind,x)`	inserts **x** at index **ind** of the list

Example 4.13

	Code	Result
4.15.1	`len(x9)`	12
4.15.2	`sum(x9)`	10
4.15.3	`min(x9)`	0
4.15.4	`max(x9)`	2

4.15.5	`x5.append('!')`
	`print(x5)`
		[‘S’, ‘C’, ‘M’, ‘A’, ‘ ‘, ‘2’, ‘4’, ‘9’, ‘!’]

4.15.6	`sorted(x9)`	[0, 0, 0, 0, 1, 1, 1, 1, 2, 2]
4.15.7	`x9.count(0)`	4
4.15.8	`x8.index(0)`	2

4.15.9	`c = x9[:]`
	`print(c)`
		[1, 2, 0, 1, 0, 1, 1, 2, 0, 1, 0, 1]

Example 4.14

Generate a list **L1** of 25 random numbers between 1 and 100.

```
from random import randint
L1 = []
for i in range(10):
    L1.append(randint(1,100))
print(L1)
```

Example 4.15

Here is a program to play a simple quiz game.

```
num_right = 0
# Question 1
print('What is the capital of Thailand?', end=' ')
guess = input()
if guess.lower()== 'bangkok':
      print('Correct!')
      num_right+=1
else:
      print('Wrong. The answer is Bangkok.')
print('You have', num_right, 'out of 1 right')

#Question 2
print('From which language is the word \'ketchup\' derived?',
end=' ')
guess = input()
if guess.lower()=='chinese':
      print('Correct!')
      num_right+=1
else:
      print('Wrong. The answer is Chinese.')
print('You have', num_right, 'out of 2 right,')
```

References

- Kittipon P, Kittipob P, Somchai P, Sukree S. Python 101. V1.0.2., Chulalongkorn University Printing House; 2018.
- Andrew J. Python: The Ultimate Beginners Guide!., CreateSpace Independent Publishing Platform; 2016.
- Brian H. A Practical Introduction to Python Programming., Crative Commons Attribution-Noncommercial-Share Alike 3.0; 2015.
- https://en.wikipedia.org/wiki/ASCII
- asciitable.com
- https://ascii.cl/