CS112 Introduction to Python Programming

Session 02: Strings

Shengwei Hou
Ph.D., Assistant Professor
Department of Ocean Science and Engineering

Fall 2022

SUSTech Southern University of Science and Technology

明德求是 日新自强
VIRTUE | TRUTH | ADVANCE

Contents

String basics

- String is a part of Python’s core data structures
- String is a list of keyboard characters as well as other characters not on your keyboard
- Strings are created by enclosing a sequence of characters within a pair of single or double quotes:

```
>>> a = "My dog's name is"
>>> a = 'My dog\'s name is'
>>> b = 'Bingo'
```

- Strings can be made up of numbers:

```
>>> d = "927"
```

String basics

- Strings can be concatenated using the “+” operator:

```
>>> c = a + " " + b
>>> c
"My dog's name is Bingo"
```

- Strings can also be repeated using the “*” operator:

```
>>> 'wow ' * 5
'wow wow wow wow wow '
```

- str() function:

```
>>> str(10)
'10'
>>> str(1.0)
'1.0'
```

String basics

- `in` and `not in`: check for the presence of a string in another string

```
>>> "apple" in "apple is a fruit"
True
>>> "orange" not in "apple is a fruit"
True
```

- Python compares strings based on the ASCII value of the characters using >, <, <=, >=, ==, !=:

```
>>> "january" == "jane"
False
>>> "january" != "jane"
True
>>> "january" < "jane"
False
```

```
>>> "january" > "jane"
True
>>> "january" <= "jane"
False
>>> "january" >= "jane"
True
```

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	*[NULL]*	32	20	*[SPACE]*	64	40	**@**	96	60	**`**
1	1	*[START OF HEADING]*	33	21	**!**	65	41	**A**	97	61	**a**
2	2	*[START OF TEXT]*	34	22	**"**	66	42	**B**	98	62	**b**
3	3	*[END OF TEXT]*	35	23	**#**	67	43	**C**	99	63	**c**
4	4	*[END OF TRANSMISSION]*	36	24	**$**	68	44	**D**	100	64	**d**
5	5	*[ENQUIRY]*	37	25	**%**	69	45	**E**	101	65	**e**
6	6	*[ACKNOWLEDGE]*	38	26	**&**	70	46	**F**	102	66	**f**
7	7	*[BELL]*	39	27	**'**	71	47	**G**	103	67	**g**
8	8	*[BACKSPACE]*	40	28	**(**	72	48	**H**	104	68	**h**
9	9	*[HORIZONTAL TAB]*	41	29	**)**	73	49	**I**	105	69	**i**
10	A	*[LINE FEED]*	42	2A	*****	74	4A	**J**	106	6A	**j**
11	B	*[VERTICAL TAB]*	43	2B	**+**	75	4B	**K**	107	6B	**k**
12	C	*[FORM FEED]*	44	2C	**,**	76	4C	**L**	108	6C	**l**
13	D	*[CARRIAGE RETURN]*	45	2D	**-**	77	4D	**M**	109	6D	**m**
14	E	*[SHIFT OUT]*	46	2E	**.**	78	4E	**N**	110	6E	**n**
15	F	*[SHIFT IN]*	47	2F	**/**	79	4F	**O**	111	6F	**o**
16	10	*[DATA LINK ESCAPE]*	48	30	**0**	80	50	**P**	112	70	**p**
17	11	*[DEVICE CONTROL 1]*	49	31	**1**	81	51	**Q**	113	71	**q**
18	12	*[DEVICE CONTROL 2]*	50	32	**2**	82	52	**R**	114	72	**r**
19	13	*[DEVICE CONTROL 3]*	51	33	**3**	83	53	**S**	115	73	**s**
20	14	*[DEVICE CONTROL 4]*	52	34	**4**	84	54	**T**	116	74	**t**
21	15	*[NEGATIVE ACKNOWLEDGE]*	53	35	**5**	85	55	**U**	117	75	**u**
22	16	*[SYNCHRONOUS IDLE]*	54	36	**6**	86	56	**V**	118	76	**v**
23	17	*[ENG OF TRANS. BLOCK]*	55	37	**7**	87	57	**W**	119	77	**w**
24	18	*[CANCEL]*	56	38	**8**	88	58	**X**	120	78	**x**
25	19	*[END OF MEDIUM]*	57	39	**9**	89	59	**Y**	121	79	**y**
26	1A	*[SUBSTITUTE]*	58	3A	**:**	90	5A	**Z**	122	7A	**z**
27	1B	*[ESCAPE]*	59	3B	**;**	91	5B	**[**	123	7B	**{**
28	1C	*[FILE SEPARATOR]*	60	3C	**<**	92	5C	\	124	7C	**|**
29	1D	*[GROUP SEPARATOR]*	61	3D	**=**	93	5D	**]**	125	7D	**}**
30	1E	*[RECORD SEPARATOR]*	62	3E	**>**	94	5E	**^**	126	7E	**~**
31	1F	*[UNIT SEPARATOR]*	63	3F	**?**	95	5F	**_**	127	7F	*[DEL]*

Strings are immutable

- As strings are immutable, it cannot be modified.
- The characters in a string cannot be changed once a string value is assigned to string variable. However, you can assign different string values to the same string variable.

```
>>> immutable = "dollar"
>>> immutable[0] = "c"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment

>>> immutable = "rollar"
>>> immutable
'rollar'
```

Built-in functions for strings

- Number of characters in a string:

```
>>> len("apple")
5
>>> len("dog's name")
10
```

- `min()` and `max()` return a character having the highest and lowest ASCII value:

```
>>> max("axel")
'x'
>>> min("brad")
'a'
```

String indexing

- Each of the string’s character corresponds to an index number starting from 0; square brackets are used to perform indexing in a string:

```
>>> phrase = "be yourself"
>>> len(phrase)
11
>>> phrase[0]
'b'
>>> phrase[10]
'f'
>>> phrase[11]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

```
>>> phrase[-1]
'f'
>>> phrase[-2]
'l'
```

Python’s index starts from zero

String slicing

- String slicing returns a sequence of characters beginning at start and extending up to but not including end; the index numbers are separated by a colon:

```
>>> phrase = "be yourself"
>>> phrase[0:4]
'be y'
>>> phrase[:4]
'be y'
>>> phrase[6:]
'rself'
>>> phrase[:]
'be yourself'
```

```
>>> phrase[4:4]
''
>>> phrase[4:20]
'ourself'
>>> phrase[-4:-1]
'sel'
>>> phrase[4:-1]
'oursel'
```

String slicing

- A third argument called **step** can be specified along with the **start** and **end** index numbers to specify steps in slicing; the default value of step is one:

```
>>> phrase = "be yourself"
>>> phrase[0:9:3]
'byr'
>>> phrase[0::3]
'byrl'
```

String slicing

- Strings can be joined with the `join()` function:

```
>>> date_of_birth = ["17", "09", "1950"]
>>> ":".join(date_of_birth)
'17:09:1950'

>>> social_app = ["instagram", "is", "a", "photo",
"sharing", "application"]
>>> " ".join(social_app)
'instagram is a photo sharing application'

>>> "123".join("amy")
'a123m123y'
```

String split

- Strings can be split with the split() function
- A given string is split into list of strings based on the specified separator; if the separator is not specified then whitespace is considered as the delimiter

```
>>> inventors = "edison, tesla, marconi, newton"
>>> ~~inventors.split(",")~~
['edison', ' tesla', ' marconi', ' newton']

>>> watches = "rolex hublot cartier omega"
>>> watches.split()
['rolex', 'hublot', 'cartier', 'omega']
```

String methods

```
>>> dir(str)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__',
'__doc__', '__eq__', '__format__', '__ge__', '__getattribute__',
'__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__',
'__init_subclass__', '__iter__', '__le__', '__len__', '__lt__',
'__mod__', '__mul__', '__ne__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__',
'__sizeof__', '__str__', '__subclasshook__', 'capitalize',
'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs',
'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha',
'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower',
'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join',
'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace',
'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip',
'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title',
'translate', 'upper', 'zfill']
```

String method examples

```
>>> "sailors".isalpha()
True

>>> "2018".isdigit()
True

>>> "Fact".islower()
False

>>> "TSAR BOMA".isupper()
True

>>> "galapagos".upper()
'GALAPAGOS'

>>> "TSAR BOMA".lower()
'tsar boma'
```

```
>>> "gAlPas".capitalize()
'Galpas'

>>> "cucumber".find("cu")
0

>>> "cucumber".find("um")
3

>>> "alpha".count("a")
2

>>> "Centennial Light".swapcase()
'cENTENNIAL lIGHT'
```

Format strings

Southern University of Science and Technology

- The `print` function prints everything as strings onto the console

```
>>> print("Hello World!!")
Hello World!!
```

- Use `str.format()` method if you need to insert the value of a variable, expression or an object into another string

```
>>> country = "China"
>>> print("I live in {0}".format(country))
I live in China
```

Index value starts from zero

```
>>> a = 10
>>> b = 20
>>> print("The values of a is {0} and b is {1}".format(a, b))
>>> print("The values of b is {1} and a is {0}".format(a, b))
```

Format strings

SUSTech Southern University of Science and Technology

- f-string format:

```
f"string_statements {variable_name [: {width}.{precision}]}"

>>> width = 10
>>> precision = 5
>>> value = 12.34567
>>> f'result: {value:{width}.{precision}}'
'result:      12.346'
>>> f'result: {value:{width}}'
'result:    12.34567'
>>> f'result: {value:.{precision}}'
'result: 12.346'
```

- Using *variable_name* along with either *width* or *precision* values should be separated by a colon
- Precision refers to the total number of digits that will be displayed in a number
- By default, strings are left-justified and numbers are right-justified

Escape sequences

- Escape sequences are a combination of a backslash (\) followed by either a letter or a combination of letters and digits

\\	Inserts a Backslash character in the string
\'	Inserts a Single Quote character in the string
\"	Inserts a Double Quote character in the string
\n	Inserts a New Line in the string
\t	Inserts a Tab in the string
\b	Inserts a Backspace in the string

```
>>> print("a"c")
  File "<stdin>", line 1
    print("a"c")
             ^
SyntaxError: invalid syntax

>>> print("a\"c")
a"c
```

\ Break a Line into Multiple lines while ensuring the continuation

```
>>> print("You can break \
... single line to \
... multiple lines")
You can break single line to multiple lines
```

Raw string

SUSTech Southern University of Science and Technology

- A raw string is created by prefixing the character `r` to the string
- A raw string ignores all types of formatting within a string including the escape characters

```
>>> print(r"he asked, \"What\'s the best way to code the
snake game?\" She answered, \"In *python* script\"")

he asked, \"What\'s the best way to code the snake
game?\" She answered, \"In *python* script\"
```

Southern University of Science and Technology

Exercises

Exercise 1: basics

```
>>> a = 'A'
>>> 2*a
```

(*a*) ‘AA’

(*b*) 2A

(*c*) A2

(*d*) None of the above

```
>>> a = 'A'
>>> b = 'B'
>>> a + b
```

(*a*) A + B

(*b*) AB

(*c*) BA

(*d*) None of the above

Exercise 2: indexing

Southern University of Science and Technology

```
>>> a = 'SUSTECH'
>>> a[1]
```

(a) S
(b) U
(c) C
(d) H

```
>>> a = 'SUSTECH'
>>> a[-1]
```

(a) S
(b) U
(c) C
(d) H

Exercise 3: slicing

```
>>> a = 'harsh'
>>> b = a[1: len(a)]
>>> b
```

(*a*) arsh

(*b*) hars

(*c*) harsh

(*d*) None of the above

```
>>> a = 'harsh'
>>> b = a[-3:len(a)]
>>> b
```

(*a*) rsh

(*b*) arsh

(*c*) harsh

(*d*) None of the above

Exercise 4: string methods

SUSTech Southern University of Science and Technology

1. Find out the output of the following code

```
>>> a = 'python programing
is fun'

>>> a.endswith('fun')

>>> a.endswith('boring')
```

Can you guess the function of `str.endswith()` ?

```
# Check the help page
>>> help(str.endswith)
```

2. Complete the following Python script

```
# Capitalize the first word
a = 'python programing is fun'
# Your code here
```

Expected output:

```
Python programing is fun
```

In Python, comments starts with a #, and will be ignored:

- Comments can be used to explain Python code.
- Comments can be used to make the code more readable.
- Comments can be used to prevent execution when testing code.

Exercise 5: string format

- Set two variables, first_name and last_name, then print your full name with str.format() and f-string methods.
- For example, if your first name is Lei, and your last name is Li, then you should print out

```
My name is Lei Li.
```

Exercise 6: escape sequence

- Write a string `harry_potter_quote` that print out the following text with escape sequences.

```
>>> print(harry_potter_quote)
"Ah, music. A 'magic' beyond all we do here!"
By Albus Dumbledore
```