07.2_Strings

March 1, 2023

1 Introduction to Python for Open Source Geocomputation

- Instructor: Dr. Wei Kang
- Class Location and Time: ENV 336, Mon & Wed 12:30 pm - 1:50 pm

Content:

- Strings:
  - slicing
  - for loops
  - comparison
  - methods

2 Standard Data Types in Python - strings

Category of Data type	Data type	Example
Numeric, scalar	Integer	1
	Floats	1.2
	Complex	1.5+0.5j
	Booleans	True
Container	strings	"Hello World"
	List	[1, "Hello World"]
	Tuple	(1, "Hello World")
	Set	{1, "Hello World"}
	Dictionary	{1: "Hello World", 2: 100}

3 Iterating over a string with `for` statements (for Loops) (traversal)

Traversal: start at the beginning, select each character in turn, do something to it, and continue until the end.

- `for` statments are used to iterate over sequences
- `for/range` statments are used to iterate over sequences using an index

[1]:
```
a = "python is fun!"
```

[2]:
```
"is" in a
```

[2]:
```
True
```

[3]:
```
for i in a:
    print(i)
```

```
p
y
t
h
o
n

i
s

f
u
n
!
```

[4]:
```
for i in range(len(a)):
    print(i, a[i])
```

```
0 p
1 y
2 t
3 h
4 o
5 n
6
7 i
8 s
9
10 f
11 u
```

```
12 n
13 !
```

3.1 Slicing strings

To access a continuous segment in a string

Structure of slicing a string: `string[start_index:end_index]` * string name * square brackets * `start`: the index to begin the slice * Colon `:` * `end`: the (non-inclusive) index to finish the slice

[5]:
```
my_string = "Hello World"
```

[6]:
```
my_string[0:5]
```

[6]:
```
'Hello'
```

Slice from the beginning of the string: `start` can be ignored

[7]:
```
my_string[:5]
```

[7]:
```
'Hello'
```

Slice all the way to the end of the string: `end` can be ignored

[8]:
```
my_string
```

[8]:
```
'Hello World'
```

[9]:
```
my_string[6:]
```

[9]:
```
'World'
```

[10]:
```
my_string[6:len(my_string)]
```

[10]:
```
'World'
```

[11]:
```
my_string[len(my_string)]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/var/folders/6m/8n2ktx1566j8yp0n_qx7x5bw0000gt/T/ipykernel_9874/1485035880.py i
  ↪<module>
----> 1 my_string[len(my_string)]

IndexError: string index out of range
```

[12]:
```
my_string
```

```
[12]: 'Hello World'
```

```
[13]: my_string[6:6]
```

```
[13]: ''
```

```
[14]: my_string[6:7]
```

```
[14]: 'W'
```

```
[15]: my_string[:]
```

```
[15]: 'Hello World'
```

```
[16]: my_string[1.0:3.0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/var/folders/6m/8n2ktx1566j8yp0n_qx7x5bw0000gt/T/ipykernel_9874/3599525107.py i
  ↪<module>
----> 1 my_string[1.0:3.0]

TypeError: slice indices must be integers or None or have an __index__ method
```

3.1.1 Group exercise:

Write python code to grab the "gin" slice from the string eng_string= 'engineer'

eng_string = 'engineer'

When you are done, raise your hand!

```
[18]: eng_string = 'engineer'
```

```
[19]: eng_string[2:5]
```

```
[19]: 'gin'
```

```
[20]: eng_string[-6:-3]
```

```
[20]: 'gin'
```

```
[21]: # Grab 'gin'slice
      eng_string[2:5]
```

```
[21]: 'gin'
```

3.1.2 Group Exercises

1. Define a string called `'banana'` and print out the first and last `'a'`.
2. Using the same string, grab the 2 possible slices that correspond to the word `'ana'` and print them out.

```
[22]: fruit_string = "banana"
```

```
[23]: fruit_string[-3:len(fruit_string)]
```

```
[23]: 'ana'
```

```
[24]: fruit_string[-3:]
```

```
[24]: 'ana'
```

```
[25]: fruit_string[1:4]
```

```
[25]: 'ana'
```

```
[26]: fruit_string[1:-2]
```

```
[26]: 'ana'
```

```
[27]: fruit_string[1]
```

```
[27]: 'a'
```

```
[28]: fruit_string[1:2]
```

```
[28]: 'a'
```

```
[29]: fruit_string[5]
```

```
[29]: 'a'
```

```
[30]: fruit_string[-1]
```

```
[30]: 'a'
```

```
[31]: fruit_string[-1:]
```

```
[31]: 'a'
```

3.1.3 `for` or `for/range` statement with string slice

Obtain each character of a substring

```
[32]: fruit_string = "banana is very sweet"
```

```
[33]: for i in range(6):
          print(fruit_string[i])
```

```
b
a
n
a
n
a
```

```
[34]: for i in fruit_string[:6]:
          print(i)
```

```
b
a
n
a
n
a
```

```
[35]: fruit_string[-5:]
```

```
[35]: 'sweet'
```

```
[36]: for i in fruit_string[-5:]:
          print(i)
```

```
s
w
e
e
t
```

```
[37]: fruit_string
```

```
[37]: 'banana is very sweet'
```

```
[38]: fruit_string[7:9]
```

```
[38]: 'is'
```

```
[39]: for i in fruit_string[7:9]:
          print(i)
```

```
i
s
```

```
[40]: ?range
```

```
[41]: for i in range(7,9):
          print(fruit_string[i])
```

```
i
s
```

3.1.4 *Translate that!*

What is indexing string in python? How do we implement it?

```
[42]: a = "python"
```

```
[43]: a[0]
```

```
[43]: 'p'
```

3.1.5 *Translate that!*

What is slicing string in python? How do we implement it?

```
[44]: a[1:4]
```

```
[44]: 'yth'
```

```
[45]: a[0:1]
```

```
[45]: 'p'
```

3.1.6 String comparison

Comparison operators `>`, `<`, `==`

```
[46]: "banana" == "banana"
```

```
[46]: True
```

```
[47]: "banana" > "pear"
```

```
[47]: False
```

```
[48]: "banana" > "Pear"
```

```
[48]: True
```

```
[49]: "banana" > "Pear".lower()
```

```
[49]: False
```

3.2 Built-in methods with strings

What is a method?

- functions associated with a particular data type or a class of objects (e.g., strings)
    - methods are essentially functions
- format: `mystring.method()`
- another way to call a method: the dot operator
    - the method comes after the dot
    - the name of the particular object it acts on comes first

```
[50]: AE_quote = "Everybody is a genius."
      AE_quote
```

```
[50]: 'Everybody is a genius.'
```

```
[51]: AE_quote.upper()
```

```
[51]: 'EVERYBODY IS A GENIUS.'
```

```
[52]: AE_quote
```

```
[52]: 'Everybody is a genius.'
```

```
[53]: AE_quote.lower()
```

```
[53]: 'everybody is a genius.'
```

```
[54]: AE_quote
```

```
[54]: 'Everybody is a genius.'
```

```
[55]: AE_quote.capitalize()
```

```
[55]: 'Everybody is a genius.'
```

```
[56]: a = AE_quote.lower()
      print(a)
      print(a.capitalize())
```

```
everybody is a genius.
Everybody is a genius.
```

```
[57]: AE_quote
```

```
[57]: 'Everybody is a genius.'
```

```
[58]: AE_quote.lower().capitalize()
```

```
[58]: 'Everybody is a genius.'
```

3.2.1 count() method

- gives the number of ocurrences of a substring in a range
- *Syntax:*

```
str.count(substring, start, end)
```

- start and end
  - integers that indicate the indices where to start and end the count
  - optional, if omitted, the whole string is inspected
  - end: non-inclusive

```
[59]: AE_quote
```

```
[59]: 'Everybody is a genius.'
```

```
[60]: AE_quote.count("e")
```

```
[60]: 2
```

```
[61]: AE_quote.count('e', 0, len(AE_quote))
```

```
[61]: 2
```

```
[62]: AE_quote
```

```
[62]: 'Everybody is a genius.'
```

```
[63]: AE_quote[10]
```

```
[63]: 'i'
```

```
[64]: AE_quote.count('e', 10, 20)
```

```
[64]: 1
```

```
[65]: AE_quote.count('e', 0, 11)
```

```
[65]: 1
```

```
[66]: AE_quote
```

```
[66]: 'Everybody is a genius.'
```

```
[67]: AE_quote.count('Everybody')
```

```
[67]: 1
```

```
[68]: AE_quote.count('EverybodyHello')
```

```
[68]: 0
```

3.2.2 find() method

- tells us if a string `'substr'` occurs in the string and return the index where the substring starts, otherwise it will return `-1`.
- *Syntax:*

`str.find(substring, start, end)`

- `start` and `end`
  - integers that indicate the indices where to start and end the count
  - optional, if omitted, the whole string is inspected

```
[69]: AE_quote
```

```
[69]: 'Everybody is a genius.'
```

```
[70]: AE_quote.find('Everybody')
```

```
[70]: 0
```

```
[71]: AE_quote[AE_quote.find('Everybody') : (AE_quote.
       ↪find('Everybody')+len("Everybody"))]
```

```
[71]: 'Everybody'
```

```
[72]: AE_quote.find('EverybodyHello')
```

```
[72]: -1
```

```
[73]: AE_quote
```

```
[73]: 'Everybody is a genius.'
```

```
[74]: AE_quote.find('genius')
```

```
[74]: 15
```

```
[75]: len('genius')
```

```
[75]: 6
```

```
[76]: AE_quote[15: 15 + len('genius')]
```

```
[76]: 'genius'
```

3.2.3 `index()` method

- tells us if a string `'substr'` occurs in the string and return the index where the substring starts, otherwise it will raise an error.
- *Syntax:*

`str.index(substring, start, end)`

- `start` and `end`
  - integers that indicate the indices where to start and end the count
  - optional, if omitted, the whole string is inspected

```
[77]: AE_quote.index('Everybody')
```

```
[77]: 0
```

```
[78]: AE_quote.index('genius')
```

```
[78]: 15
```

```
[79]: AE_quote.index('fish')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_9874/2761805799.py i␣
  ↪<module>
----> 1 AE_quote.index('fish')

ValueError: substring not found
```

3.2.4 `strip()` method

returns a copy of the string in which all characters given as argument are stripped from the beginning and the end of the string.

- Syntax:

`str.strip([chars])`

- Default argument is the space character (remove the white spaces)

```
[80]: ER_quote = "   Great minds discuss ideas; average minds discuss events; small␣
       ↪minds discuss people.   "
```

```
[81]: ER_quote
```

```
[81]: '   Great minds discuss ideas; average minds discuss events; small minds discuss
      people.   '
```

```
[82]: ER_quote.strip()
```

```
[82]: 'Great minds discuss ideas; average minds discuss events; small minds discuss
      people.'
```

```
[83]: ER_quote_new = ER_quote.strip()
      ER_quote_new
```

```
[83]: 'Great minds discuss ideas; average minds discuss events; small minds discuss
      people.'
```

```
[84]: ER_quote_new.strip('.')
```

```
[84]: 'Great minds discuss ideas; average minds discuss events; small minds discuss
      people'
```

```
[85]: ER_quote
```

```
[85]: '   Great minds discuss ideas; average minds discuss events; small minds discuss
      people.   '
```

```
[86]: ER_quote.strip('.')
```

```
[86]: '   Great minds discuss ideas; average minds discuss events; small minds discuss
      people.   '
```

4 Further readings

- "Built-in String Methods" in this tutorial.

5 Next Class

- additional string methods
- list

Readings:

- Chapter 10