08.1_Strings_Lists

March 7, 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:

- Additional Strings methods
- Lists

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}

2.0.1 Questions from our last class

- What is a method in python?

- What is a function in python?

2.1 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

2.1.1 Group Exercise

Write python code to get rid of the underscores `_` in the beginning of the sentence and the exclamation points `!` at the end of the sentence.

```
sentence = "___Great minds discuss ideas!!!"
```

When you are done, raise your hand!

```
[1]: sentence = "___Great minds discuss ideas!!!"
```

```
[2]: sentence.strip("_!")
```

```
[2]: 'Great minds discuss ideas'
```

```
[4]: sentence.strip("_").strip("!")
```

```
[4]: 'Great minds discuss ideas'
```

```
[5]: sentence
```

```
[5]: '___Great minds discuss ideas!!!'
```

The string methods are not **in-place** methods, which means the original string object/value is not changed. Instead, the methods return a value.

```
[6]: sentence.strip("_!")
```

```
[6]: 'Great minds discuss ideas'
```

```
[7]: sentence
```

```
[7]: '___Great minds discuss ideas!!!'
```

```
[8]: sentence_new = sentence.strip("_!")
     sentence_new
```

```
[8]: 'Great minds discuss ideas'
```

2.1.2 startswith() method

To find out if a string starts with a certain character(s).

- syntax:

```
str.startswith(substring)
```

- returned value: True or False

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

```
[10]: ER_quote
```

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

```
[11]: ER_quote.startswith('great')
```

```
[11]: False
```

```
[12]: ER_quote.startswith('Great')
```

```
[12]: False
```

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

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

```
[14]: ER_quote
```

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

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

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

```
[16]: ER_quote_new.startswith('great')
```

```
[16]: False
```

```
[17]: ER_quote_new.startswith('Great')
```

```
[17]: True
```

```
[18]: ER_quote
```

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

```
[19]: ER_quote_new
```

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

```
[20]: ER_quote_new.startswith('Great minds')
```

```
[20]: True
```

```
[21]: ER_quote_new.endswith('people.')
```

```
[21]: True
```

2.1.3 split() method

Returns a **list** of all the words in a string

- *Syntax:*

`str.split(separator, num)`

- separator: **a character** which splits our string
  - optional, default is None, meaning splitting according to any whitespace, and discard empty strings from the result.
- num: the number of splits
  - optional, default is unlimited

```
[22]: ER_quote
```

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

```
[24]: ER_quote.split(" ")
```

```
[24]: ['',
       '',
       '',
       'Great',
       'minds',
       'discuss',
       'ideas;',
       'average',
       'minds',
       'discuss',
       'events;',
       'small',
       'minds',
       'discuss',
       'people.',
       '',
       '',
       '']
```

```
[23]: ER_quote.split()
```

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

```
[25]: type(ER_quote.split())
```

```
[25]: list
```

```
[26]: ER_quote
```

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

```
[27]: ER_quote.split(";")
```

```
[27]: ['   Great minds discuss ideas',
 ' average minds discuss events',
 ' small minds discuss people.   ']
```

```
[28]: ER_quote.split("; ")
```

```
[28]: ['   Great minds discuss ideas',
 'average minds discuss events',
 'small minds discuss people.   ']
```

2.1.4 Group Exercise

Write python code to get each word in the sentence

`sentence = "__Great minds discuss ideas!!!"`

Hint: Use string method `split()` and `strip()`

When you are done, raise your hand!

```
[29]: sentence = "__Great minds discuss ideas!!!"
```

```
[31]: sentence.strip("_!")
```

```
[31]: 'Great minds discuss ideas'
```

```
[32]: sentence_new = sentence.strip("_!")
      sentence_new
```

```
[32]: 'Great minds discuss ideas'
```

```
[33]: sentence_new.split()
```

```
[33]: ['Great', 'minds', 'discuss', 'ideas']
```

```
[34]: sentence.strip("_!").split()
```

```
[34]: ['Great', 'minds', 'discuss', 'ideas']
```

2.1.5 Many more methods of strings

- Define a String variable `s = "python"`, use `.` and `Tab` to inspect all the methods of strings `s.[Tab]`
- Explore the functionality and syntax of a string method:
  - In a python interpreter (code cell):
    - `s.split?` (question mark after calling the method)
    - `help(s.split)` (use `help()` function)
  - google search **python strings split**
    - read documentation https://docs.python.org/3.3/library/stdtypes.html?highlight=split#str.split
    - read posts and examples from other pythoon users https://www.w3schools.com/python/ref_string_split.asp
- More on "Built-in String Methods"
  - tutorial
  - String methods on python documentation website

```
[35]: s = "python"
```

```
[36]: s.center?
```

```
[37]: s.endswith?
```

```
[38]: help(s.endswith)
```

```
Help on built-in function endswith:

endswith(…) method of builtins.str instance
    S.endswith(suffix[, start[, end]]) -> bool

    Return True if S ends with the specified suffix, False otherwise.
    With optional start, test S beginning at that position.
    With optional end, stop comparing S at that position.
    suffix can also be a tuple of strings to try.
```

```
[ ]: s.split?
```

```
[ ]: help(s.split)
```

3 Standard Data Types in Python - Lists

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}

```
[39]: sentence.split()
```

```
[39]: ['__Great', 'minds', 'discuss', 'ideas!!!']
```

3.1 What is a list in python?

- syntax:

`[value1, value2, value3]`

- A list is a ordered sequence of values
- The value can be any type
- The values in a list are called elements or sometimes items
- A list is mutable

- One of the most useful built-in types

3.2 Creating a list

- from other functions, e.g., `str.split()`
- assignment statment with `string_name = [value1, value2, value3]`
- `list` function

```
[40]: list_a = [1, "happy", 1+9j, 2.3, True]
      list_a
```

```
[40]: [1, 'happy', (1+9j), 2.3, True]
```

```
[41]: type(list_a)
```

```
[41]: list
```

```
[43]: empty_list = []
```

```
[44]: type(empty_list)
```

```
[44]: list
```

Empty list

```
[45]: a = []
```

```
[46]: a
```

```
[46]: []
```

```
[47]: a[0]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[47], line 1
----> 1 a[0]

IndexError: list index out of range
```

```
[48]: list("python")
```

```
[48]: ['p', 'y', 't', 'h', 'o', 'n']
```

```
[49]: range(2)
```

```
[49]: range(0, 2)
```

```
[50]: list(range(2))
```

```
[50]: [0, 1]
```

3.3 Indexing a list

similar to indexing string: index starts from 0!

```
[51]: list_a = [1, "happy", 1+9j, 2.3, True]
```

```
[52]: list_a[0]
```

```
[52]: 1
```

```
[53]: list_a[1]
```

```
[53]: 'happy'
```

```
[54]: list_a[-1]
```

```
[54]: True
```

3.4 Slicing a list

Lists can be sliced in a similar fashion to what we saw for strings

```
[55]: list_a
```

```
[55]: [1, 'happy', (1+9j), 2.3, True]
```

```
[56]: list_a[1:]
```

```
[56]: ['happy', (1+9j), 2.3, True]
```

```
[57]: list_a[1:-1]
```

```
[57]: ['happy', (1+9j), 2.3]
```

3.4.1 built-in functions on numerical lists

```
[1]: list_int = [3,2,4]
     list_int
```

```
[1]: [3, 2, 4]
```

```
[2]: len(list_int)
```

[2]: 3

How to calculate the sum of all the numbers in the list?

[3]:
```python
list_int[0] + list_int[1] + list_int[2]
```

[3]: 9

[4]:
```python
list_int
```

[4]: [3, 2, 4]

[5]:
```python
s = "python"
for i in s:
    print(i)
```

```
p
y
t
h
o
n
```

[6]:
```python
for i in list_int:
    print(i)
```

```
3
2
4
```

3.4.2 Group Exploration Exercise

Write python code to calculate the average value of all the numbers in a list of numbers?

```python
list_int = [3,2,4]
```

When you are done, raise your hand!

[7]:
```python
list_int = [3,2,4]
```

[8]:
```python
sum_list = 0
for i in list_int:
    sum_list = sum_list + i
    print(i, sum_list)
sum_list
```

```
3 3
2 5
4 9
```

```
[8]: 9
```

```
[9]: sum(list_int)
```

```
[9]: 9
```

```
[10]: len(list_int)
```

```
[10]: 3
```

The average value of all the numbers in a list of numbers?

- find the total value
- find the number of numbers

```
[11]: sum(list_int)/len(list_int)
```

```
[11]: 3.0
```

```
[12]: list_int
```

```
[12]: [3, 2, 4]
```

```
[13]: max(list_int)
```

```
[13]: 4
```

```
[14]: min(list_int)
```

```
[14]: 2
```

4 Next Class

- lists
- tuples

Readings:

- Chapter 12

```
[ ]:
```