05.2_Strings_Lists

September 22, 2023

1 Introduction to Python for Open Source Geocomputation

- Instructor: Dr. Wei Kang

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()`
- 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'
```

[3]:
```
sentence.strip("_").strip("!")
```

[3]:
```
'Great minds discuss ideas'
```

[4]:
```
sentence
```

[4]:
```
'___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.

[5]:
```
sentence.strip("_!")
```

[5]:
```
'Great minds discuss ideas'
```

[6]:
```
sentence
```

[6]:
```
'___Great minds discuss ideas!!!'
```

[7]:
```
sentence_new = sentence.strip("_!")
sentence_new
```

[7]:
```
'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`

[8]:
```
ER_quote = "   Great minds discuss ideas; average minds discuss events; small␣
↪minds discuss people.   "
```

[9]:
```
ER_quote
```

[9]:
```
'   Great minds discuss ideas; average minds discuss events; small minds discuss
people.   '
```

[10]:
```
ER_quote.startswith('great')
```

[10]:
```
False
```

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

[11]:
```
False
```

[12]:
```
ER_quote.strip()
```

[12]:
```
'Great minds discuss ideas; average minds discuss events; small minds discuss
people.'
```

[13]:
```
ER_quote
```

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

[14]:
```
ER_quote_new = ER_quote.strip()
ER_quote_new
```

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

[15]:
```
ER_quote_new.startswith('great')
```

[15]:
```
False
```

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

[16]:
```
True
```

[17]:
```
ER_quote
```

[17]:
```
'   Great minds discuss ideas; average minds discuss events; small minds discuss
people.   '
```

[18]:
```
ER_quote_new
```

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

[19]:
```
ER_quote_new.startswith('Great minds')
```

[19]:
```
True
```

[20]:
```
ER_quote_new.endswith('people.')
```

[20]:
```
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

[21]:
```
ER_quote
```

[21]:
```
'   Great minds discuss ideas; average minds discuss events; small minds discuss
people.   '
```

[22]:
```
ER_quote.split()
```

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

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

[23]:
```
list
```

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

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

[25]:
```
ER_quote
```

[25]:
```
'   Great minds discuss ideas; average minds discuss events; small minds discuss
people.   '
```

[26]:
```
ER_quote.split(";")
```

[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.   ']
```

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!

[28]:
```
sentence = "__Great minds discuss ideas!!!"
```

[29]:
```
sentence.strip("_!")
```

[29]:
```
'Great minds discuss ideas'
```

[30]:
```
sentence.strip("_!").split()
```

[30]:
```
['Great', 'minds', 'discuss', 'ideas']
```

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

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

[32]:
```
sentence_stripped.split(" ")
```

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

[33]:
```
sentence.strip("_!")
```

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

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

[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"
```

[ ]:
```
s.
```

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

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

[38]:
```
"ABC".center(10)
```

[38]:
```
'   ABC    '
```

[39]:
```
s.endswith?
```

[40]:
```
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.
```

[41]:
```
s.split?
```

[42]:
```
help(s.split)
```

```
Help on built-in function split:

split(sep=None, maxsplit=-1) method of builtins.str instance
    Return a list of the substrings in the string, using sep as the separator
string.

      sep
        The separator used to split the string.

        When set to None (the default value), will split on any whitespace
        character (including \\n \\r \\t \\f and spaces) and will discard
        empty strings from the result.
      maxsplit
        Maximum number of splits (starting from the left).
        -1 (the default value) means no limit.

    Note, str.split() is mainly useful for data that has been intentionally
    delimited.  With natural text that includes punctuation, consider using
    the regular expression module.
```

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}

[43]:
```
sentence
```

[43]:
```
'__Great minds discuss ideas!!!'
```

[44]:
```
sentence.split()
```

[44]:
```
['__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

[45]:
```
list_a = [1, "happy", 1+9j, 2.3, True]
list_a
```

[45]:
```
[1, 'happy', (1+9j), 2.3, True]
```

[46]:
```
type(list_a)
```

[46]:
```
list
```

[47]:
```
empty_list = []
```

[48]:
```
type(empty_list)
```

[48]:
```
list
```

Empty list

[49]:
```
empty_list[0]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[49], line 1
----> 1 empty_list[0]

IndexError: list index out of range
```

[50]:
```
list("python")
```

[50]:
```
['p', 'y', 't', 'h', 'o', 'n']
```

[51]:
```
range(2)
```

[51]:
```
range(0, 2)
```

[52]:
```
list(range(2))
```

[52]:
```
[0, 1]
```

3.3 Indexing a list

similar to indexing string: index starts from 0!

[53]:
```
list_a = [1, "happy", 1+9j, 2.3, True]
```

[54]:
```
list_a[0]
```

[54]:
```
1
```

[55]:
```
list_a[1]
```

[55]:
```
'happy'
```

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

[56]:
```
True
```

3.4 Slicing a list

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

Difference:

- slicing a list will result in a list
- indexing a list will result in potentially other data types (depend on the data type of the item)

[57]:
```
list_a
```

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

[58]:
```
list_a[1:]
```

[58]:
```
['happy', (1+9j), 2.3, True]
```

[59]:
```
list_a[1:-1]
```

[59]:
```
['happy', (1+9j), 2.3]
```

3.4.1 built-in functions on numerical lists

[60]:
```
list_int = [3,2,4]
list_int
```

[60]:
```
[3, 2, 4]
```

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

[61]:
```
list_int[0] + list_int[1] + list_int[2]
```

[61]:
```
9
```

[62]:
```
s = "python"
for i in s:
    print(i)
```

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

[63]:
```
list_int
```

[63]:
```
[3, 2, 4]
```

[64]:
```
for i in list_int:
    print(i)
```

```
3
2
4
```

[65]:
```
sum_list = 0
for i in list_int:
    sum_list = sum_list + i
print(sum_list)
```

```
9
```

3.4.2 Group Exercise

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

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

When you are done, raise your hand!

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

- find the total value
- find the length of numbers (the list)

[66]:
```
sum_list = 0
for i in list_int:
    sum_list = sum_list + i
    print(i, sum_list)
```

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

[67]:
```
sum_list / len(list_int)
```

[67]:
```
3.0
```

[68]:
```
sum(list_int)
```

[68]:
```
9
```

[69]:
```
sum(list_int)/len(list_int)
```

[69]:
```
3.0
```

[70]:
```
list_int
```

[70]:
```
[3, 2, 4]
```

Functions `max()` and `min`

[71]:
```
max(list_int)
```

[71]: 4

[72]:
```
min(list_int)
```

[72]: 2

[73]:
```
list_heter = ["2",1]
```

[74]:
```
max(list_heter)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[74], line 1
----> 1 max(list_heter)

TypeError: '>' not supported between instances of 'int' and 'str'
```

4 Next Class

- lists

Readings:

- Chapter 12