Python List Length and Element Checks and Defaults

Fan Wang

2020-10-24

Contents

1 List

Go to the **RMD**, **PDF**, or **HTML** version of this file. Go back to fan's Python Code Examples Repository (bookdown site).

```
import numpy as np
```

1.1 Convert a List to a String List

Given a list of string and numeric values, convert to a list of string values. The MAP function is like apply in R.

- How to concatenate items in a list to a single string?

```
ls_spec_key = ['ng_s_d', 'esti_test_11_simu', 2, 3]
ls_st_spec_key = list(map(str, ls_spec_key))
print(ls_st_spec_key)
```

```
## ['ng_s_d', 'esti_test_11_simu', '2', '3']
```

Additionally, append some common element to each element of the string using MAP.

```
ls_spec_key = ['ng_s_d', 'esti_test_11_simu', 2, 3]
ls_st_spec_key = list(map(lambda x: 'add++' + str(x), ls_spec_key))
print(ls_st_spec_key)
```

```
## ['add++ng_s_d', 'add++esti_test_11_simu', 'add++2', 'add++3']
```

Equivalently, via list comprehension

```
ls_spec_key = ['ng_s_d', 'esti_test_11_simu', 2, 3]
ls_st_spec_key = ['list_comprehension' + str(spec_key) for spec_key in ls_spec_key]
print(ls_st_spec_key)
```

```
## ['list_comprehensionng_s_d', 'list_comprehensionesti_test_11_simu', 'list_comprehension2', 'list_com
```

1.2 Concatenate a List to a String with Separator

Given a list of strings and numeric data types, concatenate list to a string with some separator. Also in reverse, generate a list by breaking a string joined by some separator.

```
ls_spec_key = ['ng_s_d', 'esti_test_11_simu', 2, 3]
st_separator = '='
st_spec_key = st_separator.join(list(map(lambda x : str(x), ls_spec_key)))
print(st_spec_key)
```

```
## ng_s_d=esti_test_11_simu=2=3
```

Now break string apart:

```
st_spec_key = '='.join(list(map(lambda x : '$' + str(x) + '$', ['ng_s_d', 'esti_test_11_simu', 2, 3])))
print(st_spec_key.split('='))
```

```
## ['$ng_s_d$', '$esti_test_11_simu$', '$2$', '$3$']
```

1.3 Add Nth Element to List when Nth Element Does not Exist

There is a list with 2 elements, check if the list has 3 elements, if not, add another element.

```
ls_string_A = ['c', '20180918']
ls_string_B = ['c', '20180918', ['esti_param.alpha_k']]

for ls_string in [ls_string_A, ls_string_B]:
  if len(ls_string) == 2:
    ls_string.insert(2, None)

  print(ls_string)
```

```
## ['c', '20180918', None]
## ['c', '20180918', ['esti_param.alpha_k']]
```

1.4 Check If List Has N Elements of None for Some Elements

In the example below, for A, B and C, do something, for D and E do something else.

```
ls_string_A = ['c', '20180918']
ls_string_B = ['c', '20180918', None]
ls_string_C = ['c', '20180918', None, None]
ls_string_D = ['c', '20180918', ['esti_param.alpha_k'], None]
ls_string_E = ['c', '20180918', ['esti_param.alpha_k'], 5]

for ls_string in [ls_string_A, ls_string_B, ls_string_C, ls_string_D, ls_string_E]:
  if len(ls_string) >= 3 and ls_string[2] is not None:
    print(ls_string)
  else:
    print(ls_string[0:2])
```

```
## ['c', '20180918']
## ['c', '20180918']
## ['c', '20180918']
## ['c', '20180918', ['esti_param.alpha_k'], None]
## ['c', '20180918', ['esti_param.alpha_k'], 5]
```

1.5 Add a Default Value to Nth Element of List

There is a string list with potential potentially three elements. But sometimes the input only has two elements. Provide default third element value if third element is NONE or if the string list only has two elements.

```
ls_string_A = ['c', '20180918']
ls_string_B = ['c', '20180918', None]
ls_string_C = ['c', '20180918', ['esti_param.alpha_k']]

for ls_string in [ls_string_A, ls_string_B, ls_string_C]:

  if len(ls_string) <= 2:
    # Deals with situation A
    ls_string.append(['esti_param.alpha_k'])
  elif ls_string[2] is None:
    # Deals with situation B
    ls_string[2] = ['esti_param.alpha_k']
  else:
    # Situation C
    pass

  print(ls_string)
```

```
## ['c', '20180918', ['esti_param.alpha_k']]
## ['c', '20180918', ['esti_param.alpha_k']]
## ['c', '20180918', ['esti_param.alpha_k']]
```

Now do the same thing for a numeric list:

```
ls_string_A = [11, 22]
ls_string_B = [11, 22, None]
ls_string_C = [11, 22, 33]

for ls_string in [ls_string_A, ls_string_B, ls_string_C]:

  if len(ls_string) <= 2:
    # Deals with situation A
    ls_string.append(33)
  elif ls_string[2] is None:
    # Deals with situation B
    ls_string[2] = 33
  else:
    # Situation C
    pass

  print(ls_string)
```

```
## [11, 22, 33]
## [11, 22, 33]
## [11, 22, 33]
```