Expand and pass list, tuple, dict to function arguments in Python

Posted: 2019-06-24 / Tags: Python, List, Dictionary

In Python, you can expand list, tuple, and dictionarie (`dict`), and pass each element to function arguments.

Add `*` to a list or tuple and `**` to a dictionary when calling a function, then elements are passed to arguments.Note the number of asterisks `*`.

Here, the following contents will be described.

- Expand list and tuple with `*`
  - With default arguments
  - With variable-length arguments
- Expand the dictionary (`dict`) with `**`
  - With default arguments
  - With variable-length arguments

See the following posts for basic usage of Python functions, default arguments, and variable-length arguments with `*` and `**` when defining functions.

- **Related:** Define and call functions in Python (def, return)
- **Related:** Default arguments in Python
- **Related:** Variable-length arguments (*args,* *kwargs) in Python

```
func(*t)
# one
# two
# three

func(*('one', 'two', 'three'))
# one
# two
# three
```

source: argument_expand_list_tuple.py

The following description is given in the case of a list, but the same applies to tuples.

If the number of elements does not match the number of arguments, `TypeError` will occur.

```
# func(*['one', 'two'])
# TypeError: func() missing 1 required positional argument: 'arg3'

# func(*['one', 'two', 'three', 'four'])
# TypeError: func() takes 3 positional arguments but 4 were given
```

source: argument_expand_list_tuple.py

With default arguments

If the function has default arguments, the default arguments will be used if the number of elements is insufficient. If there are many elements, `TypeError` will occur.

```
def func_default(arg1=1, arg2=2, arg3=3):
    print(arg1)
    print(arg2)
    print(arg3)

func_default(*['one', 'two'])
# one
# two
```

Expand list and tuple with *

When specifying a list or tuple with `*` as an argument, it is expanded and each element is passed to each argument.

```
def func(arg1, arg2, arg3):
    print(arg1)
    print(arg2)
    print(arg3)

l = ['one', 'two', 'three']

func(*l)
# one
# two
# three

func(*['one', 'two', 'three'])
# one
# two
# three

t = ('one', 'two', 'three')
```

```
# 3

func_default(*['one'])
# one
# 2
# 3

# func_default(*['one', 'two', 'three', 'four'])
# TypeError: func_default() takes from 0 to 3 positional arguments but 4 were gi
```

source: argument_expand_list_tuple.py

With variable-length arguments

If the function has a variable-length argument (`*args`), all elements after the positional argument are passed to the variable-length argument.

```
def func_args(arg1, *args):
    print(arg1)
    for arg in args:
        print(arg)

func_args(*['one', 'two'])
# one
# two

func_args(*['one', 'two', 'three'])
# one
# two
# three

func_args(*['one', 'two', 'three', 'four'])
# one
# two
# three
# four
```

source: argument_expand_list_tuple.py

Expand the dictionary (dict) with **

When specifying a dictionary (`dict`) with `**` as an argument, `key` will be expanded as an argument name and `value` as the value of the argument. Each element will be passed as keyword arguments.

```
def func(arg1, arg2, arg3):
    print(arg1)
    print(arg2)
    print(arg3)

d = {'arg1': 'one', 'arg2': 'two', 'arg3': 'three'}

func(**d)
# one
# two
# three

func(**{'arg1': 'one', 'arg2': 'two', 'arg3': 'three'})
# one
# two
# three
```

source: argument_expand_dict.py

If there is no key that matches the argument name, or if there is a key that does not match the argument name, `TypeError` will occur.

```
# func(**{'arg1': 'one', 'arg2': 'two'})
# TypeError: func() missing 1 required positional argument: 'arg3'

# func(**{'arg1': 'one', 'arg2': 'two', 'arg3': 'three', 'arg4': 'four'})
# TypeError: func() got an unexpected keyword argument 'arg4'
```

source: argument_expand_dict.py

With default arguments

If the function has default arguments, only the value of the argument name matching the dictionary key is updated.

If there is a key that does not match the argument name, `TypeError` will occur.

```
def func_default(arg1=1, arg2=2, arg3=3):
    print(arg1)
    print(arg2)
    print(arg3)

func_default(**{'arg1': 'one'})
# one
# 2
# 3

func_default(**{'arg2': 'two', 'arg3': 'three'})
# 1
# two
# three

# func_default(**{'arg1': 'one', 'arg4': 'four'})
# TypeError: func_default() got an unexpected keyword argument 'arg4'
```

source: argument_expand_dict.py

With variable-length arguments

If the function has a variable-length argument (`**kwargs`), all elements with keys that do not match the argument name are passed to the variable-length argument.

```
def func_kwargs(arg1, **kwargs):
    print('arg1', arg1)
    for k, v in kwargs.items():
        print(k, v)

func_kwargs(**{'arg1': 'one', 'arg2': 'two', 'arg3': 'three'})
# arg1 one
```

Related Posts

- Convert lists and tuples to each other in Python
- Shuffle a list, string, tuple in Python (random.shuffle, sample)
- Sort a list, string, tuple in Python (sort, sorted)
- Add an item to a list in Python (append, extend, insert)
- How to flatten a list of lists in Python
- Remove an item from a list in Python (clear, pop, remove, del)
- How to slice a list, string, tuple in Python
- Reverse a list, string, tuple in Python (reverse, reversed)
- Transpose 2D list in Python (swap rows and columns)
- Swap values in a list or values of variables in Python
- Convert numpy.ndarray and list to each other
- Convert pandas.DataFrame, Series and list to each other
- Unpack a tuple / list in Python
- Random sampling from a list in Python (random.choice, sample, choices)
- Alpha blending and masking of images with Python, OpenCV, NumPy

---


```
# arg2 two
# arg3 three

func_kwargs(**{'arg1': 'one', 'arg2': 'two', 'arg3': 'three', 'arg4': 'four'})
# arg1 one
# arg2 two
# arg3 three
# arg4 four

func_kwargs(**{'arg1': 'one', 'arg3': 'three'})
# arg1 one
# arg3 three
```

source: argument_expand_dict.py

Related Categories

- Python
- List
- Dictionary