Python Function Tuple and Dictionary as Arguments with *args and **kwargs

Fan Wang

2020-10-21

Contents

1 Function Arguments

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

```
import pprint
```

1.1 Python Dictionary As Argument via kwargs

There is a python function that outputs a dictionary with key and value pairs that specify key aspects of how a model should be solved. For example, one of the parameters could specify the *vcpu* requirement. This *vcpu* requirement might change, and so it should be easy to update this key with alternative values.

These are accomplished in the following manner. Define the full key-value pair list, with default values for several dictionaries, with model simulation, support, and compute parameters for example. These lists could be updated with some default alternative combinations, or alternatively, it could be updated with externally provided dictionary with both updated values for existing keys, or even additional key value pairs.

First, we create a function that processes and outputs default parameters, it has two inputs, *it_default_group* to specify pre-fixed adjustments from defaults, and *kwargs* that allows for arbitrarily modifications and additions to parameter dictionary.

```
def gen_compesti_spec(it_default_group=None, **kwargs):
    # A. Define the default parameter keys and values
    esti_specs = {'esti_method': 'MomentsSimuStates',
                  'momsets_type': ['a', '20180805a'],
                  'esti_param_vec_count': 1,
                  'esti_max_func_eval': 10,
                  'graph_frequncy': 20}
    compute_specs = {'cpu': str(1024 * 1),
                     'memory': str(517),  # only need about 160 mb in reality
                     'workers': 1,
                     'aws_fargate': False}

    # B. For different
    compesti_specs = {**compute_specs, **esti_specs}
```

```
    # C. Update dictionaries with parameter group values
    if it_default_group == 1:
        compesti_specs_updates = {'memory': str(1024 * 55),
                                  'compute_param_vec_count': 6,
                                  'esti_param_vec_count': 640}
        compesti_specs.update(compesti_specs_updates)

    # D. Update with kward, could append new
    compesti_specs.update(kwargs)

    return compesti_specs
```

Second, we test the defaults:

```
compesti_specs = gen_compesti_spec()
pprint.pprint(compesti_specs, width=1)
```

```
## {'aws_fargate': False,
##  'cpu': '1024',
##  'esti_max_func_eval': 10,
##  'esti_method': 'MomentsSimuStates',
##  'esti_param_vec_count': 1,
##  'graph_frequncy': 20,
##  'memory': '517',
##  'momsets_type': ['a',
##                   '20180805a'],
##  'workers': 1}
```

Third, we test using default group 1, pre-fixed changes to defaults:

```
compesti_specs = gen_compesti_spec(it_default_group=1)
pprint.pprint(compesti_specs, width=1)
```

```
## {'aws_fargate': False,
##  'compute_param_vec_count': 6,
##  'cpu': '1024',
##  'esti_max_func_eval': 10,
##  'esti_method': 'MomentsSimuStates',
##  'esti_param_vec_count': 640,
##  'graph_frequncy': 20,
##  'memory': '56320',
##  'momsets_type': ['a',
##                   '20180805a'],
##  'workers': 1}
```

Fourth, we use kwargs to feed in arbitrary dictionary to update and append to existing parameter dictionary:

```
compesti_specs_updates = {'esti_method': 'MomentsSimuStateszzz',
                          'moments_type': ['a', '20180805azzz'],
                          'momsets_type': ['a', '20180805azzz'],
                          'momsets_type_uuu': ['a', '20180805azzz']}
compesti_specs = gen_compesti_spec(it_default_group=None, **compesti_specs_updates)
pprint.pprint(compesti_specs, width=1)
```

```
## {'aws_fargate': False,
##  'cpu': '1024',
```

```
##  'esti_max_func_eval': 10,
##  'esti_method': 'MomentsSimuStateszzz',
##  'esti_param_vec_count': 1,
##  'graph_frequncy': 20,
##  'memory': '517',
##  'moments_type': ['a',
##                   '20180805azzz'],
##  'momsets_type': ['a',
##                   '20180805azzz'],
##  'momsets_type_uuu': ['a',
##                       '20180805azzz'],
##  'workers': 1}
```

1.2 Named Argument List and Dictionary

Define a function with named and unnamed arguments:

```
def gen_compesti_spec_named(it_default_group, esti_method, memory=123, graph_frequncy=10):
    # A. Define the default parameter keys and values
    esti_specs = {'esti_method': 'MomentsSimuStates',
                  'momsets_type': ['a', '20180805a'],
                  'it_default_group': it_default_group,
                  'esti_param_vec_count': 1,
                  'esti_max_func_eval': 10,
                  'graph_frequncy': graph_frequncy}
    compute_specs = {'cpu': str(1024 * 1),
                     'memory': str(memory),  # only need about 160 mb in reality
                     'workers': 1,
                     'aws_fargate': False}

    # B. For different
    compesti_specs = {**compute_specs, **esti_specs}

    return compesti_specs
```

Provide inputs for the first two unnamed parameters explicitly. Then provided the two named parameters via a dictionary:

```
dc_inputs = {'memory':12345, 'graph_frequncy':2}
compesti_specs = gen_compesti_spec_named(None, 'MomentsSimuStates', **dc_inputs)
pprint.pprint(compesti_specs, width=1)
```

```
## {'aws_fargate': False,
##  'cpu': '1024',
##  'esti_max_func_eval': 10,
##  'esti_method': 'MomentsSimuStates',
##  'esti_param_vec_count': 1,
##  'graph_frequncy': 2,
##  'it_default_group': None,
##  'memory': '12345',
##  'momsets_type': ['a',
##                   '20180805a'],
##  'workers': 1}
```