3. Argument

Fig. 3.5 Photo by Alejandro Piñero Amerio on Unsplash

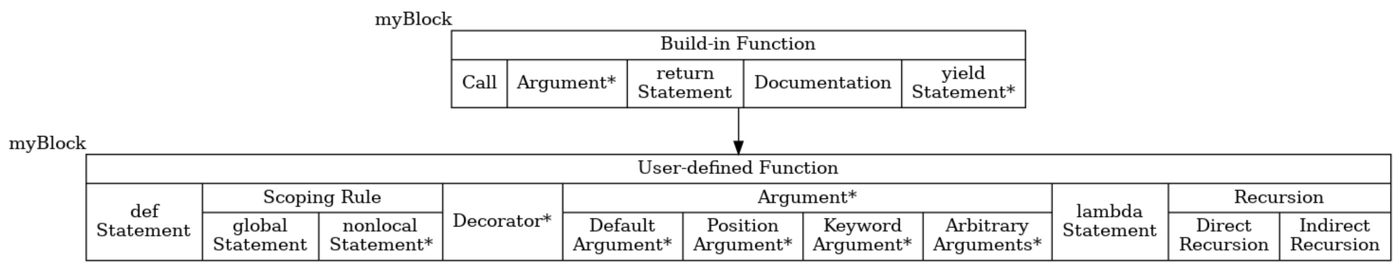
Note

Outline (v20220501)

Note

Roadmap

1. This topic: Function



2. Course: Python 1
3. Subject: Programming
4. Field
    a. Software Engineering (SE)
    b. Computer Science and Information Engineering (CSIE)
    c. Electrical/Electronics Engineering (EE)

3.1. Function Argument

1. Arguments are specified after the function name, inside the parentheses.
2. We can add as many arguments as we want, just separate them with a comma.
3. Syntax

```python
def FunctionName(arg1, arg2, ...):
    # arg are separated by comma
    # something to do...
```

3.1.1. Ex1: Full Name

1. Code

Listing 3.1.1.10 /src/Function/ArgumentFunction.py

```python
'''
@since: 20150524
@author: cph
'''
def FullName(sLastName, sFirstName):
    print('My full name is', sFirstName, sLastName + '.')

FullName('Cheng', 'Po-Hsun')
```

2. Output

```
My full name is Po-Hsun Cheng.
```

3.2. Keyword Argument

1. We can also send arguments with the key = value syntax.
2. This method the order of the arguments does not matter.
3. Usage
    a. We can often leave out arguments that have default values.
    b. We can rearrange arguments in a way that makes them most readable.
    c. We call arguments by their names to make it more clear what they represent.
4. Syntax

```
def FunctionName(key1=value1, key2=value2, ...):
    # arguments are shown key-value pairs
    # arguments are separated by comma
    # something to do...
```

3.2.1. Ex1: Full Name Again

1. Code

Listing 3.2.1.2 /src/Function/ArgumentKeyword.py

```
'''
@since: 20150524
@author: cph
'''
def FullName(sLastName, sFirstName):
    print('My full name is', sFirstName, sLastName + '.')

FullName(sLastName = 'Cheng',
         sFirstName = 'Po-Hsun')

FullName(sFirstName = 'Po-Hsun',
         sLastName = 'Cheng')
```

2. Output

```
My full name is Po-Hsun Cheng.
My full name is Po-Hsun Cheng.
```

3.3. Default Argument

1. If we call the function without argument, it uses the default value.
2. Syntax

```
def FunctionName(key1=defaultValue1, key2=defaultValue2, ...):
    # arguments are shown key-value pairs
    # arguments are separated by comma
    # some arguments can be assigned as default values
    # something to do...
```

3.3.1. Ex1: Last Name

1. Code

Listing 3.3.1.2 /src/Function/ArgumentDefault.py

```python
'''
@since: 20150524
@author: cph
'''
def getLastName(sLastName = 'Cheng'):
    print('My last name is', sLastName + '.')

getLastName(sLastName = 'Cheng')
getLastName()
getLastName(sLastName = 'Pang')
```

2. Output

```
My last name is Cheng.
My last name is Cheng.
My last name is Pang.
```

3.4. Arbitrary Argument: *arg

1. If we do not know how many arguments that will be passed into our function, add a * before the parameter name in the function definition.
2. This way the function will receive a tuple of arguments, and can access the items accordingly
3. Syntax

```python
def FunctionName(*arg):
    # arg's date type is tuple
    # something to do...
```

3.4.1. Ex1: Youngest Child

1. Code

Listing 3.4.1.2 /src/Function/ArgumentArbitraryArg.py

```python
'''
@since: 20150524
@author: cph
'''
def myFun(*arg):     # Use tuple
    print(*arg)
    print(arg)
    print(f'The youngest child is {arg[2]}.')

def myFun2(lsIn):    # Use list
    print(lsIn)
    print(f'The youngest child is {lsIn[2]}.')

myFun('Jessie', 'Coco', 'Sean')
```

```
myFun2(['Jessie', 'Coco', 'Sean'])

```

2. Output

```
Jessie Coco Sean
('Jessie', 'Coco', 'Sean')
The youngest child is Sean.
['Jessie', 'Coco', 'Sean']
The youngest child is Sean.
```

3.5. Arbitrary Argument: **kwargs

1. If we do not know how many keyword arguments that will be passed into our function, add two asterisk: [**] before the parameter name in the function definition.
2. This way the function will receive a dictionary of arguments, and can access the items accordingly.
3. So [**kwargs] provides a way to accept flexible keyword arguments in Python functions.
4. The function does NOT have to know ahead of time what keywords will be passed.
5. Syntax

```
def FunctionName(**kwargs):
    # kwargs' date type is dictionary
    # something to do...
```

3.5.1. Ex1: Oldest Child

1. Code

Listing 3.5.1.1 /src/Function/ArgumentArbitraryKwargs.py

```
 1  '''
 2  @since: 20150524
 3  @author: cph
 4  '''
 5  def myFun(**kwargs):
 6      #print(**kwargs)    # Error
 7      print(*kwargs)
 8      print(kwargs)
 9      print(f'The oldest child is {kwargs["sChild1"]}.')
10
11  myFun(sChild1 = 'Jessie',
12        sChild2 = 'Coco',
13        sChild3 = 'Sean')
```

2. Output

```
sChild1 sChild2 sChild3
{'sChild1': 'Jessie', 'sChild2': 'Coco', 'sChild3': 'Sean'}
The oldest child is Jessie.
```

a. If we uncomment Line 6 to print(**kwargs), the error message is shown as below.

```
Traceback (most recent call last):
  File
"W:\_D\Data.cph\Workspace\Py3\Py7.WhoNotes.Py\src\Function\ArgumentArbitraryKwargs.py",
line 11, in <module>
    myFun(sChild1 = 'Jessie',
    ^^^^^^^^^^^^^^^^^^^^^^^^^
  File
"W:\_D\Data.cph\Workspace\Py3\Py7.WhoNotes.Py\src\Function\ArgumentArbitraryKwargs.py",
line 6, in myFun
    print(**kwargs)    # Error
    ^^^^^^^^^^^^^^^
TypeError: 'sChild1' is an invalid keyword argument for print()
```

Note

1. Start: 20170719
2. System Environment

Listing 3.5.1.2 requirements.txt

```
sphinx==7.1.2                           # Sphinx
graphviz>=0.20.1                        # Graphviz
sphinxbootstrap4theme>=0.6.0            # Theme: Bootstrap
sphinx-material>=0.0.35                 # Theme: Material
sphinxcontrib-plantuml>=0.25            # PlantUML
sphinxcontrib.bibtex>=2.5.0             # Bibliography
sphinx-autorun>=1.1.1                   # ExecCode: pycon
sphinx-execute-code-python3>=0.3        # ExecCode
btd.sphinx.inheritance-diagram>=2.3.1   # Diagram
sphinx-copybutton>=0.5.1                # Copy button
sphinx_code_tabs>=0.5.3                 # Tabs
sphinx-immaterial>=0.11.3               # Tabs

#------------------------------------------------------
#-- Library Upgrade Error by Library Itself
#   >> It needs to fix by library owner
#   >> After fixed, we need to try it later
#------------------------------------------------------
pydantic==1.10.10                       # 2.0: sphinx compiler error, 20230701

#------------------------------------------------------
#-- Minor Extension
#------------------------------------------------------
sphinxcontrib.httpdomain>=1.8.1         # HTTP API

#sphinxcontrib-blockdiag>=3.0.0         # Diagram: block
#sphinxcontrib-actdiag>=3.0.0           # Diagram: activity
#sphinxcontrib-nwdiag>=2.0.0            # Diagram: network
#sphinxcontrib-seqdiag>=3.0.0           # Diagram: sequence

#------------------------------------------------------
#-- Still Wait For Upgrading Version
#------------------------------------------------------

#------------------------------------------------------
#-- Still Under Testing
#------------------------------------------------------
#numpy>=1.24.2                          # Figure: numpy

#------------------------------------------------------
#-- NOT Workable
#------------------------------------------------------
#sphinxcontrib.jsdemo==0.1.4    # ExecCode: Need replace add_js_file()
#jupyter-sphinx==0.4.0          # ExecCode: Need gcc compiler
#sphinxcontrib.slide==1.0.0     # Slide: Slideshare
#hieroglyph==2.1.0              # Slide: make slides
#matplotlib>=3.7.1              # Plot: Need Python >= v3.8
#manim==0.17.2                  # Diagram: scipy, numpy need gcc
#sphinx_diagrams==0.4.0         # Diagram: Need GKE access
#sphinx-tabs>=3.4.1                     # Tabs: Conflict w/ sphinx-material
```