```
Python 3.14.0 (tags/v3.14.0:ebf955d, Oct  7 2025, 10:15:03) [MSC v.1944 64 bit
(AMD64)] on win32
Enter "help" below or click "Help" above for more information.
# Extra - non - keyword argument -> isolation
def testFunction(*args):
    print(args, type(args))


testFunction(10, 20, 30, 40)
(10, 20, 30, 40) <class 'tuple'>
t
testFunction()
() <class 'tuple'>
# Step 1: Lets add some formal paraemters before extra-nonkeyword argument
d
def testFunction(a, b, c, *args):
    print(a, b, c)
    print(args)


testFunction(10, 20, 30)
10 20 30
()
testFunction(10, 20, 30, 40)
10 20 30
(40,)
testFunction(10, 20, 30, 40, [100, 200, 300])
10 20 30
(40, [100, 200, 300])
testFunction(10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160)
10 20 30
(40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160)
def testFunctionOne(a, b, c):
    print(a, b, c)


testFunctionOne(10, 20, 30)
10 20 30
testFunctionOne(a=10, b=20, c=30)
10 20 30
testFunction(a=10, b=20, c=30, 40, 50, 60)
SyntaxError: positional argument follows keyword argument
# Step 2: Lets add some formal paraemeters to the right of extra-nonkeyword
argument
def testFunction(a, b, c, *args, p, q):
    print(a, b, c)
    print(args)
    print(p, q)
```

```
testFunction(10, 20, 30, 40, 50, 60, 70, 80, 90) # expectation: a=10, b=20, c=30,
args=(40, 50, 60, 70), p=80, q=90
Traceback (most recent call last):
  File "<pyshell#28>", line 1, in <module>
    testFunction(10, 20, 30, 40, 50, 60, 70, 80, 90) # expectation: a=10, b=20,
c=30, args=(40, 50, 60, 70), p=80, q=90
TypeError: testFunction() missing 2 required keyword-only arguments: 'p' and 'q'
testFunction(10, 20, 30, 40, 50, 60, 70, p=80, q=90)
10 20 30
(40, 50, 60, 70)
80 90
# Default arguments
def testFunction(a=10, b=20, c=30):
    print(a, b, c)


testFunction()
10 20 30
testFunction(a=100)
100 20 30
testFunction(100)
100 20 30
testFunction(10, 20, 300)
10 20 300
testFunction(c=300)
10 20 300
print('Hello,world')
Hello,world
print(10)
10
print(10, 20)
10 20
print(10, 20, 30, 40, sep=':')
10:20:30:40
# Integrating default parameters with positional, extra-non-keyword and keyword
only arguments
def testFunction(a, b, c, *args, x=1.1, y=2.2, p, q):
    print(a, b, c)  # Positional arguments
    print(args)     # Extra-nonkeyword argument
    print(x, y)     # Default arguments
    print(p, q)     # Keyword only arguments


testFunction(10, 20, 30, 40, 50, 60, 70, y=20.25, p=False, q=True)
10 20 30
(40, 50, 60, 70)
1.1 20.25
False True
# Extra keyword argument isolation
def testFunction(**kwargs):
```

```
    print(kwargs)


testFunction()
{}
testFunction(a=10, b=20, c=30)
{'a': 10, 'b': 20, 'c': 30}
# Integrating extra-keyword argument with existing parameter types
def MasterFunction(a, b, c, *args, x=1.1, y=2.2, p, q, **kwargs):
    print('Positional arguments')
...     print(f'a:{a}, b:{b}, c:{c}')
...     print('Extra non-keyword arguments')
...     for i in range(len(args)):
...         print(f'args[{i}]:{args[i]}')
...     print('Defeault arguments')
...     print(f'x:{x}, y:{y}')
...     print('Keyword only arguments')
...     print(f'p:{p}, q:{q}')
...     print('Extra-keyword arguments')
...     for (key, val) in kwargs.items():
...         print(key, val)
...
...
>>> MasterFunction(10, 20, 30, 40, 50, 60, 70, y=20.25, p=False, q=True, u=False,
w=True, v=False)
Positional arguments
a:10, b:20, c:30
Extra non-keyword arguments
args[0]:40
args[1]:50
args[2]:60
args[3]:70
Defeault arguments
x:1.1, y:20.25
Keyword only arguments
p:False, q:True
Extra-keyword arguments
u False
w True
v False
```