*args and **kwargs
In Python

«««

Introduction:

In Python, we can pass a variable number of arguments to a function using special symbols.

There are two special symbols

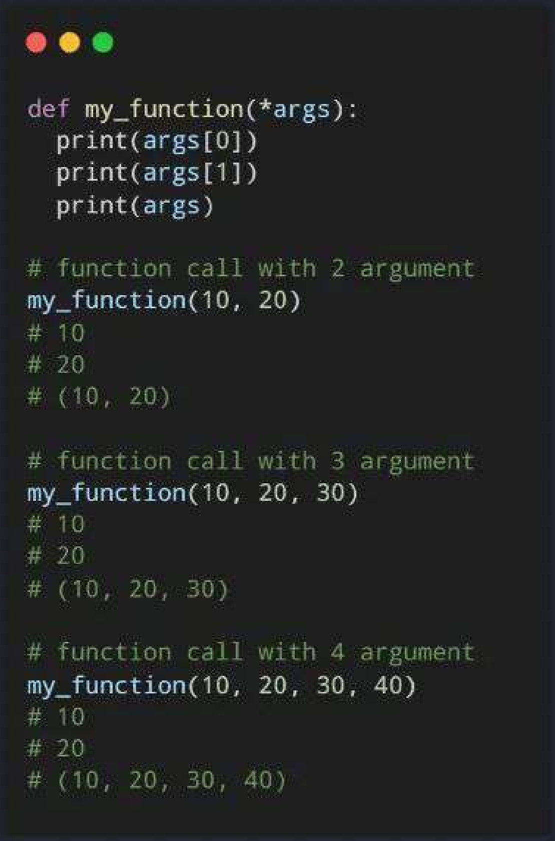
1.)*args (Non-Keyword Arguments)

2.)**kwargs (Keyword Arguments)

Note: We use *args and **kwargs as an argument when we have no doubt about the number of arguments we should pass in a function.

*args:

The special syntax *args in function definitions in python is used to pass a variable number of arguments to a function.



```python
def my_function(*args):
  print(args[0])
  print(args[1])
  print(args)

# function call with 2 argument
my_function(10, 20)
# 10
# 20
# (10, 20)

# function call with 3 argument
my_function(10, 20, 30)
# 10
# 20
# (10, 20, 30)

# function call with 4 argument
my_function(10, 20, 30, 40)
# 10
# 20
# (10, 20, 30, 40)
```

The args is python tuple.

You can use any identifier instead of args but for standards we use args

args stands for arguments.

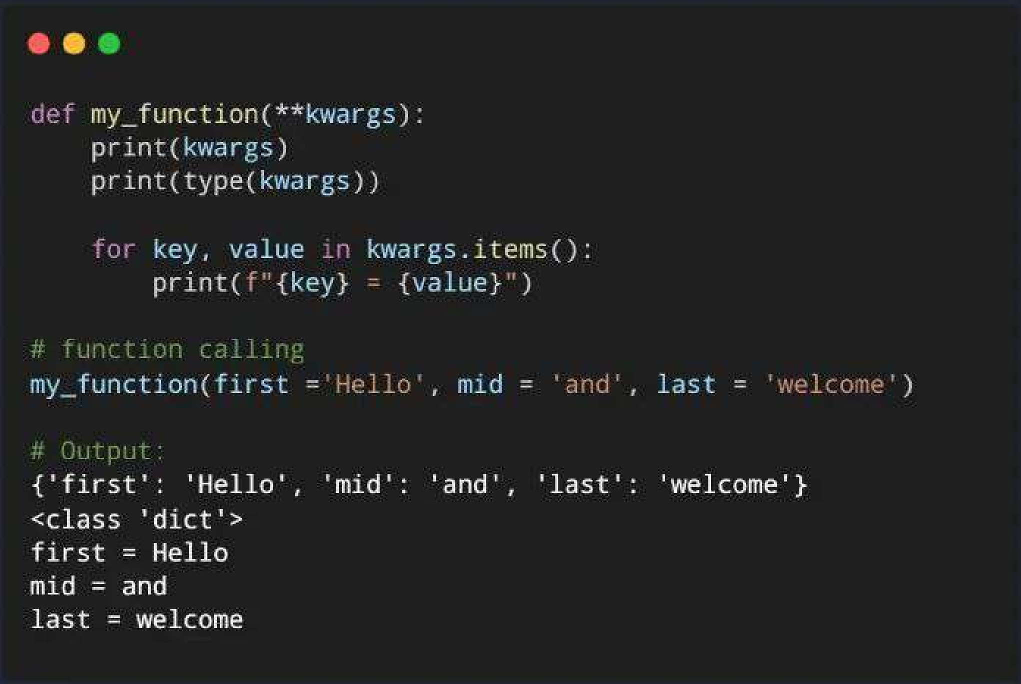
**kwargs:

**kwargs in function definitions is used to pass a keyworded variable-length argument list.

We use the name kwargs with the double star. The reason is that the double star allows us to pass through keyword arguments (and any number of them)

kwargs stands for keyworded arguments.

kwargs Example:



```
def my_function(**kwargs):
    print(kwargs)
    print(type(kwargs))

    for key, value in kwargs.items():
        print(f"{key} = {value}")

# function calling
my_function(first ='Hello', mid = 'and', last = 'welcome')

# Output:
{'first': 'Hello', 'mid': 'and', 'last': 'welcome'}
<class 'dict'>
first = Hello
mid = and
last = welcome
```

Inside function you can use kwargs as a python dictionary where the arguments name is key and its value represents the key value.

same like args, you can use any identifier instead of kwargs but for standards we use kwargs