Overview: Function Parameters

- **Passed by Position**

Parameters identified by their order in the parameter list

- **Passed by Keyword**

Parameters passed in any order, identified by keyword

- **Defaults**

Default values for Parameters

- **Optional by Position (adv)**

Optional parameters passed by position

- **Optional by Keyword (adv)**

Optional parameters passed by keyword

- **Passed by Reference (adv)**

Python's parameter passing

Parameters vs Arguments

```
# Define 'connect'
def connect(dev_IP, username, password):
    ...
    ...
```

parameters

arguments

```
...
# Call 'connect'
session = connect('10.0.0.1','cisco','cisco')
```


Parameters by Position

```
def connect(dev_IP, username, password):
    ...
    ...
```

must match positionally

```
...
session = connect('10.0.0.1','cisco','cisco')
```


Parameters by Keyword

```
def connect(dev_IP, username, password):
    ...
    ...

...
session = connect(username = 'cisco',
                  dev_IP = '10.0.0.1'
                  password = 'cisco')
```

Keywords

any order

Parameters by Position and Keyword

```
def connect(dev_IP, username, password):
    ...
    ...

...
session = connect('10.0.0.1',
                  password = 'cisco',
                  username = 'cisco')
```

positional arguments must be first

any order

keyword arguments

Python provides a rich set of alternatives for passing parameters to a function, including:

- **Positional parameters:** These parameters are in a specific sequence, and the calling code must put their input values (arguments) in the same order as the parameters are listed in the function definition.
- **Keyword parameters:** Python allows parameters to be passed by keyword, meaning that the caller specifies the parameter name, and equals sign, and a value. When done in this way, the parameters can be in any order.
- **Defaults:** Python allows the function to specify default values, so that if the caller does not provide a value for a parameter, the default value is used.
- **Parameter gathering:** Python allows for any number of positional or keyword values to be provided in the function call, and which can be gathered up into a tuple (for positional) or dictionary (for keywords).
- **Parameter passing:** Programming languages had different ways of passing parameters to functions, sometimes referred to as "pass by value" and "pass by reference." Python passes parameters by object reference, since everything is an object.

Parameters versus Arguments

Terminology:

- When you define the list of items that get passed to your function, those items are called 'parameters'.
- When code calls your function, the items it passes are referred to as 'arguments'.

The following example shows the difference. In the function definition at the top, the items are called parameters. In the actual call to the function at the bottom, the values that are passed to the function are called arguments.

```
# Define 'connect'
def connect(dev_IP, username, password):
   ...
   ...
...
# Call 'connect'
session = connect('10.0.0.1','cisco','cisco')
```

Parameters by Position

Positional parameters are parameters which require calling in a specific order, which is the traditional way that other languages have handled the passing of parameters.

In the example below, on the bottom line of code, the calling program provides the device IP address, the username, and the password, in the correct order, as specified in the formal definition of the function, which is shown in the top line.

```
def connect(dev_IP, username, password):
   ...
   ...
...
session = connect('10.0.0.1','cisco','cisco')
```

Parameters by Keyword

Keyword parameters allow the calling program to provide the parameters in any order, provided they specify which parameters are which, by using the keywords in the function call.

In the example below, the function definition at the top is identical to when parameters are passed positionally. In fact, the same function definition is used for either method, positional or keyword.

The difference is seen in the function call at the bottom, where the actual keywords (matching the parameters in the definition) are specified in the actual call.

```
def connect(dev_IP, username, password):
   ...
   ...
...
session = connect(username = 'cisco',
                  dev_IP = '10.0.0.1'
                  password = 'cisco')
```

Parameters by Position and Keyword

It is possible to mix positional and keyword parameters, with the rule that all positional parameters come first, in correct order, followed by keyword arguments (which can be in any order).

In the example below, once again the function definition is the same. The calling code however is passing the first argument positionally, without a keyword; and the remaining arguments are being passed by keyword.

```
def connect(dev_IP, username, password):
   ...
   ...
...
session = connect('10.0.0.1',
                  password = 'cisco',
                  username = 'cisco')
```

Recall that positional parameters must come first, and be in the proper order; and (b) Keyword parameters must come after, and can be in any order.