1. Create function

Start with def keyword which stands for definition.

In [2]:
```
def hello_func():
    pass

print(hello_func())
```

```
None
```

We can leave pass keyword if we don't want to leave any code. The function with empty code return None datatype.

With function, we can repeat the same tasks without repeating one by one.

2. Return the result

The return statement is used when we want to output the result of function.

In [3]:
```
def hello_func():
    return "Hello Function."

str_ = hello_func()
print(str_)
```

```
Hello Function.
```

3. Pass arguments to function

Information can be passed into function as arguments. Arguments are specified after the function name, inside the parentheses. We can add as many arguments as we want. just separate them with a comma.

The default arguments is argument that work as argument when we didn't pass that argument in.

In [7]:
```
def hello_func(greeting='Hello'):
    return f"{greeting} Function."

res1 = hello_func()
res2 = hello_func('Bye')
print(res1, res2, sep="\n")
```

```
Hello Function.
Bye Function.
```

3.1 Special arguments *args and **kwargs

args

*args allows us to pass a variable number of non-keyword arguments to a Python function. In the function, we should use an asterisk(*) before the parameter name to pass a variable number of arguments.

Thus, we're sure that these passed arguments make a tuple inside the function with the same name as the parameter excluding *.

kwargs

**kwargs allows us to pass a variable number of keyword arguments to a Python function. In the function, we use double-asterisk(**) before the parameter name to denote this type of arguments.

In [8]:
```
def student_info(*args, **kwargs):
    print(args)
    print(kwargs)

student_info('Math', 'Art', name='John', age=22)
```

```
('Math', 'Art')
{'name': 'John', 'age': 22}
```