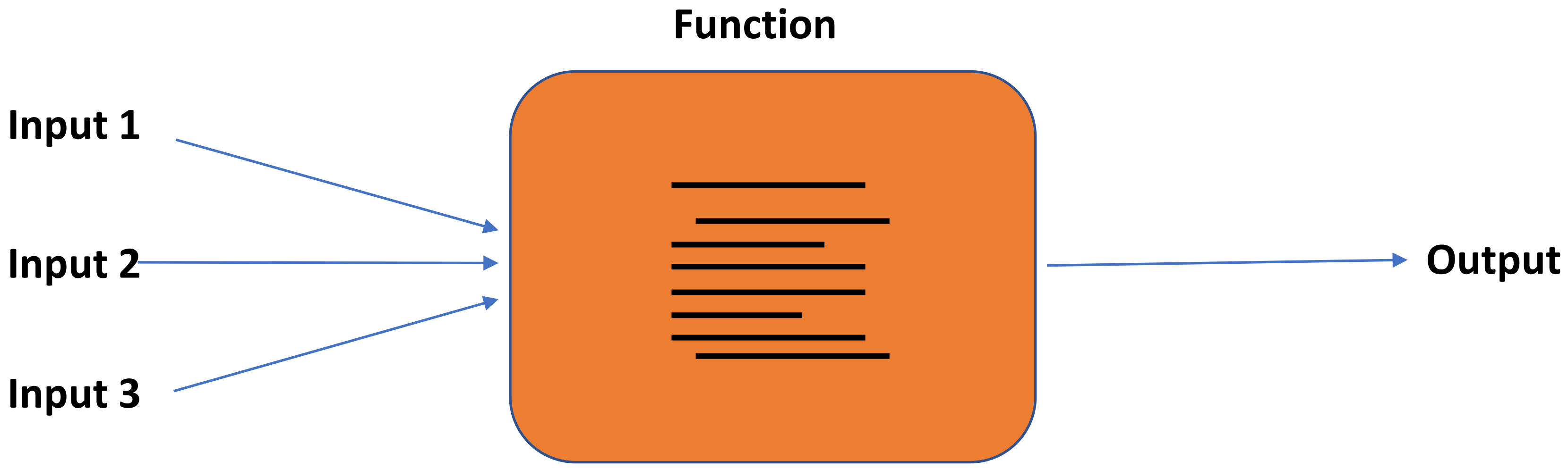
Functions

Function

- Piece of code that's executed **only** when called by its **name**.
- *Usually* takes in input(s), process it, and then produce an output.
- can be reused many times.
- Divides a large program into smaller, manageable chunks.



Definition VS Call

Define	**Call**
Def Function_name (input1, input2):	Statement 1
statement 1	Statement 2
statement 2	.
.	.
.	.
return *output*	Function_name **(** input1_value, input2_value **)**

Function examples

Definition VS Call

Define

```
Def Square ( my_number ):
        X = my_number * my_number
        return X
```

Call

```
Statement 1
Statement 2
.
.
.
Num_A = Square ( 3 ) → 9
Num_B = Square ( 9 ) → 81
```

Definition VS Call

Define

```
Def find_average ( grade1, grade2, grade3):
        total = grade1 + grade2 + grade3
        avg = total / 3
        return avg
```

Call

```
Statement 1
Statement 2
.
.
.
Average_A = find_average ( 93, 86, 90 ) → 89.7
Average_B = find_average ( 95, 97, 72 ) → 88
Average_C = find_average ( 93, 84, 88 ) → 88.3
```

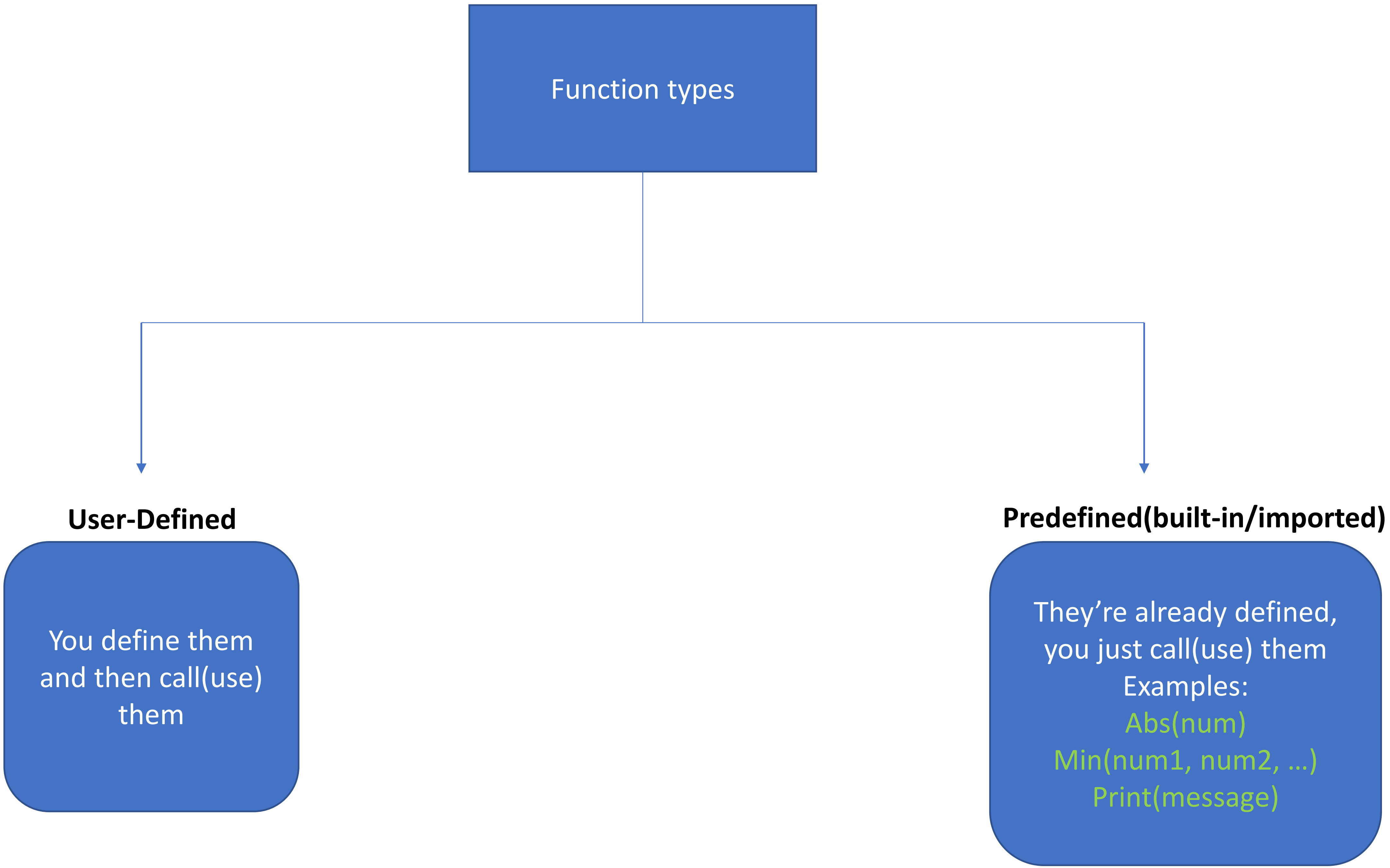
Definition VS Call

Define

```
Def welcome_message ( ):
        print ( " Welcome! " )
```

Call

```
Statement 1
Statement 2
.
.
.
welcome_message ( )
```



Function types

User-Defined

You define them and then call(use) them

Predefined(built-in/imported)

They’re already defined, you just call(use) them

Examples:

Abs(num)

Min(num1, num2, …)

Print(message)

Colon(:) and Indentation(   )

Used together to indicate a code block

```
If ( condition ) :
    action1
    action2
Else :
    action3
    action4
```

```
For item in my_list :
    action1
    action2
    action3

action4
```

```
def my_function( input1 ) :
    If ( condition ) :
        action1
        action2
    Else :
        action3
        action4
    For item in my_list :
        action5
        action6
        action7
```