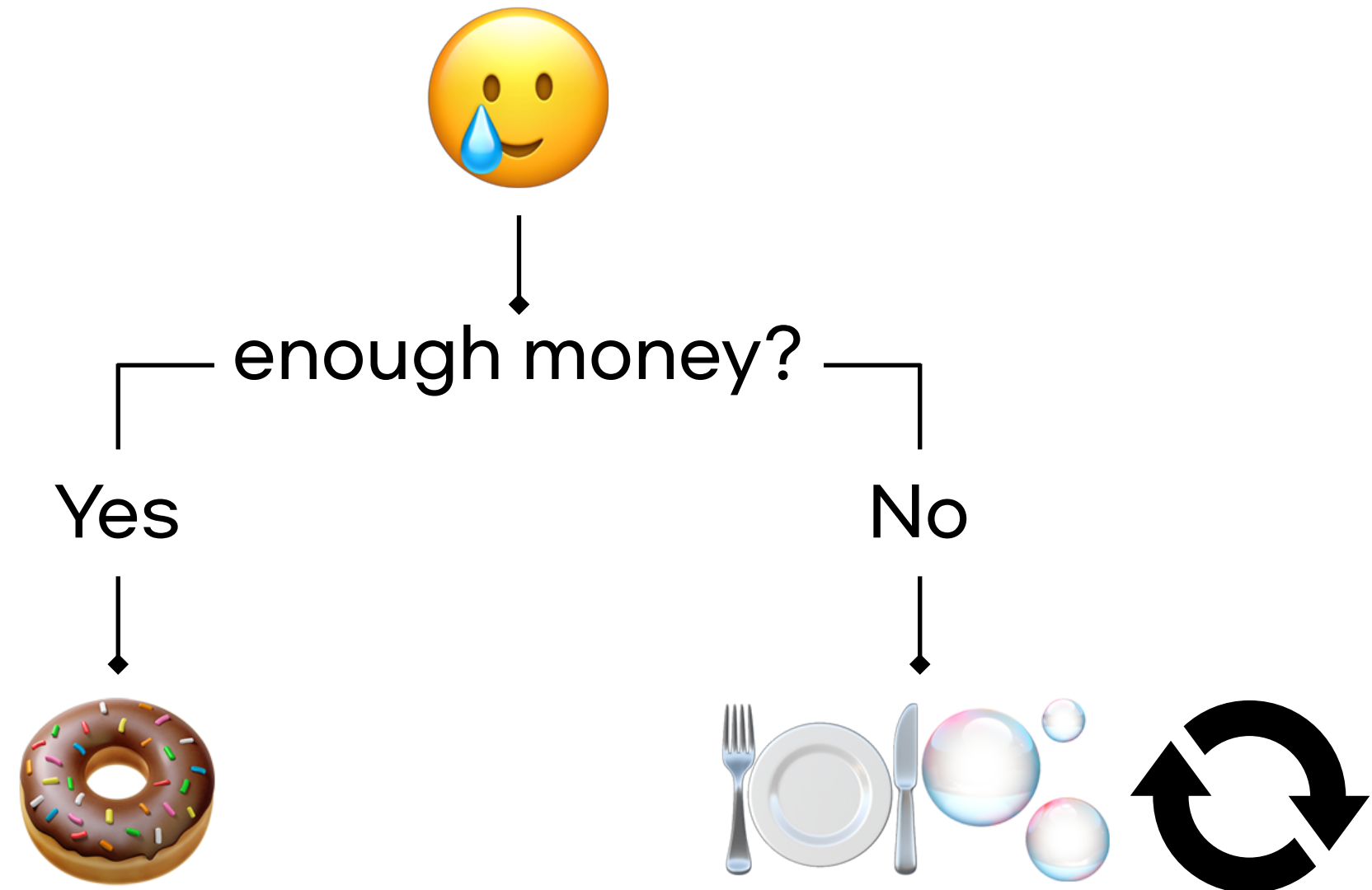


Control Flow & Function

Control Flow



Control Flow



Control Flow

if for while

If Statement

if condition:

└ code to run

If Statement

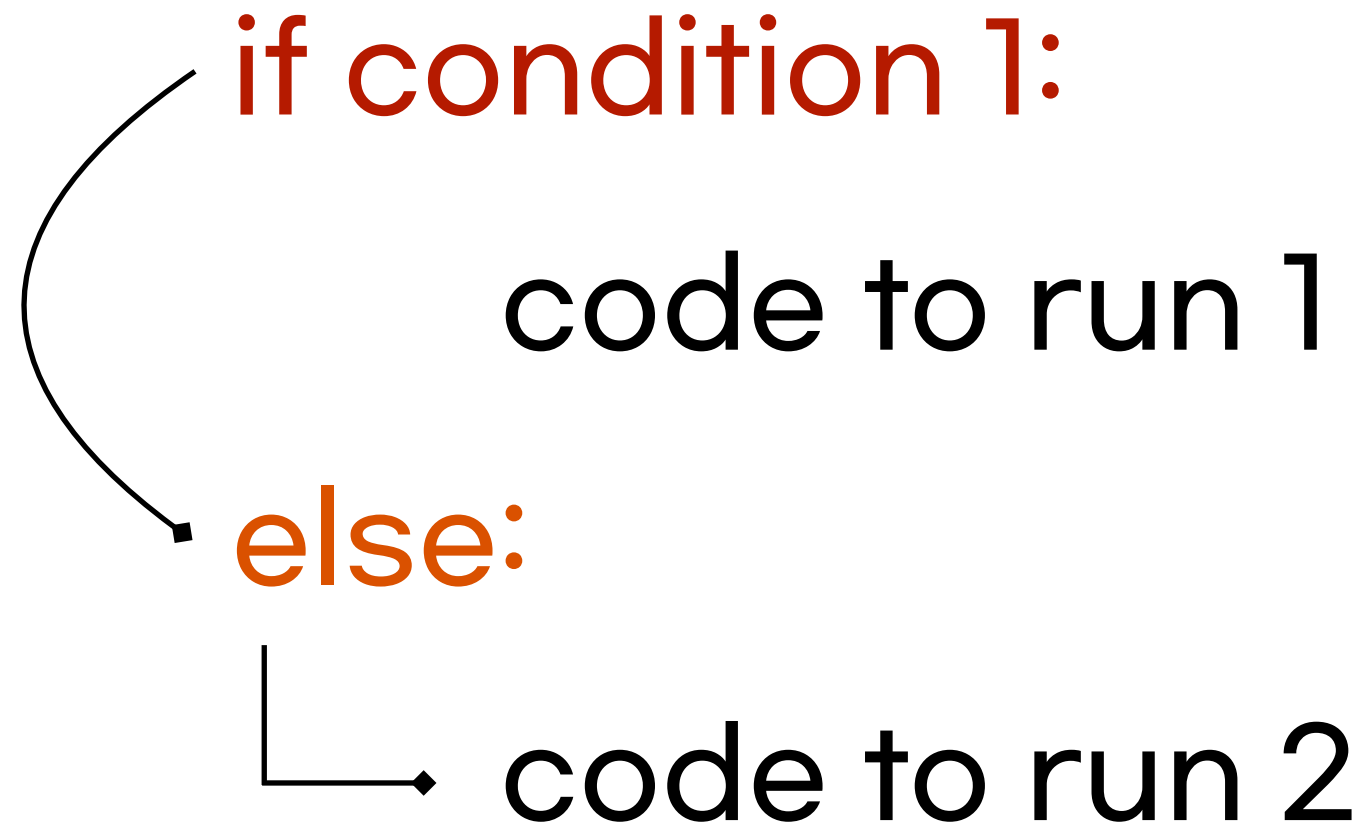
if condition 1:

└→ code to run 1

else:

code to run 2

If Statement



If Statement

if condition 1:

└→ code to run 1

elif condition 2:

code to run 2

else:

code to run 3

If Statement

if condition 1:

code to run 1

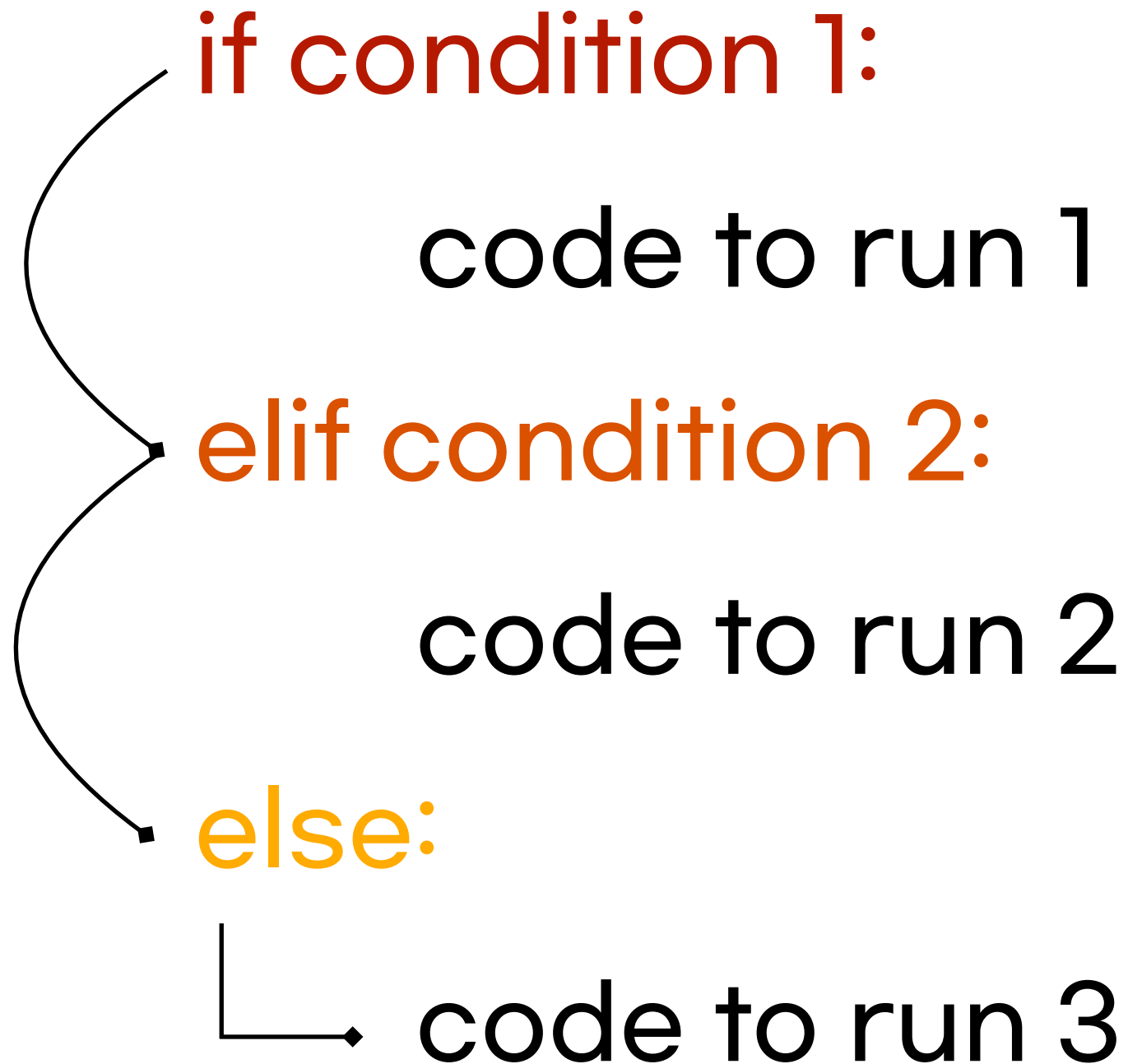
elif condition 2:

└→ code to run 2

else:

code to run 3

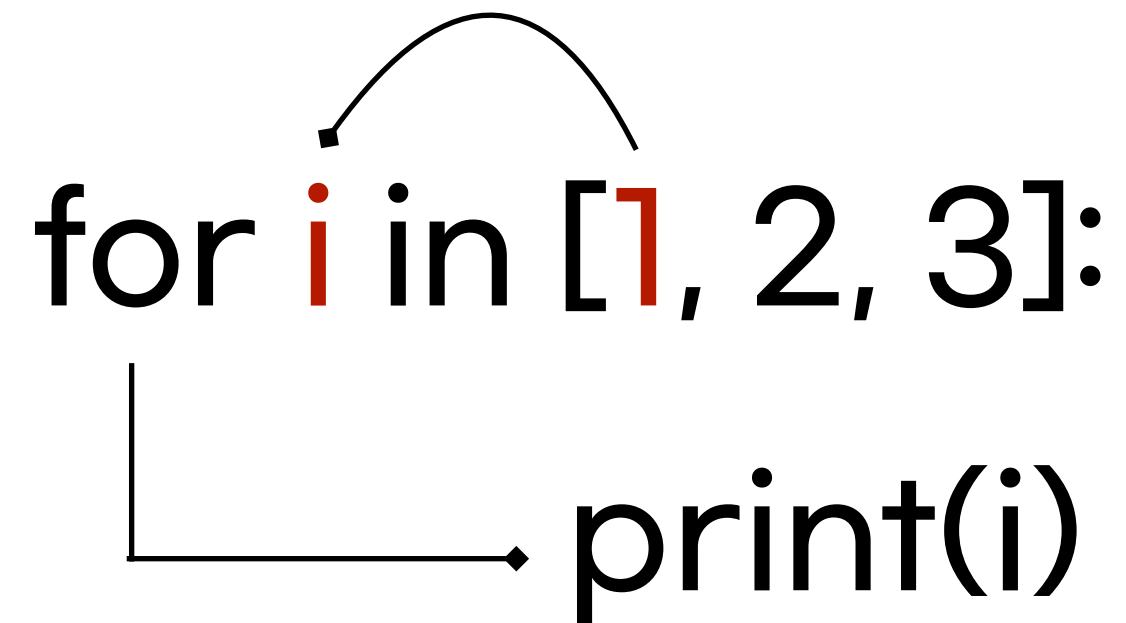
If Statement



For Loop

```
for i in [1, 2, 3]:  
    print(i)
```

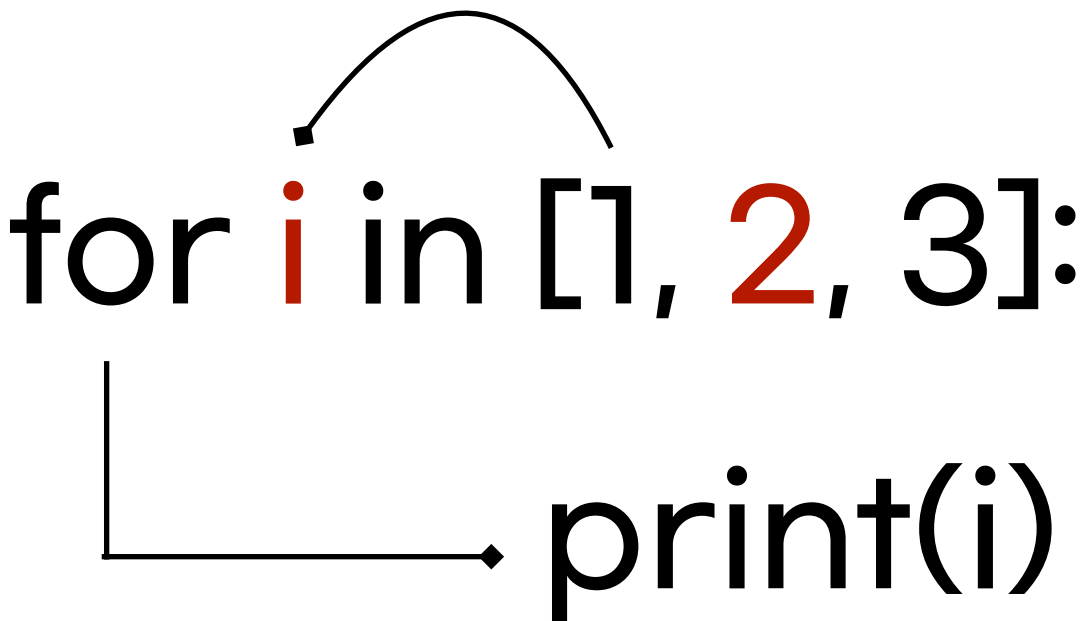
For Loop



```
for i in [1, 2, 3]:  
    print(i)
```

The diagram illustrates a for loop. The variable `i` is highlighted in red. A curved arrow points from the first element of the list, `1`, to the variable `i`. A straight arrow points from the colon at the end of the loop header to the `print(i)` statement, indicating the flow of execution.

For Loop



```
for i in [1, 2, 3]:  
    print(i)
```

For Loop



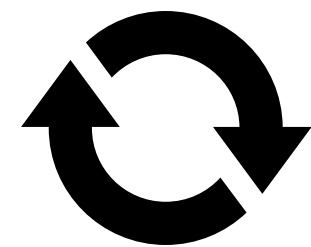
```
for i in [1, 2, 3]:  
    print(i)
```

While Loop

while condition:



code to run



List Comprehension

a = [0, 1, 2, 3]



b = [1, 2, 3, 4]

List Comprehension

```
b = []
```

```
for i in a:
```

```
    b.append(i + 1)
```

List Comprehension

```
b = [for i in a]
```



```
for i in a:
```

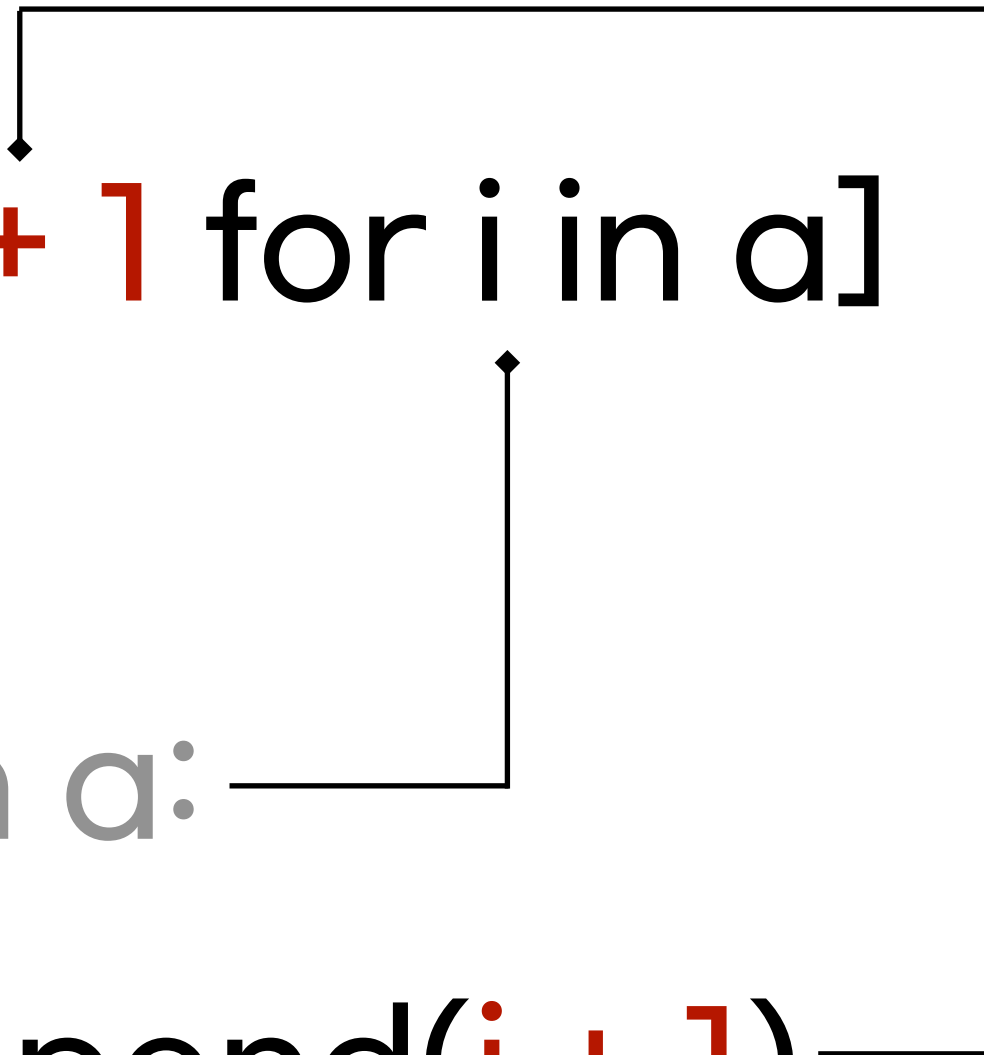
```
    b.append(i + 1)
```

List Comprehension

`b = [i + 1 for i in a]`

`for i in a:`

`b.append(i + 1)`

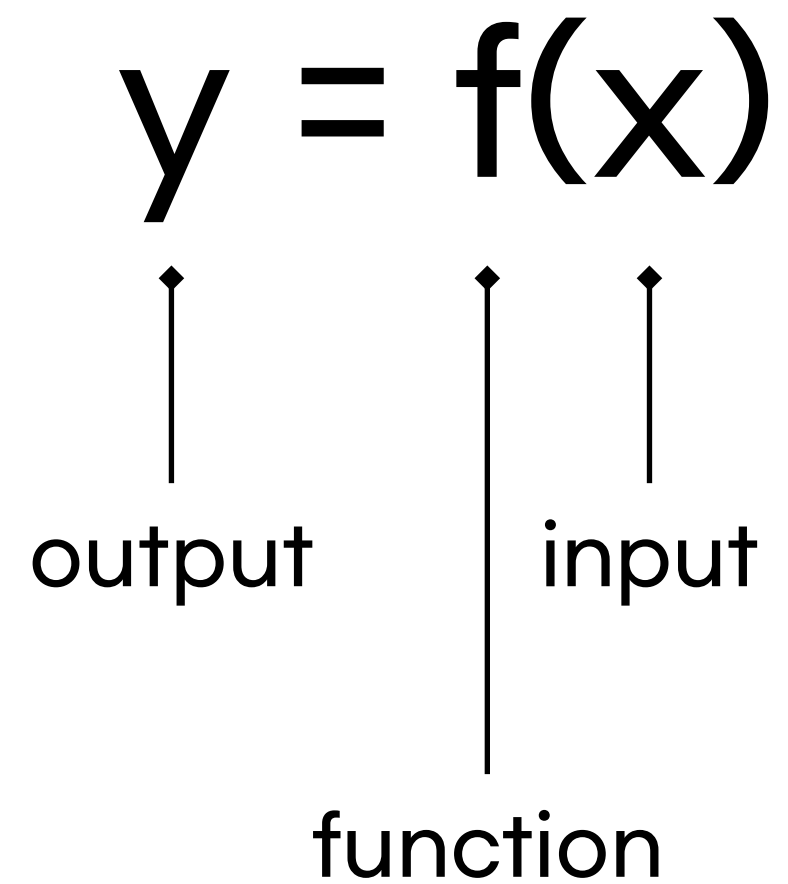


List Comprehension

`a = [0, 1, 2, 3]`

`b = [i + 1 for i in a]`

Function



Function

```
def function_name(input):  
    code_to_run  
    return output
```

Local Scope

```
def function_name(input):  
    code_to_run  
    return output
```

Global Scope

```
a = "global variable"
```

```
def function_name(input):
```

```
    code_to_run
```

```
    return output
```