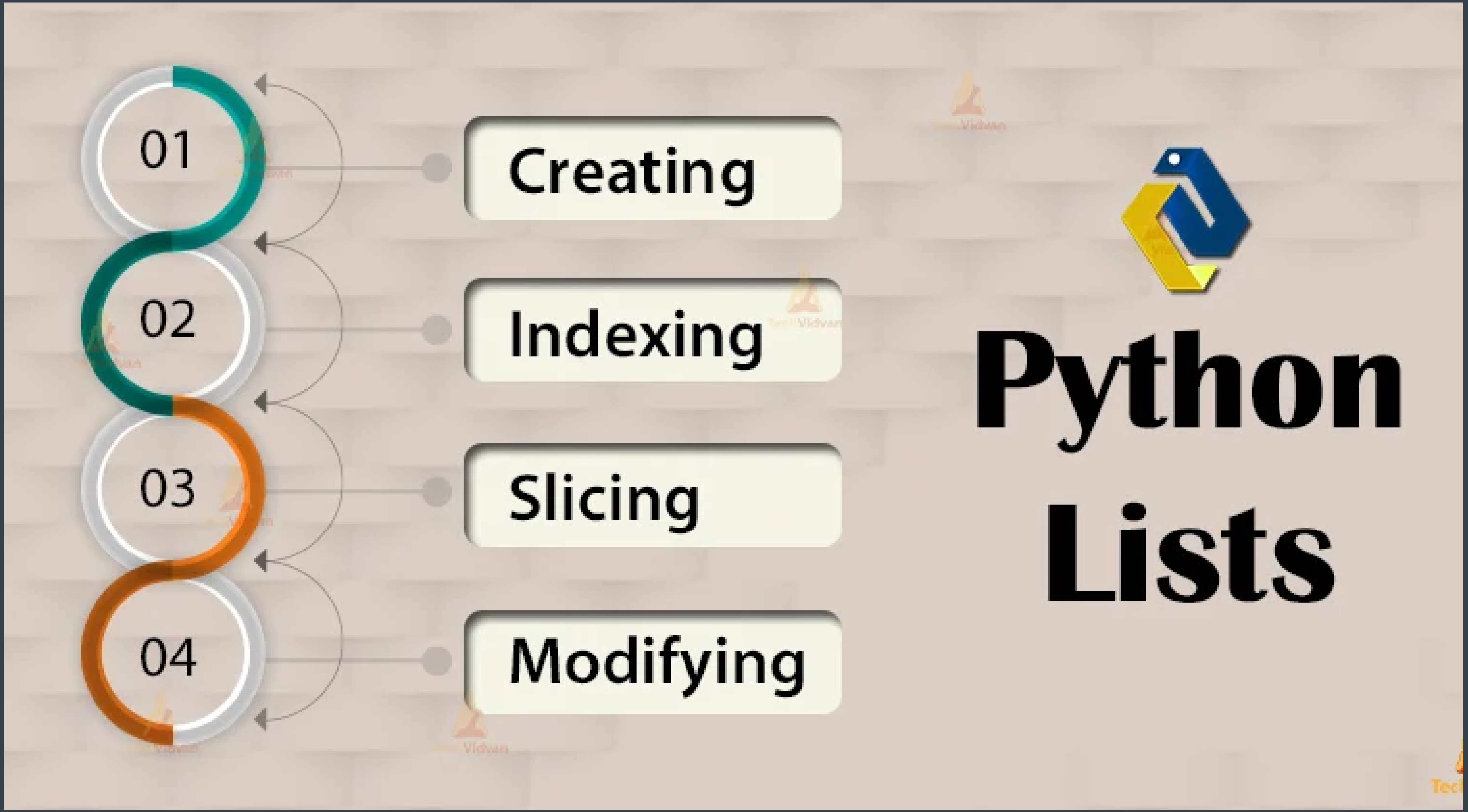
Lists

Defnition

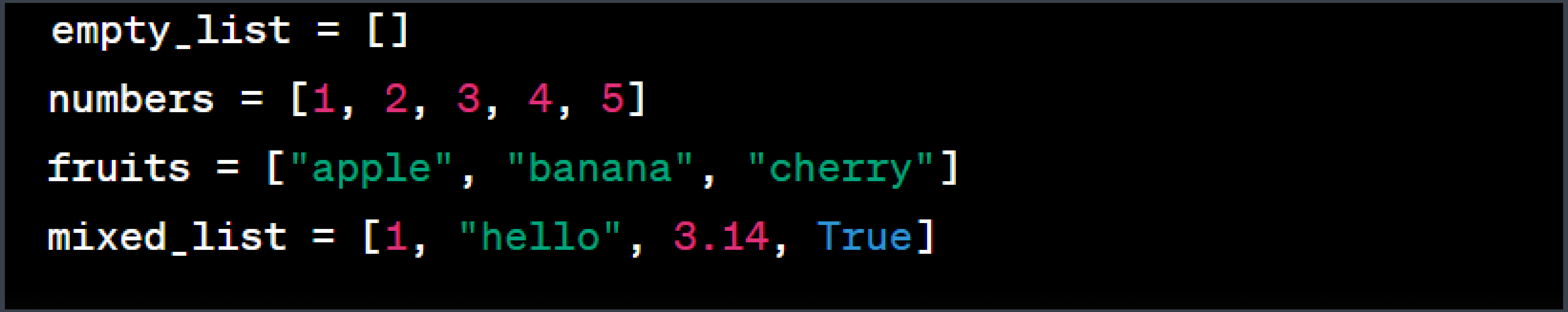
- A list is a versatile and mutable data structure used to store a collection of items.
- Defined using square brackets []
- Contain elements of different data types
- Allow duplicates.



01
Creating
02
Indexing
03
Slicing
04
Modifying
Python
Lists

Creating Lists

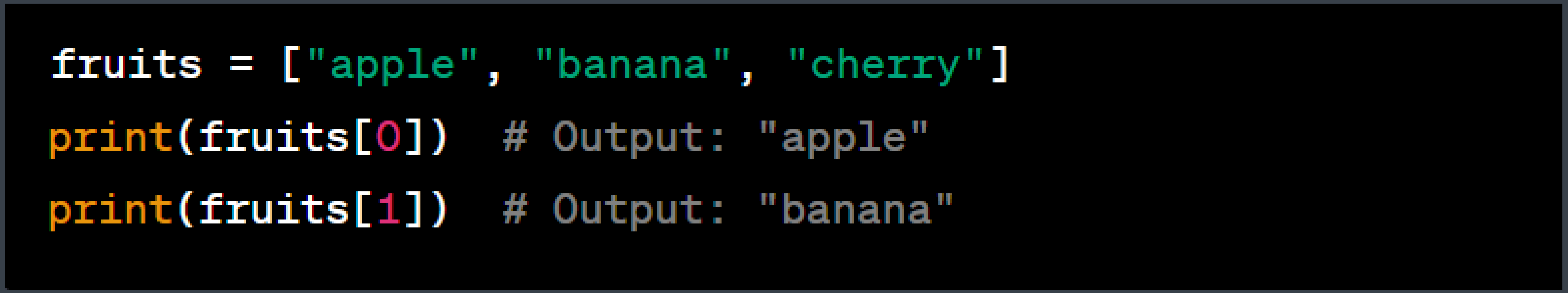
you enclose the elements inside square brackets, separated by commas.



```
empty_list = []
numbers = [1, 2, 3, 4, 5]
fruits = ["apple", "banana", "cherry"]
mixed_list = [1, "hello", 3.14, True]
```

Accessing Elements

You can access individual elements of a list using their index. Indexing in Python starts from 0.

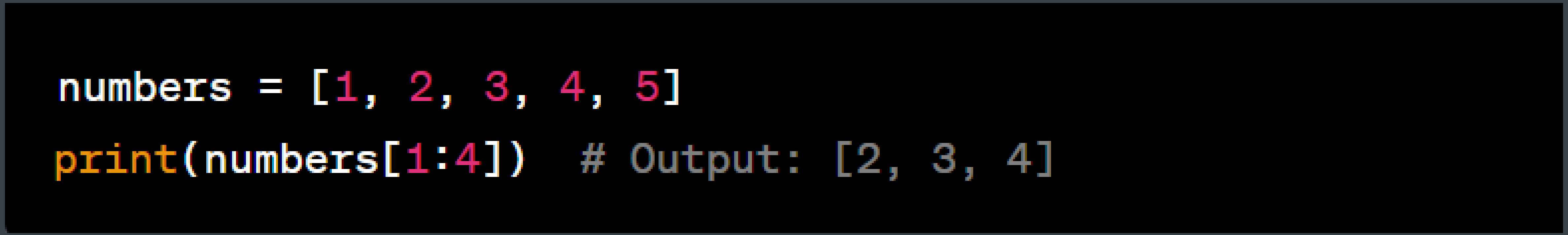


```python
fruits = ["apple", "banana", "cherry"]
print(fruits[0])  # Output: "apple"
print(fruits[1])  # Output: "banana"
```

Python also supports negative indexing, where -1 refers to the last element

Slicing Lists

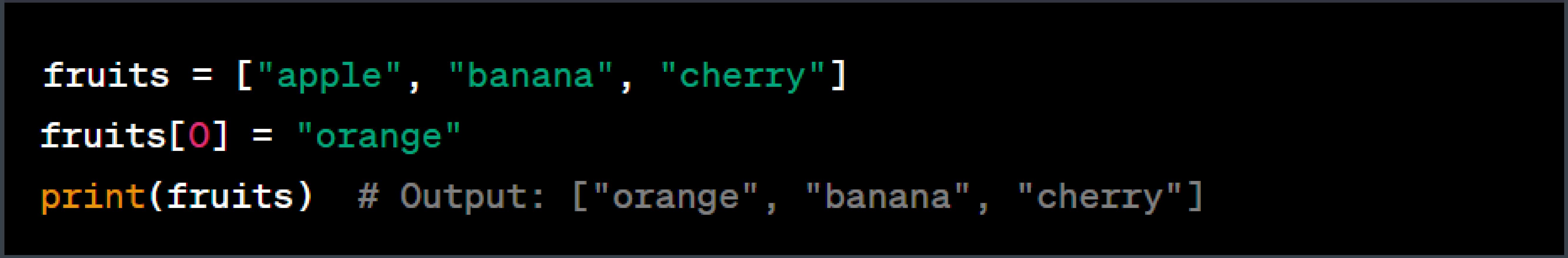
You can extract a portion of a list using slicing. Slicing allows you to create a new list with a subset of elements.



```
numbers = [1, 2, 3, 4, 5]
print(numbers[1:4])  # Output: [2, 3, 4]
```

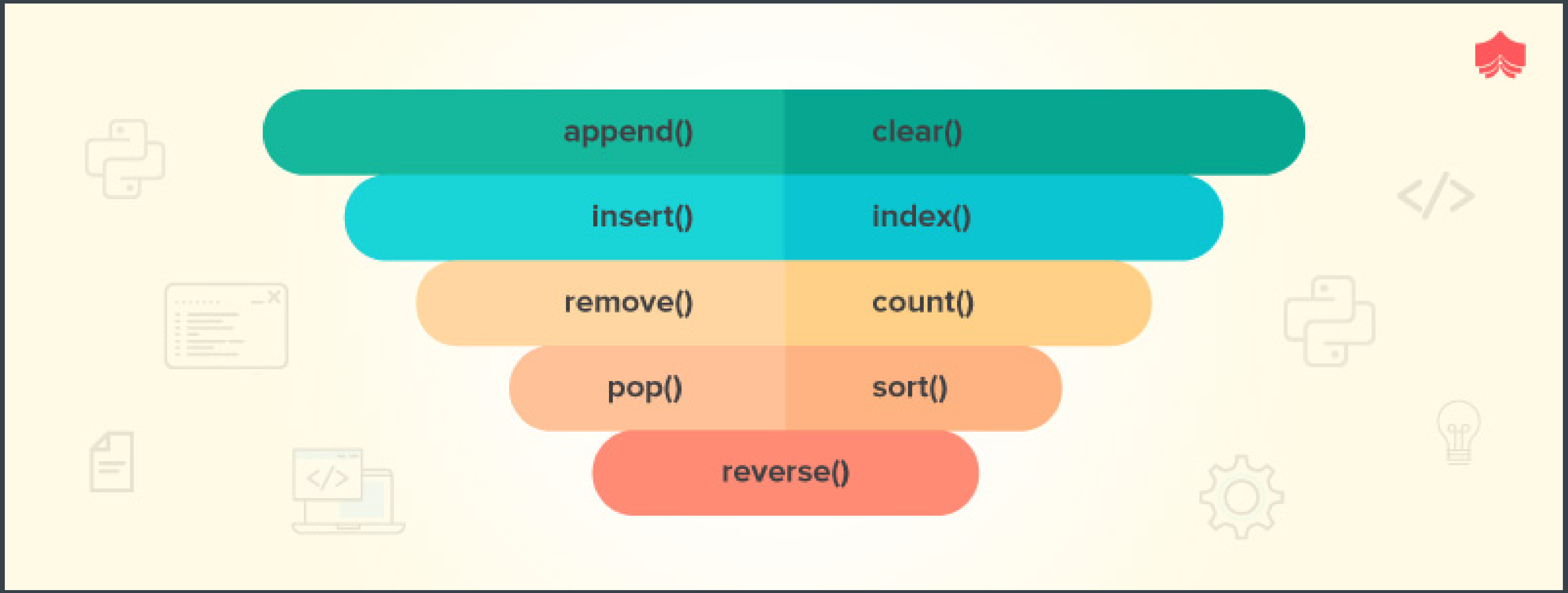
Modifying Elements

You can modify individual elements in a list by accessing them using their index and then assigning a new value.



```
fruits = ["apple", "banana", "cherry"]
fruits[0] = "orange"
print(fruits)  # Output: ["orange", "banana", "cherry"]
```

Methods



append(): Adds an element to the end of the list.

insert(): Inserts an element at a specific index.

remove(): Removes the first occurrence of a specified element.

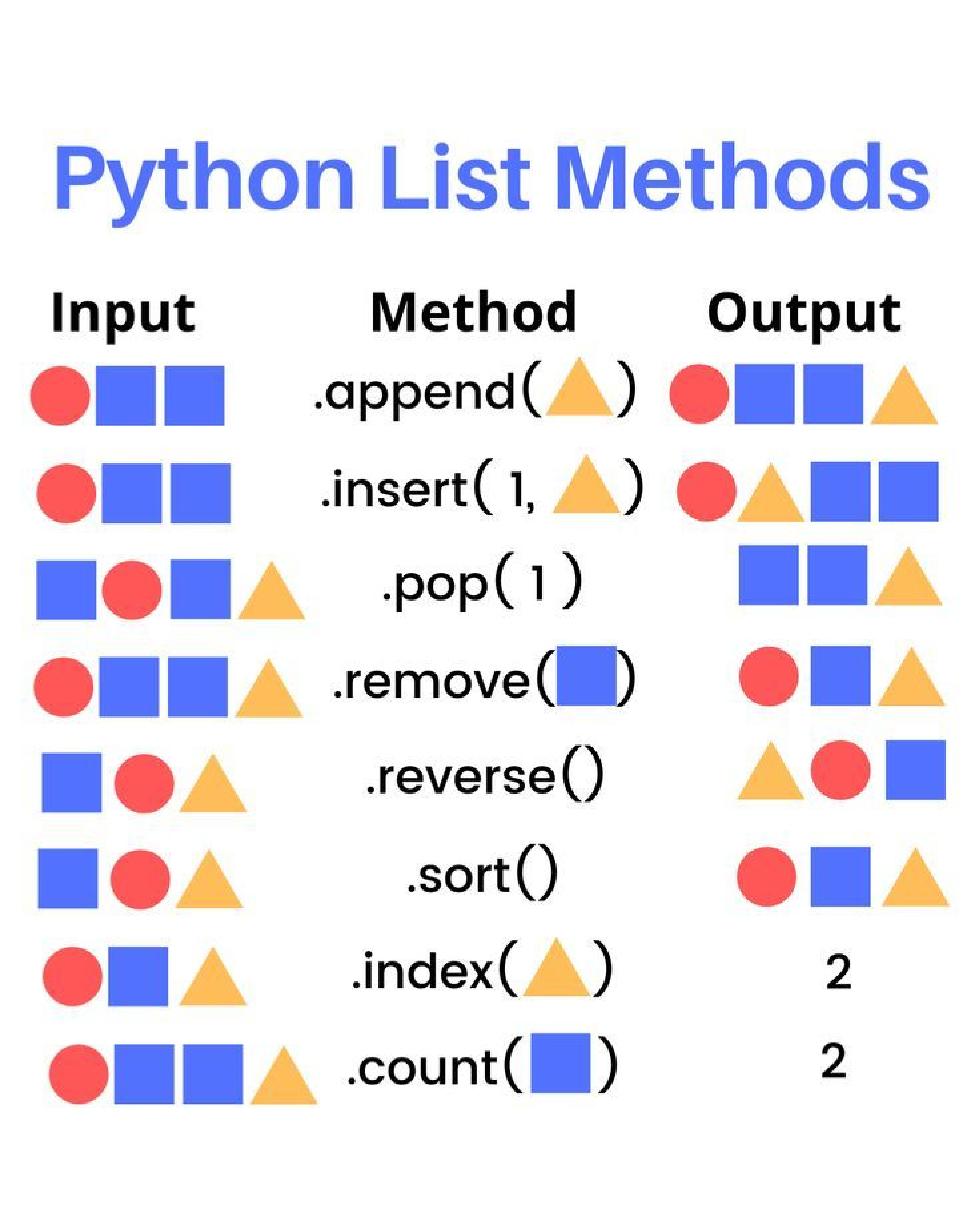
pop(): Removes and returns the element at a specified index (or the last element if no index is given).

index(): Returns the index of the first occurrence of a specified element.

count(): Returns the number of occurrences of a specified element in the list.

sort(): Sorts the list in ascending order.

reverse(): Reverses the order of the elements in the list.

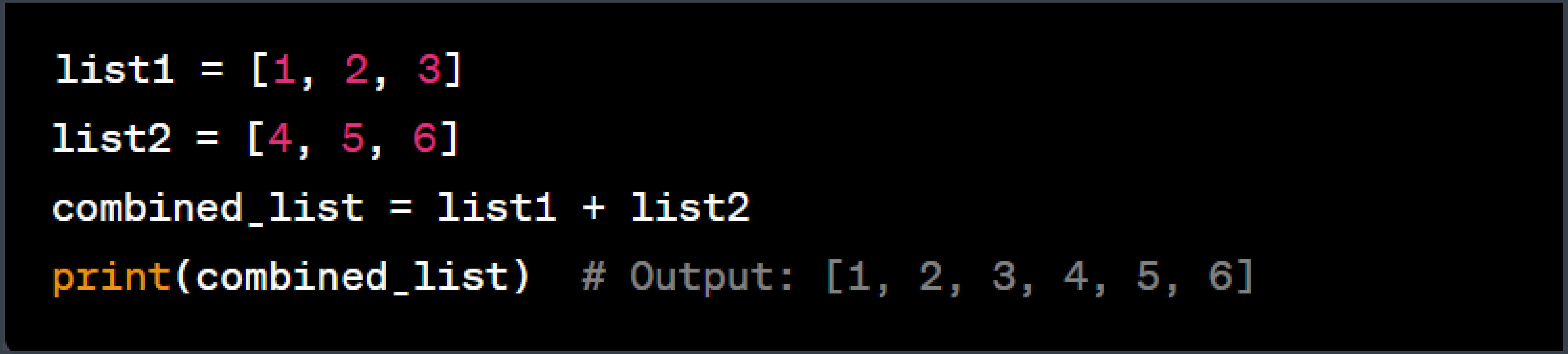


Python List Methods

Input	Method	Output
●■■	.append(▲)	●■■▲
●■■	.insert(1, ▲)	●▲■■
■●■▲	.pop(1)	■■▲
●■■▲	.remove(■)	●■▲
■●▲	.reverse()	▲●■
■●▲	.sort()	●■▲
●■▲	.index(▲)	2
●■■▲	.count(■)	2

List Concatenation

Lists can be concatenated using the + operator, which creates a new list containing elements from both lists.



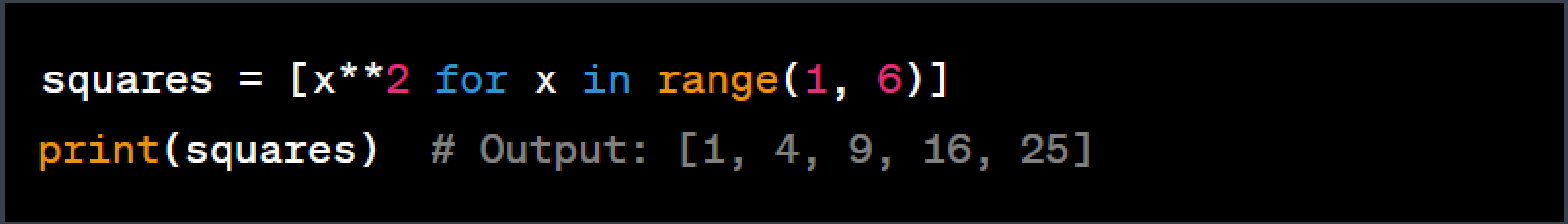
```
list1 = [1, 2, 3]
list2 = [4, 5, 6]
combined_list = list1 + list2
print(combined_list)  # Output: [1, 2, 3, 4, 5, 6]
```

List Comprehensions

List comprehensions provide a concise way to create lists using a single line of code.

Syntax

new_list = [expression for item in iterable if condition]



```
squares = [x**2 for x in range(1, 6)]
print(squares)  # Output: [1, 4, 9, 16, 25]
```