14

Lists in Python-3

by Gladden Rumao

C01: Problem Solving with Programming

Newton School of Technology

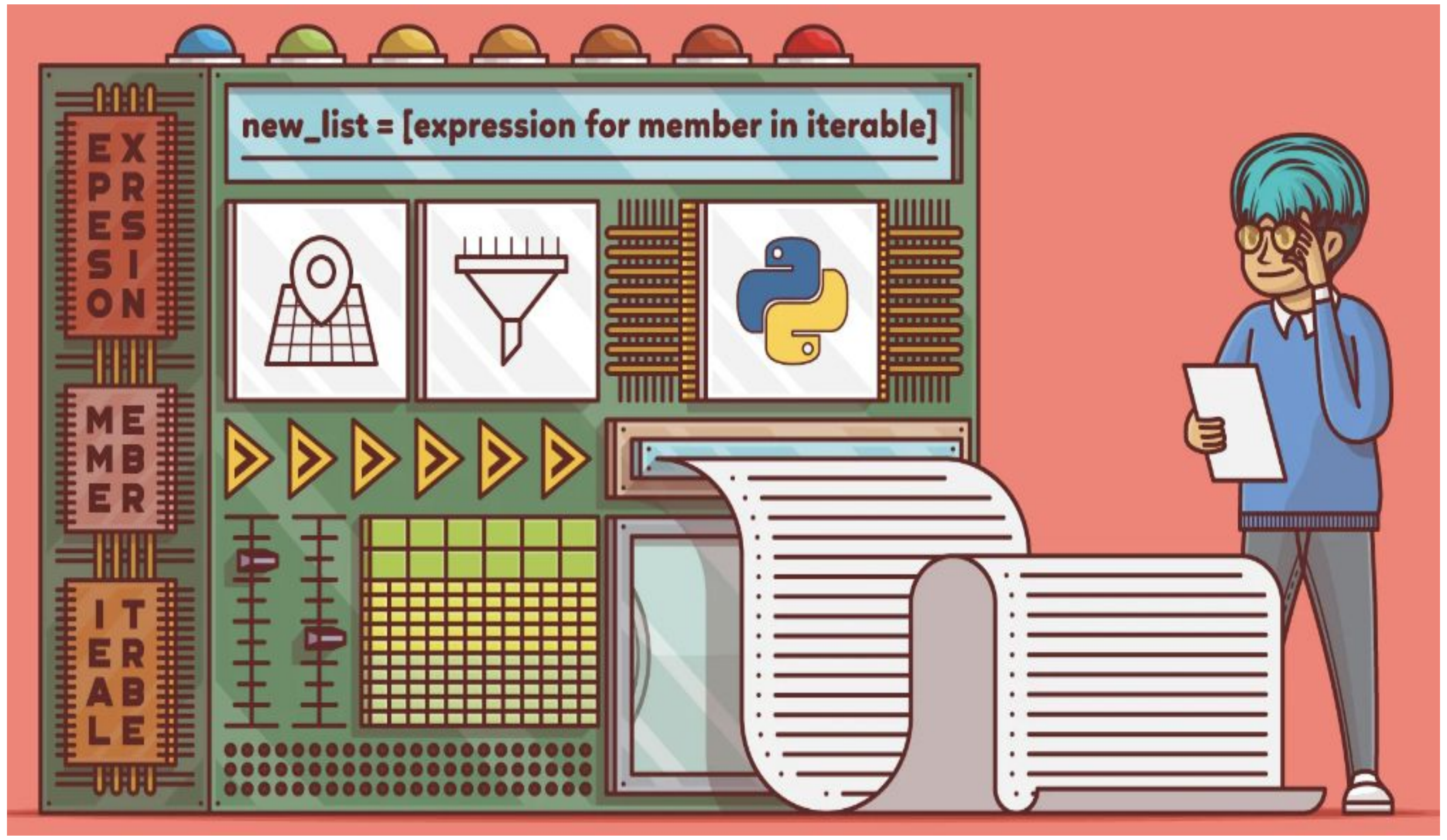
List Comprehensions

List Comprehensions :

List comprehensions provide a concise way to create lists in Python.

They allow you to **create a new list** by **applying an expression** to each item.

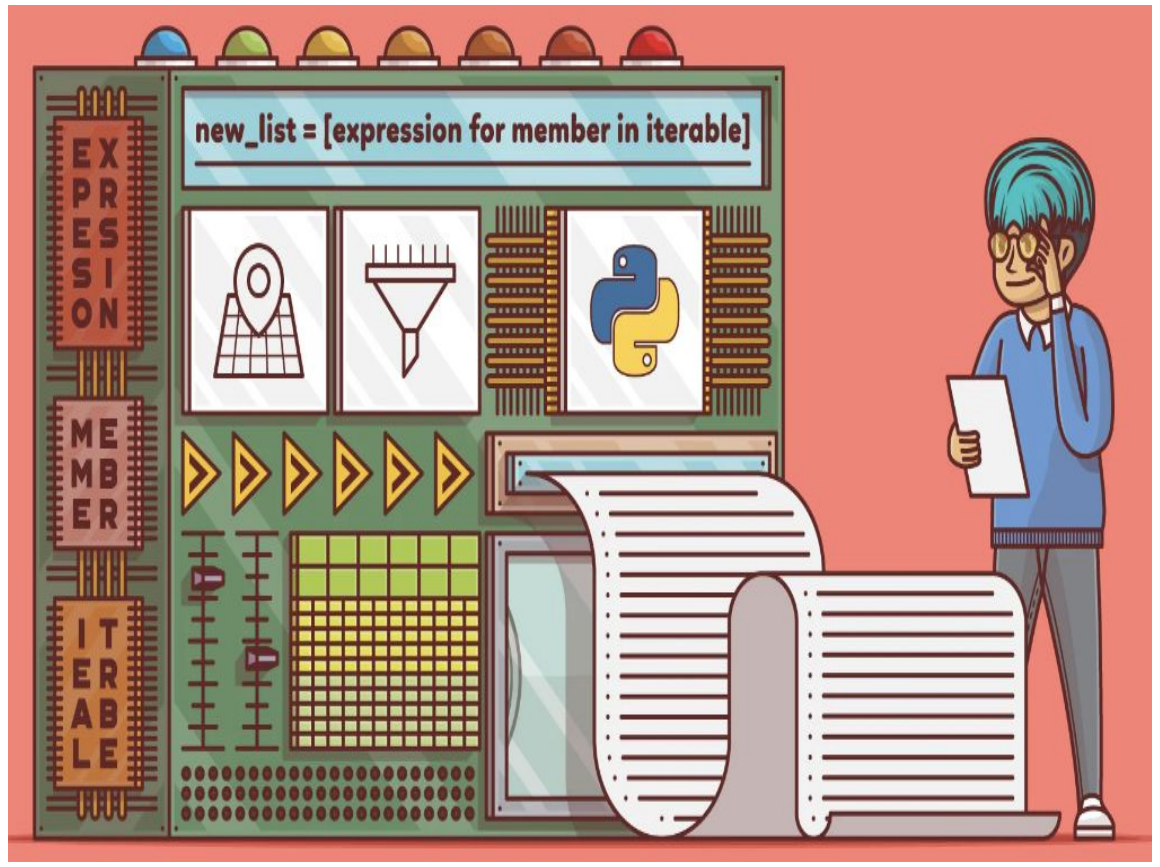
Enables filtering the items **based on a condition**.



List Comprehensions :

Syntax

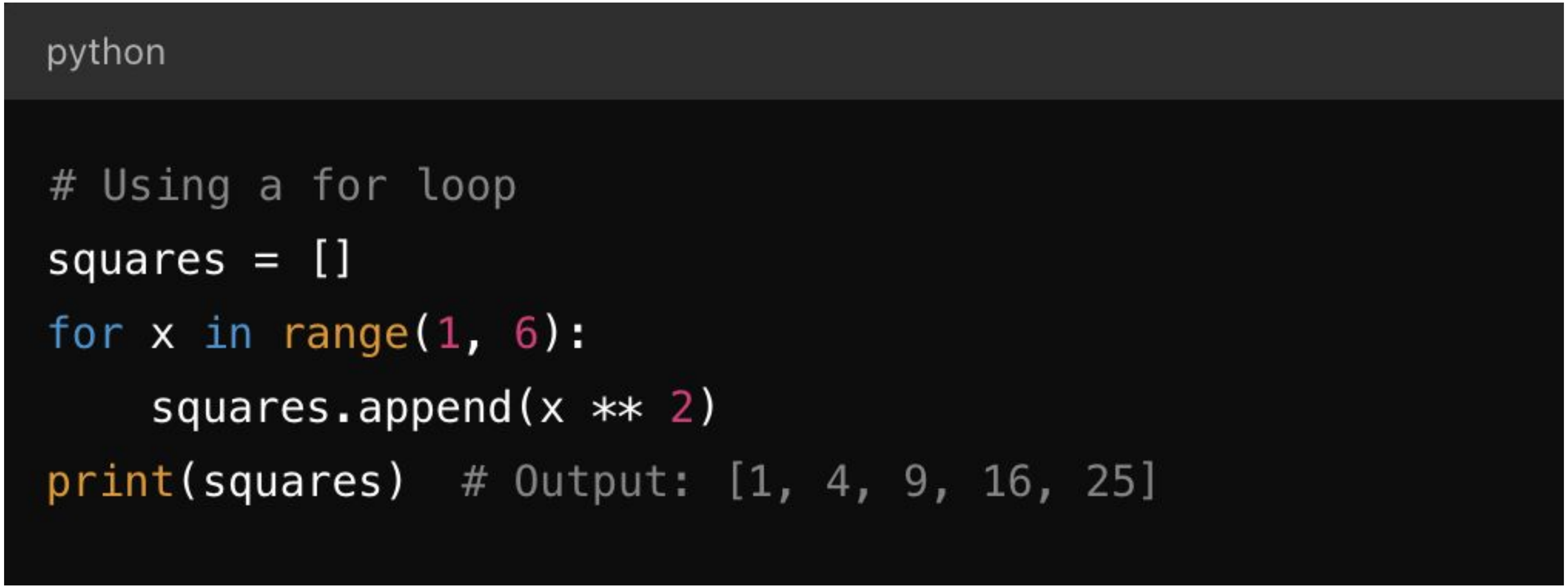
```
[expression for item in iterable if condition]
```



`expression`: This is the expression to **evaluate and include** in the new list.

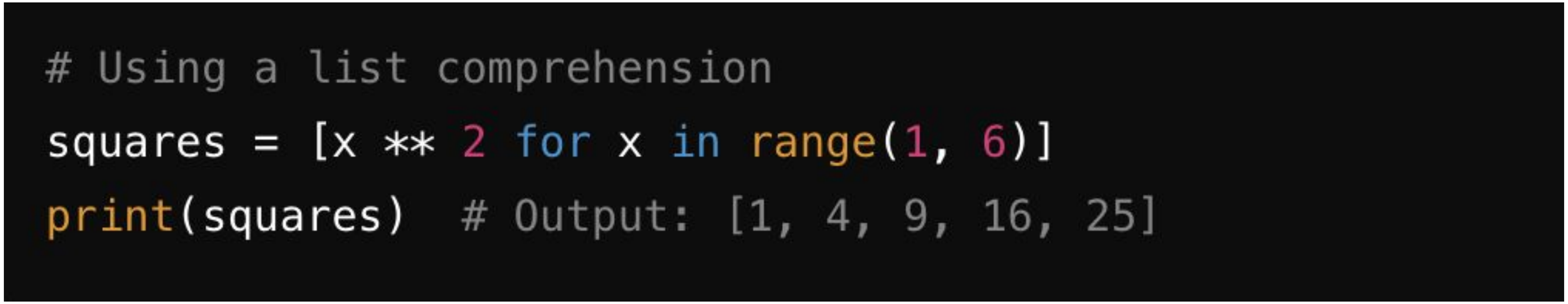
`iterable`: This is the **sequence** of items that you want to iterate over (e.g., a list, tuple, range, etc.).

List Comprehensions -1 :


python

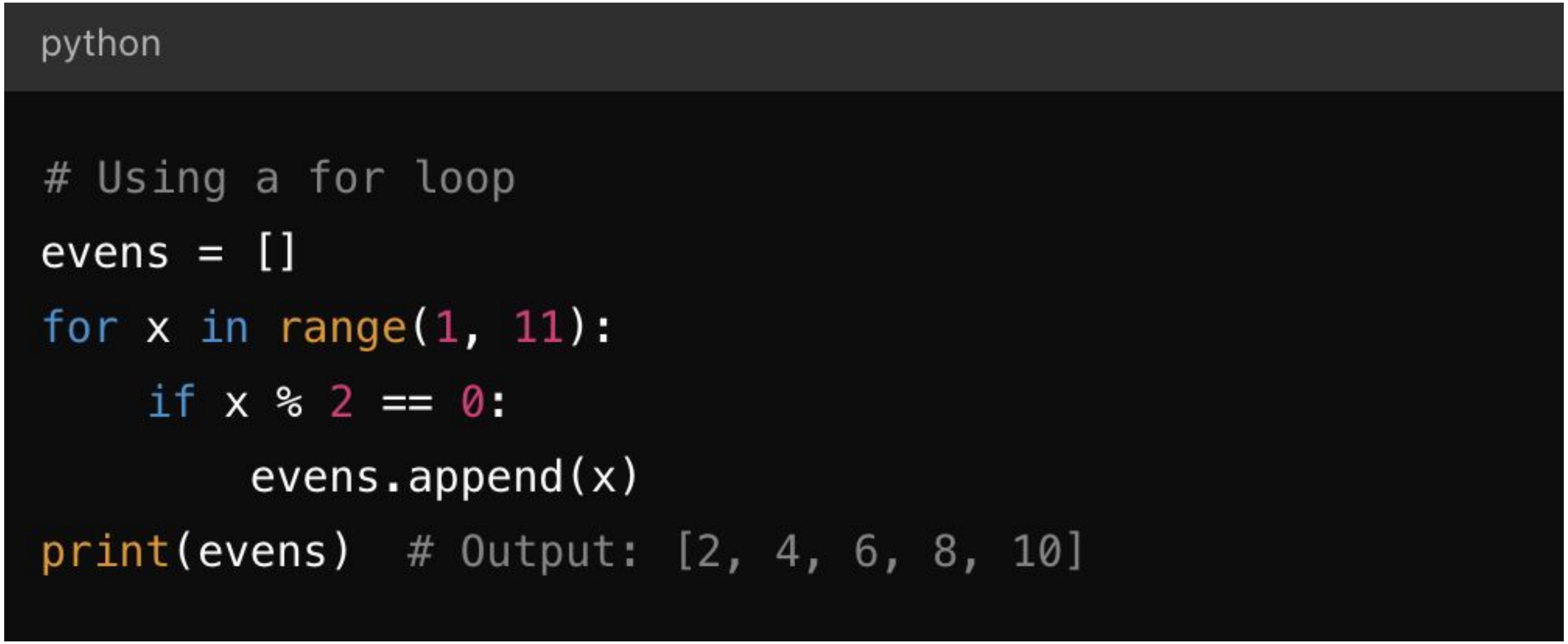
```python
# Using a for loop
squares = []
for x in range(1, 6):
    squares.append(x ** 2)
print(squares)  # Output: [1, 4, 9, 16, 25]
```

List Comprehensions -1 :



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

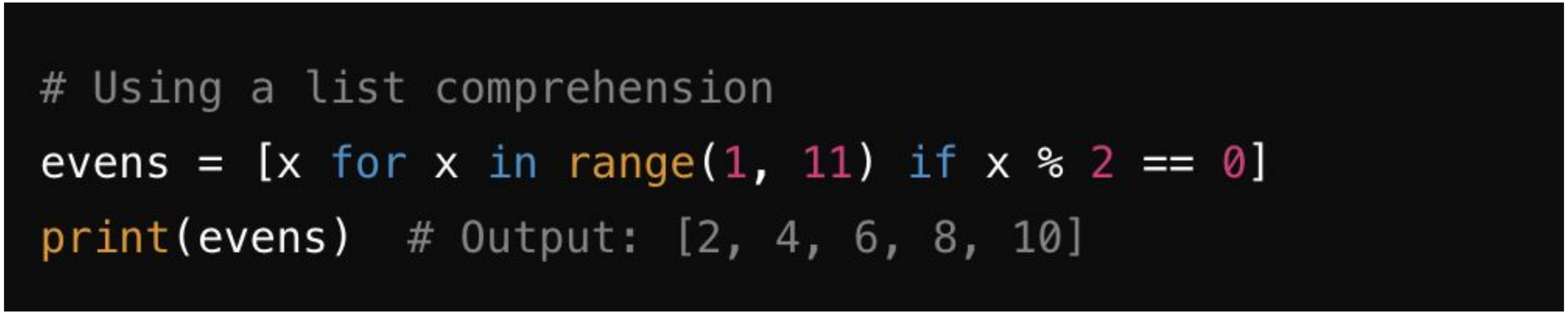
List Comprehensions –2 :



python

```python
# Using a for loop
evens = []
for x in range(1, 11):
    if x % 2 == 0:
        evens.append(x)
print(evens)  # Output: [2, 4, 6, 8, 10]
```

List Comprehensions –2 :

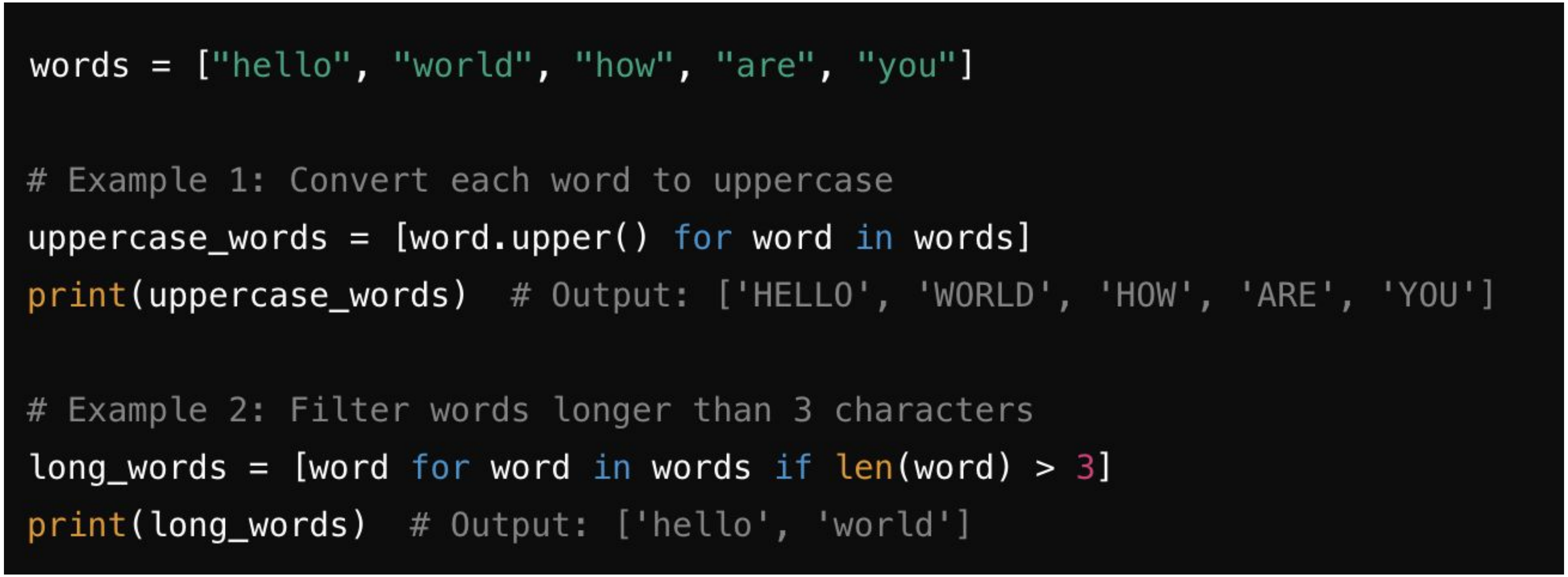


```
# Using a list comprehension
evens = [x for x in range(1, 11) if x % 2 == 0]
print(evens)  # Output: [2, 4, 6, 8, 10]
```

Newton School of Technology

Lets Practice on the Playground!

List Comprehensions –3 :



```python
words = ["hello", "world", "how", "are", "you"]

# Example 1: Convert each word to uppercase
uppercase_words = [word.upper() for word in words]
print(uppercase_words)  # Output: ['HELLO', 'WORLD', 'HOW', 'ARE', 'YOU']

# Example 2: Filter words longer than 3 characters
long_words = [word for word in words if len(word) > 3]
print(long_words)  # Output: ['hello', 'world']
```

Newton School of Technology

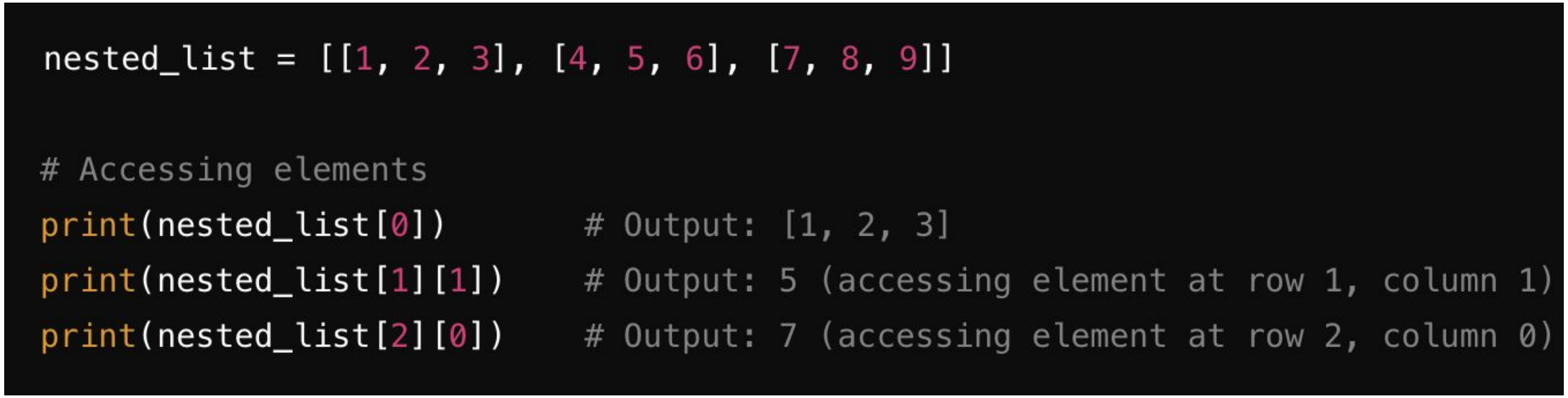
Question 1 – Square of Evens

Newton School of Technology

Nested Lists

Nested Lists

Nested lists are lists that contain one or more other **lists as their elements**.

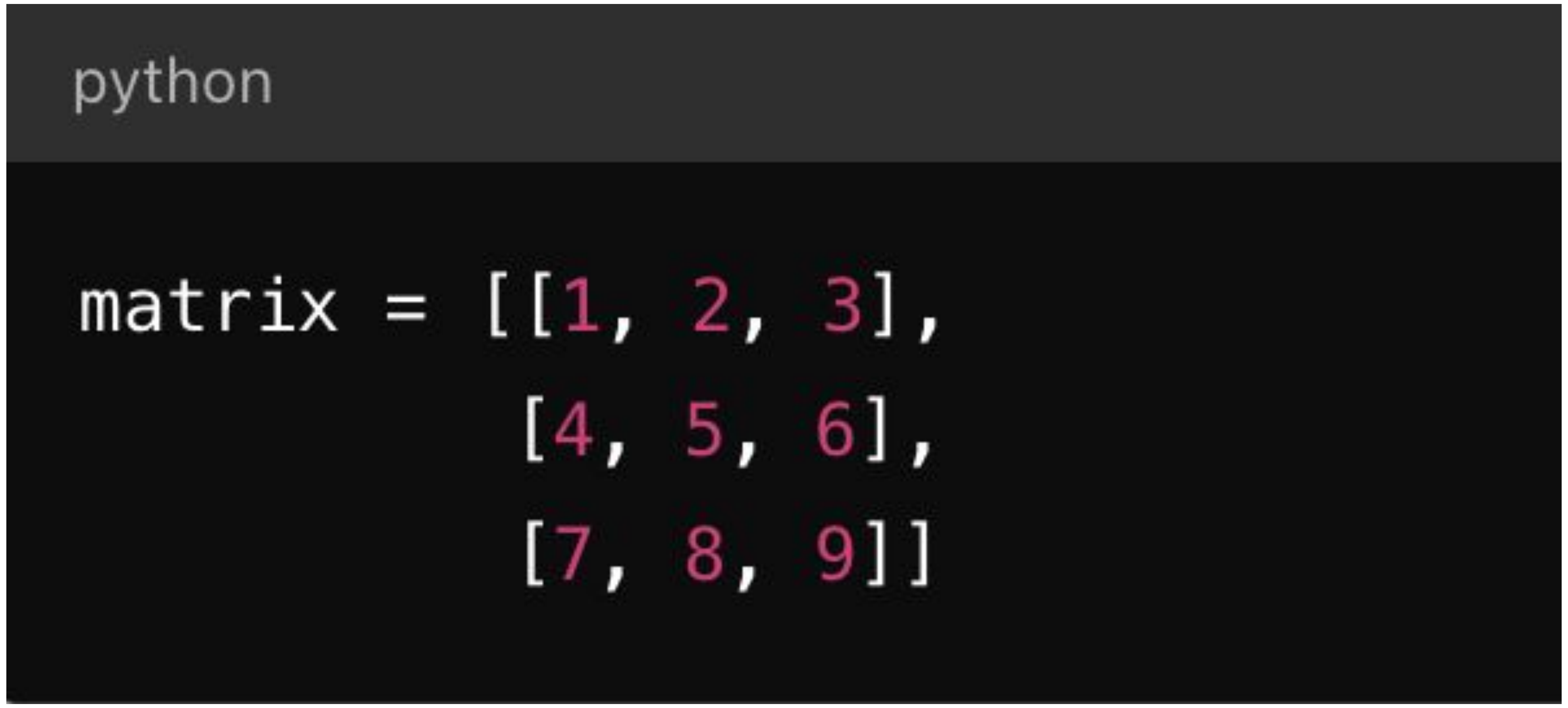


```python
nested_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

# Accessing elements
print(nested_list[0])       # Output: [1, 2, 3]
print(nested_list[1][1])    # Output: 5 (accessing element at row 1, column 1)
print(nested_list[2][0])    # Output: 7 (accessing element at row 2, column 0)
```

Use Cases : Matrices

Nested lists are commonly used to represent **matrices** in mathematics or grids of data.

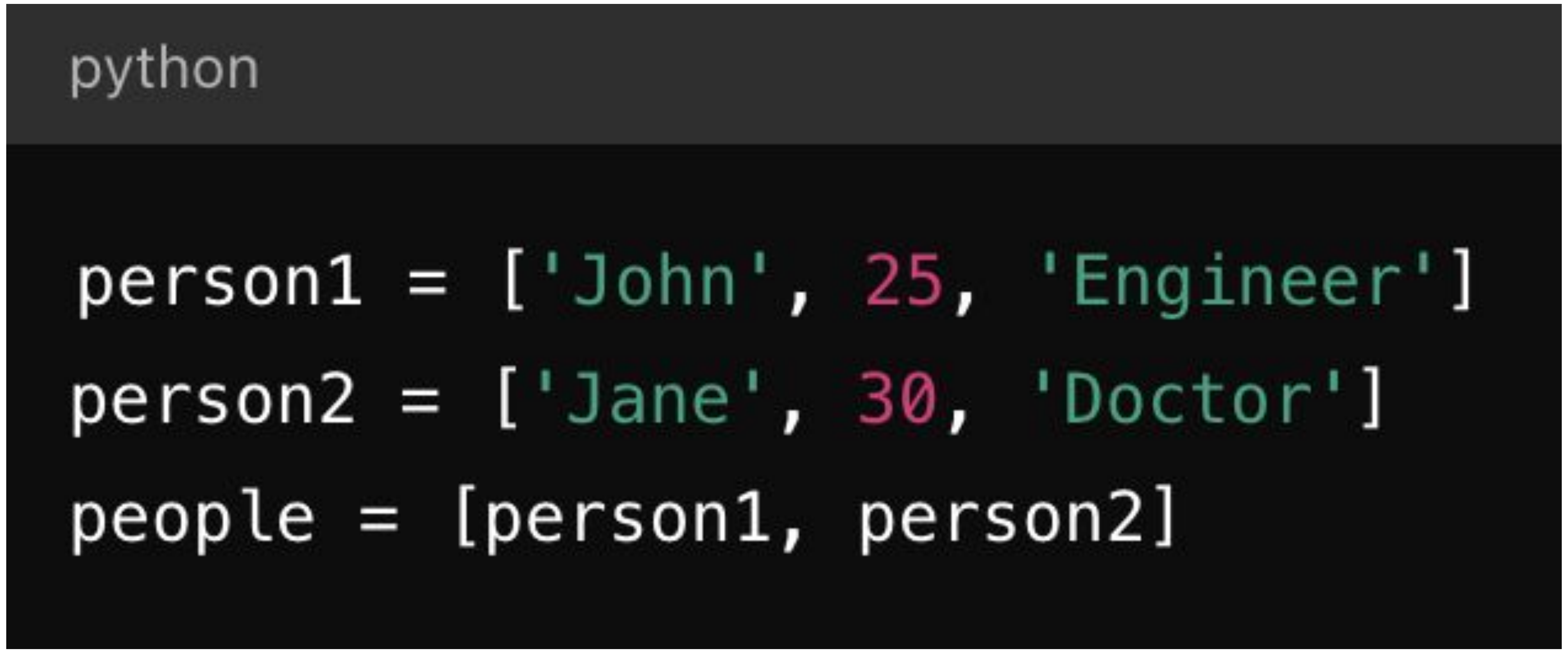


python

```
matrix = [[1, 2, 3],
          [4, 5, 6],
          [7, 8, 9]]
```

Use Cases : Hierarchical Data

They are useful for representing **hierarchical data** structures where each level of the hierarchy can contain **multiple elements** or substructures.

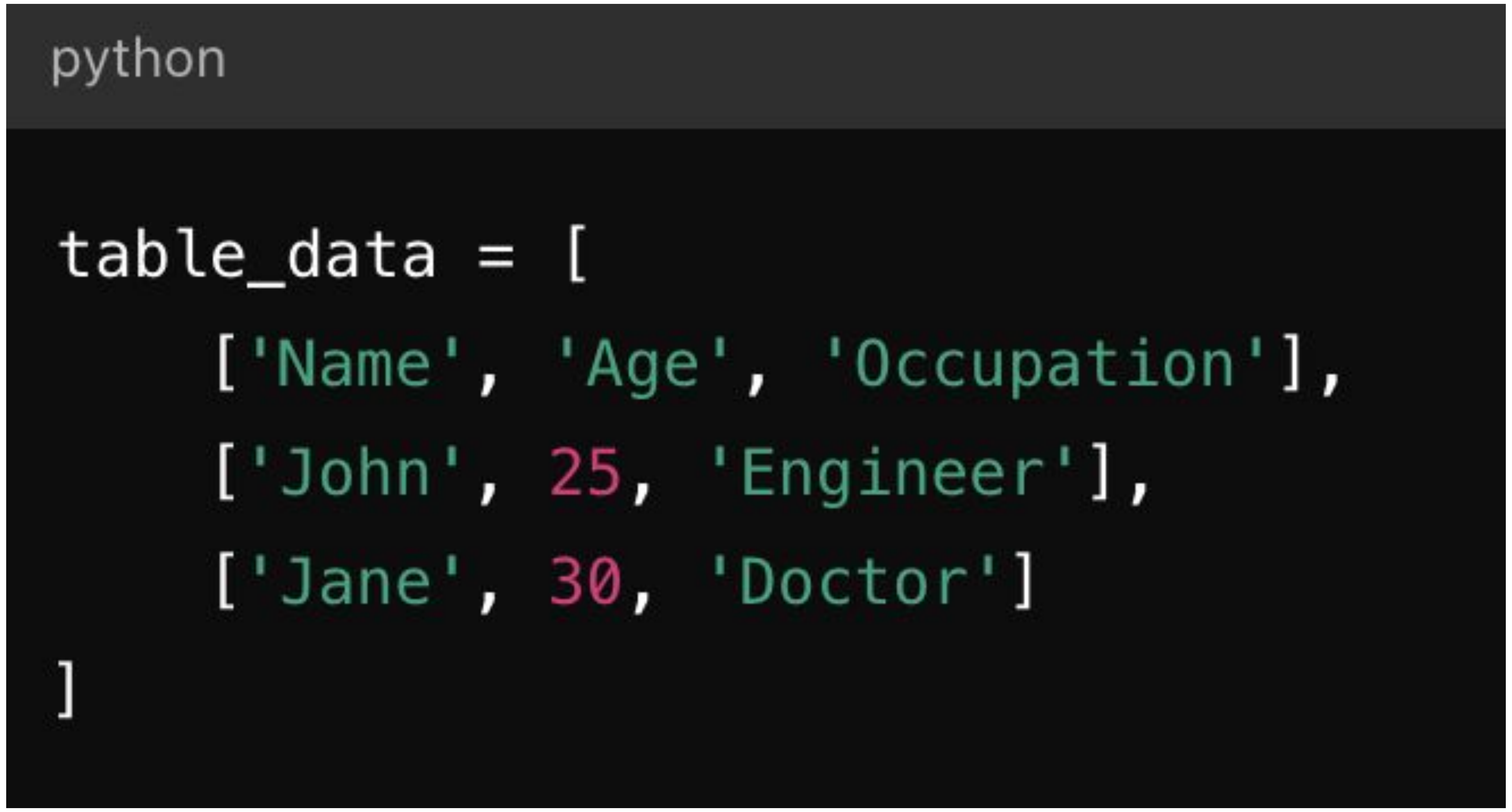


python

```python
person1 = ['John', 25, 'Engineer']
person2 = ['Jane', 30, 'Doctor']
people = [person1, person2]
```

Use Cases : Table Like Data

When working with **tabular data** that has **rows and columns**, nested lists can be a natural choice.



python

```python
table_data = [
    ['Name', 'Age', 'Occupation'],
    ['John', 25, 'Engineer'],
    ['Jane', 30, 'Doctor']
]
```

Newton School of Technology

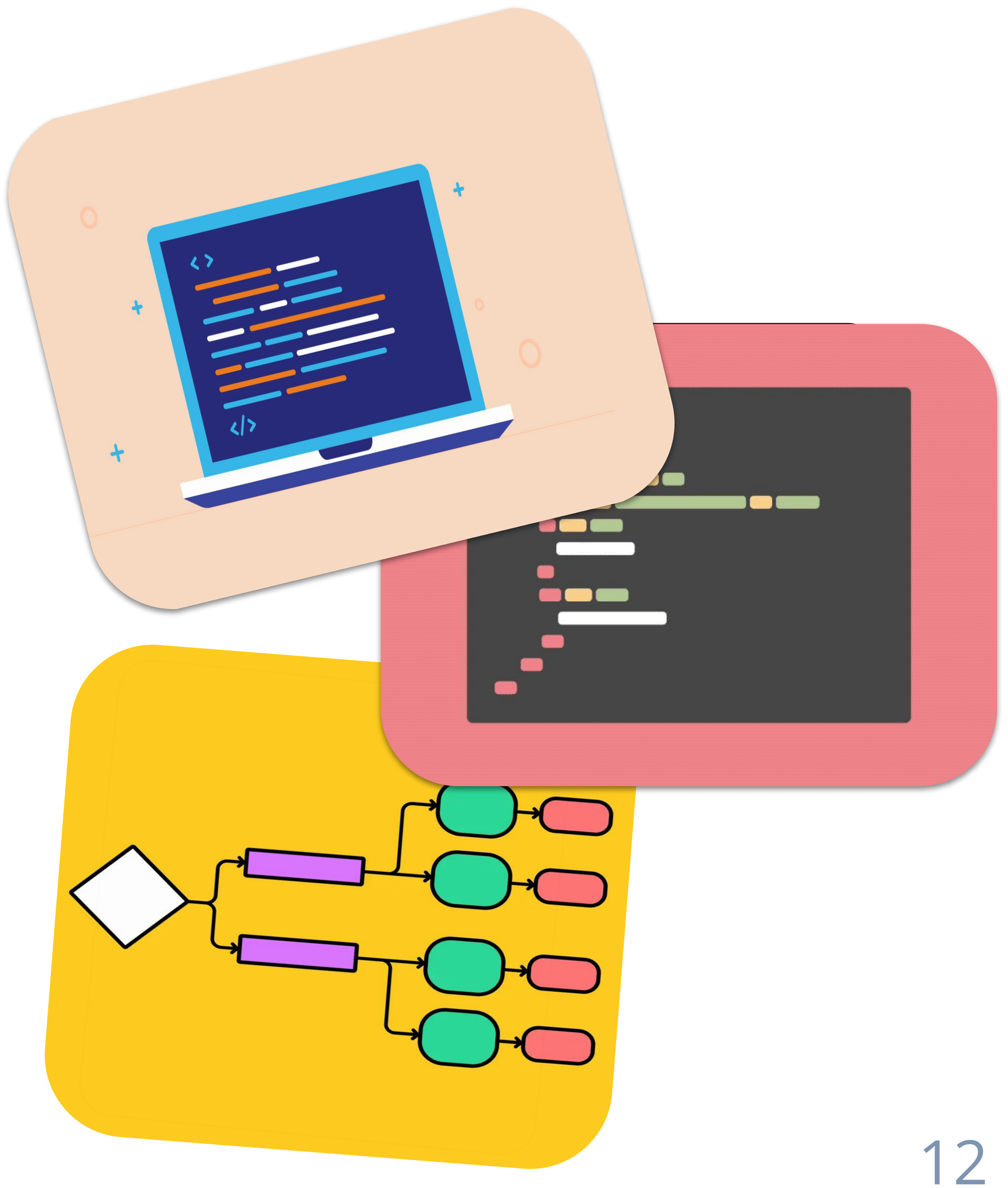
Lets Practice on the Playground!

Newton School of Technology

Question 2 - Book Catalog

Summary

- **List Comprehension:** A concise way to create lists using a single line with a loop and optional condition.
- **Nested Lists:** Lists within lists, often used for multi-dimensional data



Thank You!