Programming Concepts
L-03

By Vladyslav **BARDIN**

01: Loops

➢ **Iteration:**
The process of going through each item in a collection, one at a time.

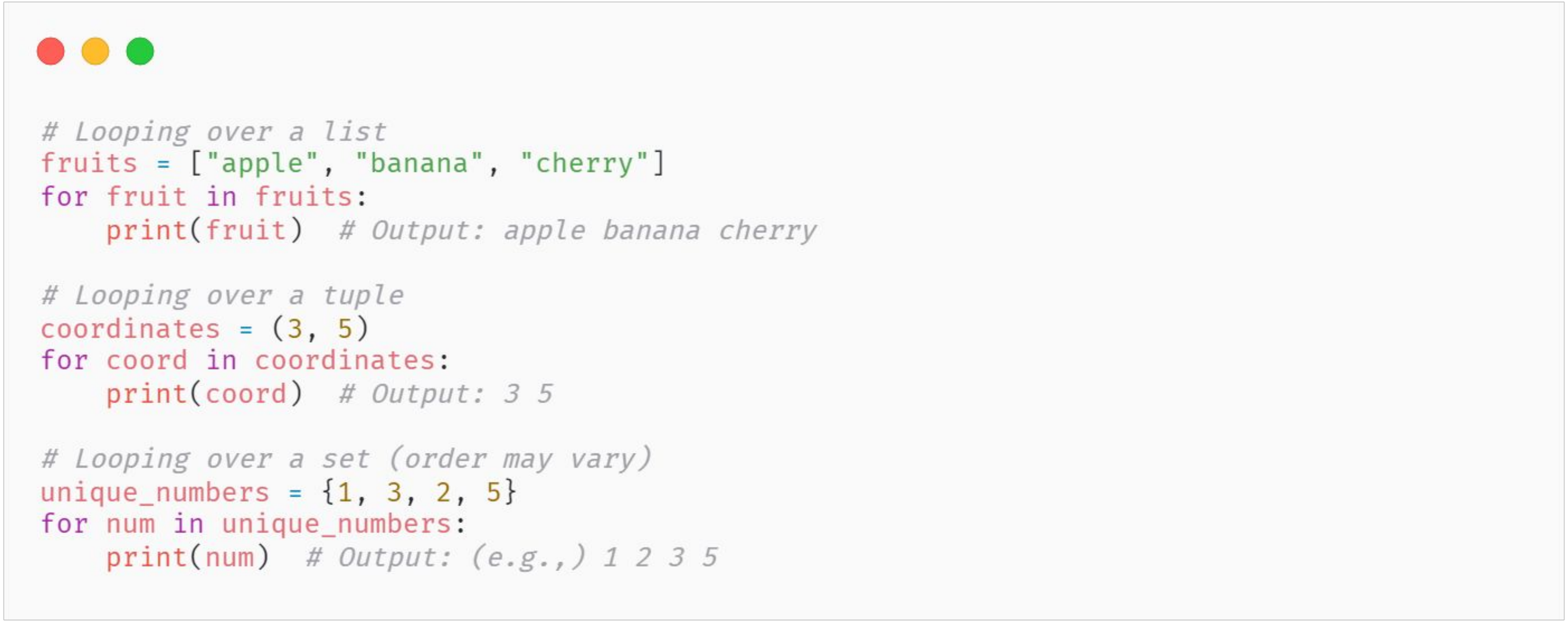
➢ **Why Loops Matter:**
Loops allow you to automate repetitive tasks, perform calculations on each element, and dynamically interact with your data.

➢ **Types of Loops in Python:**

✓ for loops: Ideal for iterating over sequences (lists, tuples, strings, etc.)

✓ while loops: Used when the number of iterations is unknown beforehand.

02: Loops

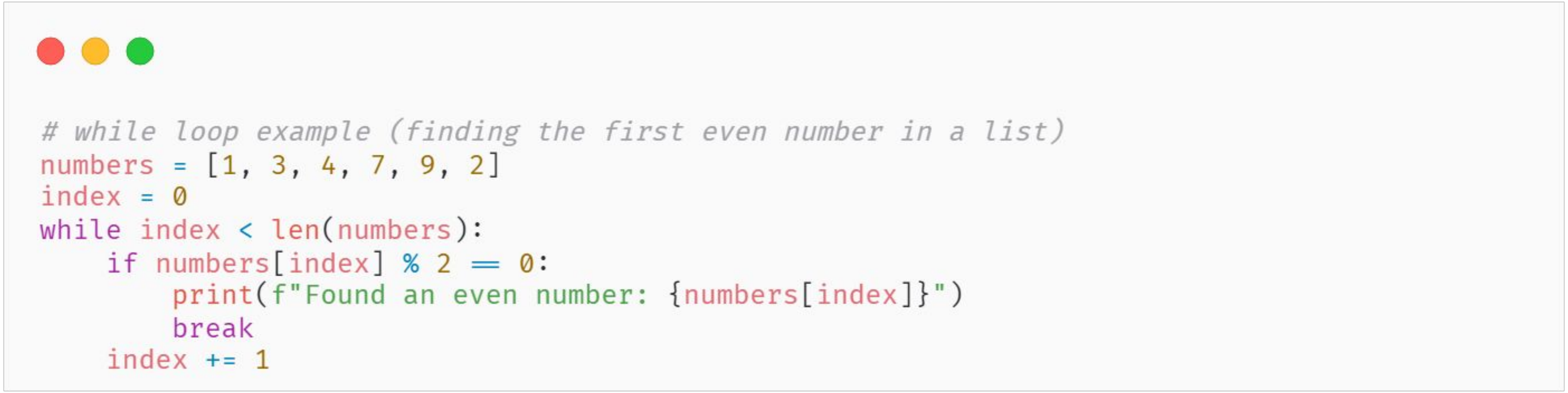


```python
# Looping over a list
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)  # Output: apple banana cherry

# Looping over a tuple
coordinates = (3, 5)
for coord in coordinates:
    print(coord)  # Output: 3 5

# Looping over a set (order may vary)
unique_numbers = {1, 3, 2, 5}
for num in unique_numbers:
    print(num)  # Output: (e.g.,) 1 2 3 5
```

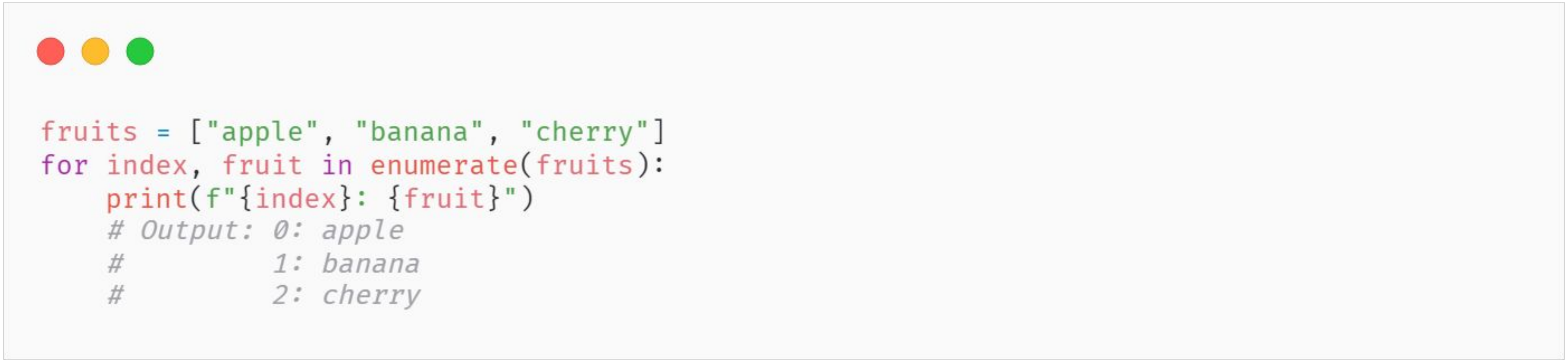
03: Loops



```python
# while loop example (finding the first even number in a list)
numbers = [1, 3, 4, 7, 9, 2]
index = 0
while index < len(numbers):
    if numbers[index] % 2 == 0:
        print(f"Found an even number: {numbers[index]}")
        break
    index += 1
```

04: Loops

- Use **enumerate()** when you need both the index and the value of each item.



```python
fruits = ["apple", "banana", "cherry"]
for index, fruit in enumerate(fruits):
    print(f"{index}: {fruit}")
    # Output: 0: apple
    #         1: banana
    #         2: cherry
```

05: What are Collections?

- ➢ **Containers for Data:**
  Think of them as specialized boxes to hold and organize multiple pieces of information.

- ➢ **More Than Just Variables:**
  A single collection can store many values, simplifying data management.

- ➢ **Building Blocks for Programs:**
  Collections are essential tools for creating complex and useful applications.

06: Why are Collections Important?

- ➢ **Organize and Structure:**
  Make your data easy to access and manipulate.

- ➢ **Solve Real Problems:**
  From managing inventories to analyzing text, collections are at the core of many programming tasks.

- ➢ **Flexibility:**
  Adapt your code to handle varying amounts of data.

- ➢ **Efficiency:**
  Collections offer optimized ways to work with data, improving performance.

07: Types We'll Cover Today

➢ **Lists:**
Ordered, mutable sequences – our workhorse for general-purpose collections.
Your grocery shopping list, where order matters and you can add/remove items.

➢ **Tuples:**
Ordered, immutable sequences – ideal for fixed data that shouldn't change.
A person's name and birthdate, which should stay fixed.

➢ **Sets:**
Unordered, unique collections – perfect for eliminating duplicates and performing set operations.
A collection of unique ingredients in a recipe.

08: Lists

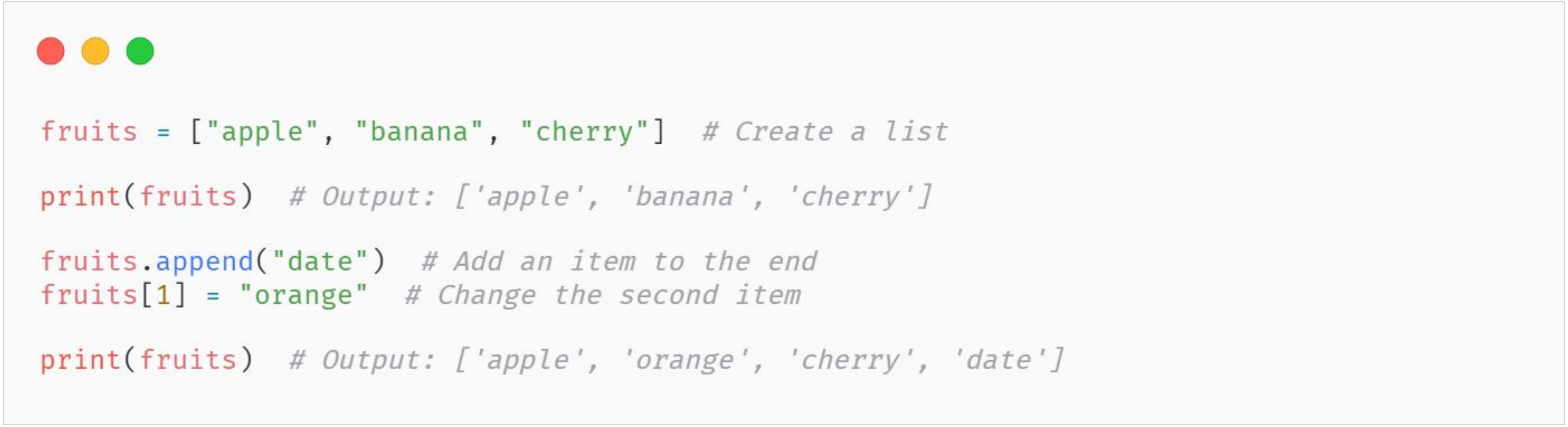
- **Lists:**
  Ordered, mutable sequences – our workhorse for general-purpose collections.
  Your grocery shopping list, where order matters and you can add/remove items.

- **Ordered:**
  Items have a specific position, like numbers in a line.

- **Mutable:**
  You can change, add, or remove items after the list is created.

- **Sequence:**
  A type of collection where items can be accessed by their index (position).

09: Lists



```python
fruits = ["apple", "banana", "cherry"]  # Create a list

print(fruits)  # Output: ['apple', 'banana', 'cherry']

fruits.append("date")  # Add an item to the end
fruits[1] = "orange"  # Change the second item

print(fruits)  # Output: ['apple', 'orange', 'cherry', 'date']
```

10: Lists

Method	Description	Example
append()	Adds an item to the end of the list	fruits.append("grape")
insert()	Inserts an item at a specific index	fruits.insert(2, "mango")
pop()	Removes and returns the last item	last_fruit = fruits.pop()
remove()	Removes the first occurrence of a value	fruits.remove("orange")

Slicing	Extracts a portion of the list	first_two = fruits[:2]
		all_but_first = fruits[1:]

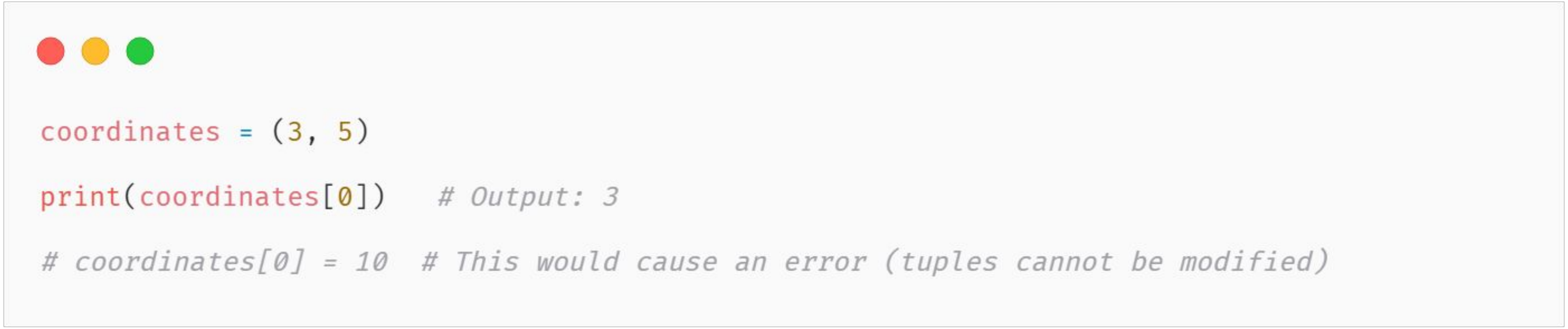
11: Lists

- Lists are incredibly flexible – use them to store collections of any data type.
- Their mutability makes them great for dynamic data.
- Numerous built-in methods make list manipulation easy.

12: Tuples

- **Tuples:**
  Ordered, immutable sequences – ideal for fixed data that shouldn't change.
  A person's name and birthdate, which should stay fixed.
- **Ordered:**
  Like lists, tuples maintain the order of their items.
- **Immutable:**
  Once created, you cannot change the values within a tuple.
- **Sequence:**
  Also accessed by index, just like lists.

13: Tuples



```python
coordinates = (3, 5)

print(coordinates[0])   # Output: 3

# coordinates[0] = 10  # This would cause an error (tuples cannot be modified)
```

14: Tuples

- ➢ **Data Integrity:**
  Tuples ensure your data remains consistent and unaltered, preventing accidental modifications.

- ➢ **Efficiency:**
  Under the hood, Python can optimize tuples for faster performance compared to lists.

- ➢ **Key Use Cases:**
  - ✓ Representing fixed data (e.g., coordinates, RGB colors)
  - ✓ Returning multiple values from functions
  - ✓ Dictionaries use tuples as keys ***(more on this later!)***

15: Tuples

Method	Description	Example
count()	Returns the number of times a value appears	my_tuple.count(5)
index()	Returns the index of the first occurrence of a value	my_tuple.index('a')

16: Tuples

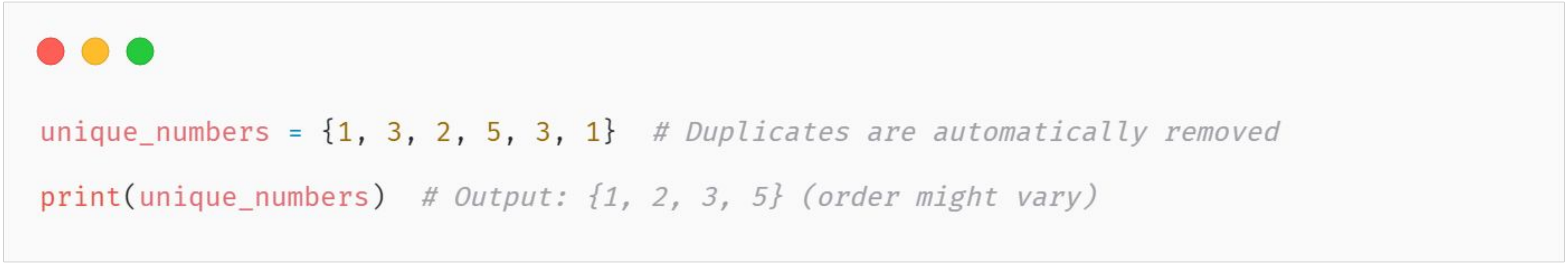
- ➢ Tuples are like engraved stone tablets – once the message is etched, it cannot be changed.
- ➢ This makes them perfect for data that needs to be reliable and secure.

- ➢ Tuples are denoted by parentheses ().
- ➢ You can create an empty tuple: empty_tuple = ()
- ➢ You can access tuple elements by index: my_tuple[2]

17: Sets

- **Sets:**
  Unordered, unique collections – perfect for eliminating duplicates and performing set operations.
  A collection of unique ingredients in a recipe.
- **Unordered:**
  Items have no specific position; you cannot access them by index.
- **Unique:**
  Each value can appear only once in a set.
- **Mutable:**
  You can add or remove items, but not modify existing ones.

18: Sets



```
unique_numbers = {1, 3, 2, 5, 3, 1}  # Duplicates are automatically removed
print(unique_numbers)  # Output: {1, 2, 3, 5} (order might vary)
```

19: Sets

- ➢ **Eliminate Duplicates:**
  Automatically ensures unique values, great for data cleaning.

- ➢ **Membership Testing:**
  Quickly check if a value exists in the set.

- ➢ **Mathematical Set Operations:**
  Perform union, intersection, difference, etc.

20: Sets

Method	Description	Example
add()	Adds an element to the set	unique_numbers.add(8)
remove()	Removes a specific element (raises an error if not found)	unique_numbers.remove(3)
discard()	Removes a specific element (no error if not found)	unique_numbers.discard(10)

21: Sets

Method	Description	Example
union()	Combines elements from two or more sets	set1.union(set2)
intersection()	Returns a new set with elements common to both sets	set1.intersection(set2)
difference()	Returns a new set with elements in the first set but not the second	set1.difference(set2)

22: Sets

- Sets are like a bag of marbles – each marble is unique, and there's no inherent order to them.
- You can quickly reach into the bag to check if a specific marble is there.

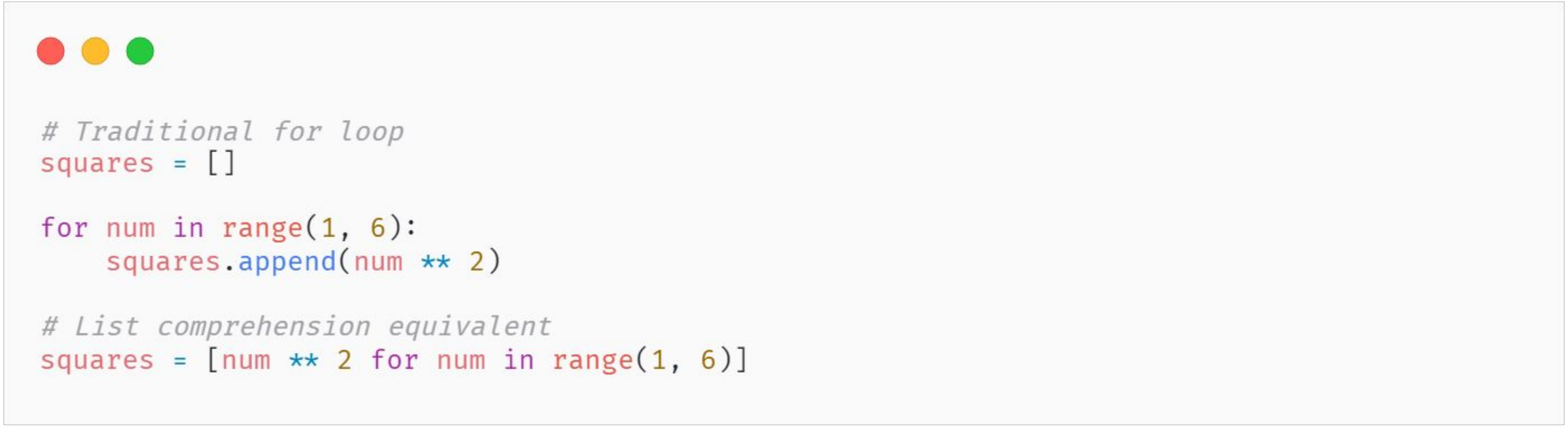
23: Real-World Examples

- **Lists:**
  Storing customer orders, a deck of cards in a game, a playlist of songs.
- **Tuples:**
  Representing geographic coordinates, dates (year, month, day), database records.
- **Sets:** Finding unique words in a text, eliminating duplicate entries in a database, managing user permissions.

24: List Comprehensions

- **Concise Syntax:**
  Create new lists in a single line of code.
- **Readability:**
  Express complex list transformations in a clear way.
- **Performance:**
  Often faster than traditional loops.

25: List Comprehensions



```
# Traditional for loop
squares = []

for num in range(1, 6):
    squares.append(num ** 2)

# List comprehension equivalent
squares = [num ** 2 for num in range(1, 6)]
```

26: Additional Resources

- Python Loops Tutorial: for Loops, while Loops and Nested Loops by Programming with Mosh: https://www.codingem.com/nested-loops-in-python/
- Python Tutorial for Beginners 5: Lists, Tuples, and Sets by Corey Schafer: https://www.youtube.com/watch?v=W8KRzm-HUcc
- Lists and Tuples in Python on Real Python: https://realpython.com/python-lists-tuples/
- Python Sets on W3Schools: https://www.w3schools.com/python/python_sets.asp