Programming Concepts
L-04

By Vladyslav **BARDIN**

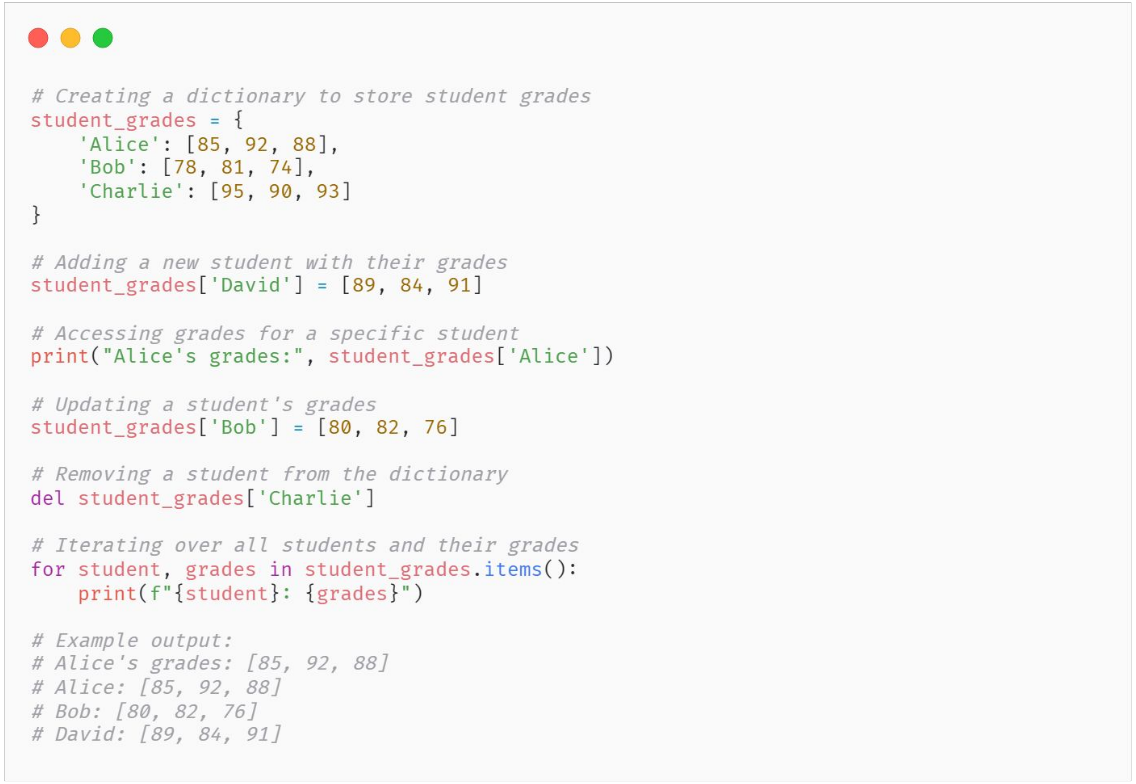
01: Types We'll Cover Today

- **Dictionary:**
  Unordered, mutable collections of key-value pairs – our go-to for fast lookups and associations.
  Think of a contact list, where each contact name (key) is associated with a phone number (value) for quick access and updates.

02: Dictionary

- ➢ **Dictionary:**
  Unordered, mutable collections of key-value pairs – our go-to for fast lookups and associations.
- ➢ **Unordered:**
  Items do not have a specific position..
- ➢ **Mutable:**
  You can change, add, or remove key-value pairs after the dictionary is created.
- ➢ **Key-Value Pairs:**
  Each key is unique and maps to a value, enabling efficient data retrieval based on keys.

03: Dictionary

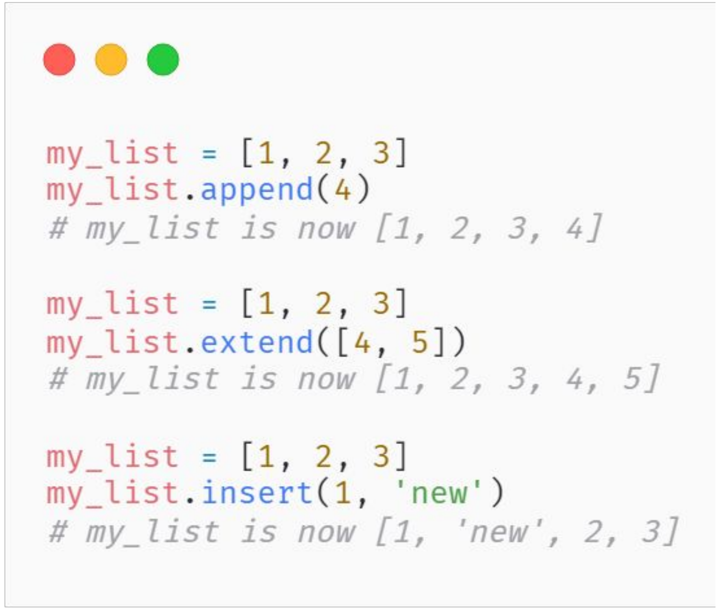


```python
# Creating a dictionary to store student grades
student_grades = {
    'Alice': [85, 92, 88],
    'Bob': [78, 81, 74],
    'Charlie': [95, 90, 93]
}

# Adding a new student with their grades
student_grades['David'] = [89, 84, 91]

# Accessing grades for a specific student
print("Alice's grades:", student_grades['Alice'])

# Updating a student's grades
student_grades['Bob'] = [80, 82, 76]

# Removing a student from the dictionary
del student_grades['Charlie']

# Iterating over all students and their grades
for student, grades in student_grades.items():
    print(f"{student}: {grades}")

# Example output:
# Alice's grades: [85, 92, 88]
# Alice: [85, 92, 88]
# Bob: [80, 82, 76]
# David: [89, 84, 91]
```

04: Dictionary

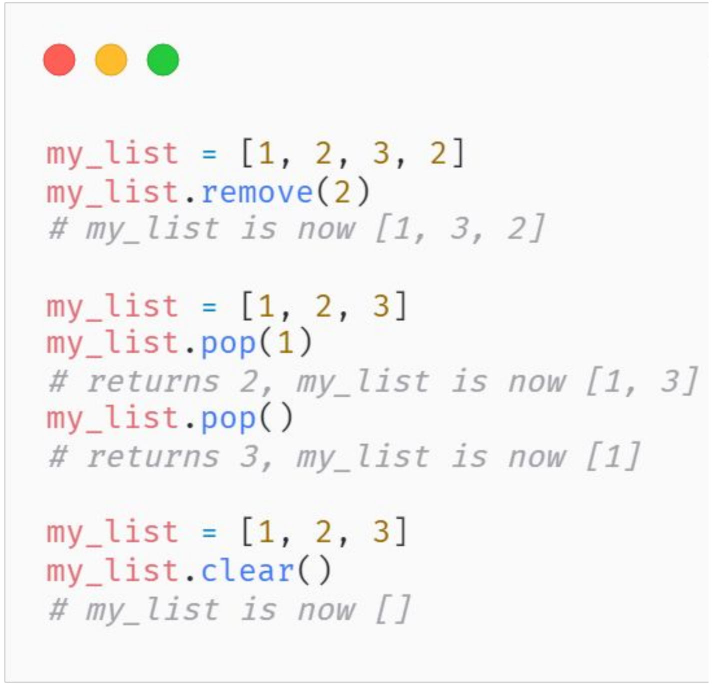
Method	Description	Example
dict.clear()	Removes all items from the dictionary.	student_grades.clear()
dict.copy()	Returns a shallow copy of the dictionary.	grades_copy = student_grades.copy()
dict.fromkeys()	Creates a new dictionary with keys from a sequence and values set to a specified value.	new_dict = dict.fromkeys(['Alice', 'Bob'], 0)
dict.get()	Returns the value for a specified key if the key is in the dictionary.	student_grades.get('Alice', 'No record')
dict.items()	Returns a view object that displays a list of dictionary's key-value tuple pairs.	for item in student_grades.items(): print(item)
dict.keys()	Returns a view object that displays a list of all the keys.	keys = student_grades.keys()
dict.pop()	Removes the specified key and returns the corresponding value.	grades = student_grades.pop('Alice')
dict.popitem()	Removes and returns the last inserted key-value pair.	last_item = student_grades.popitem()
dict.setdefault()	Returns the value of a key if it is in the dictionary. If not, inserts the key with a specified value.	grade = student_grades.setdefault('Eve', [80, 85, 90])
dict.update()	Updates the dictionary with elements from another dictionary object or from an iterable of key-value pairs.	student_grades.update({'Eve': [80, 85, 90]})
dict.values()	Returns a view object that displays a list of all the values.	values = student_grades.values()

05: Collections methods



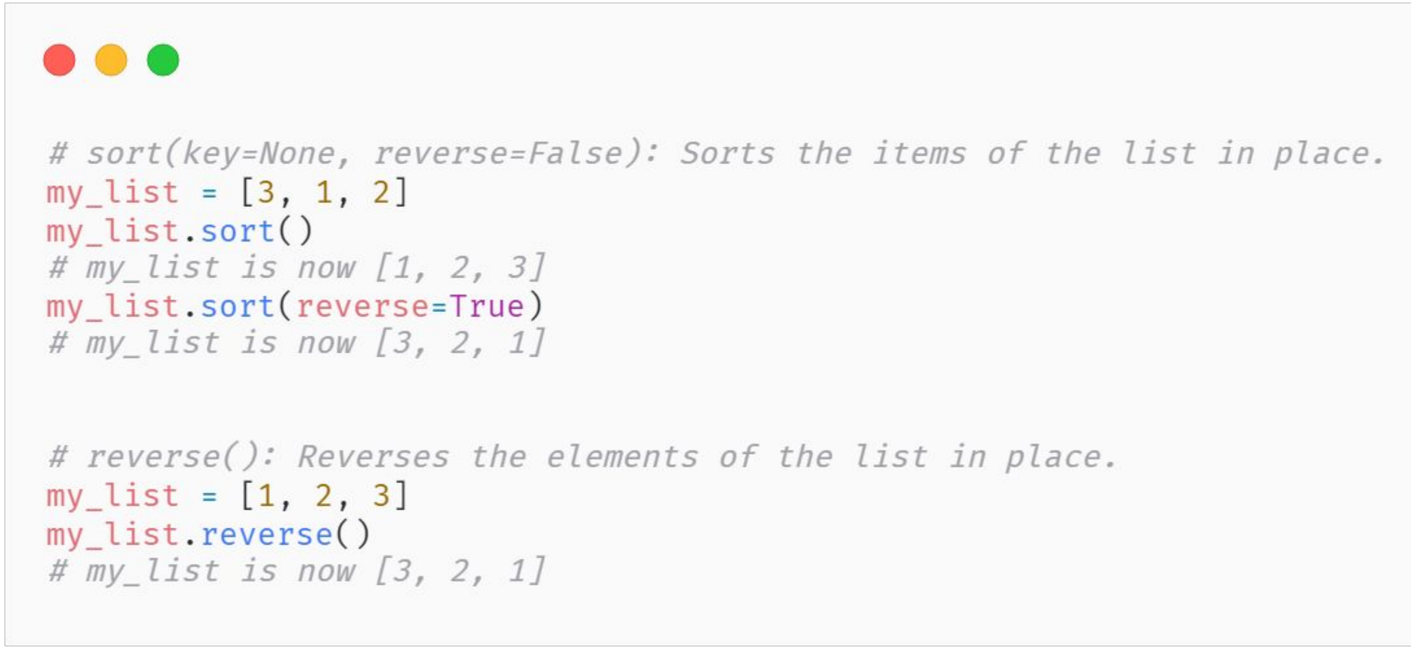
```
my_list = [1, 2, 3]
my_list.append(4)
# my_list is now [1, 2, 3, 4]

my_list = [1, 2, 3]
my_list.extend([4, 5])
# my_list is now [1, 2, 3, 4, 5]

my_list = [1, 2, 3]
my_list.insert(1, 'new')
# my_list is now [1, 'new', 2, 3]
```

06: Collections methods



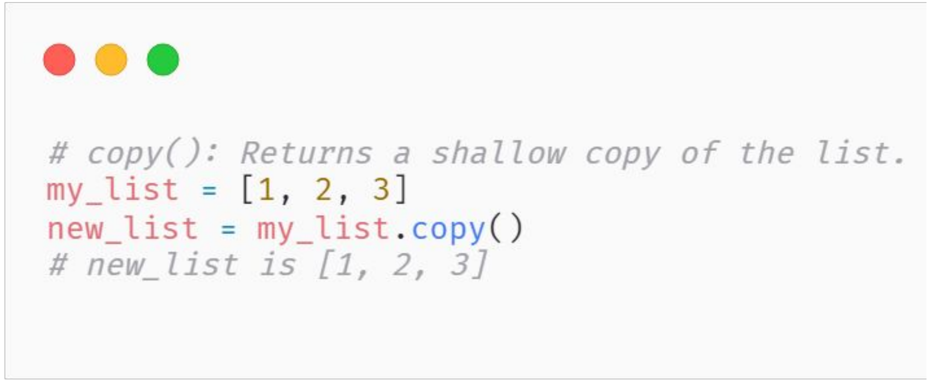
```python
my_list = [1, 2, 3, 2]
my_list.remove(2)
# my_list is now [1, 3, 2]

my_list = [1, 2, 3]
my_list.pop(1)
# returns 2, my_list is now [1, 3]
my_list.pop()
# returns 3, my_list is now [1]

my_list = [1, 2, 3]
my_list.clear()
# my_list is now []
```

07: Collections methods



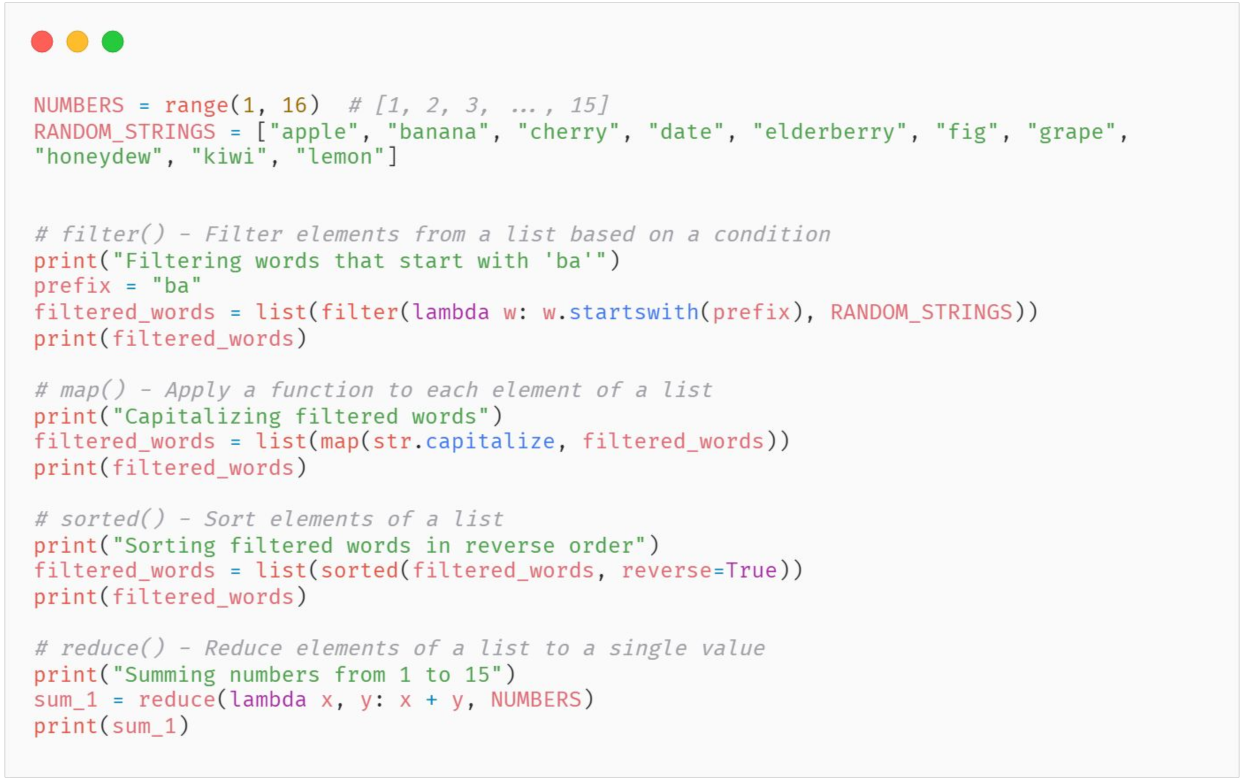
```python
# sort(key=None, reverse=False): Sorts the items of the list in place.
my_list = [3, 1, 2]
my_list.sort()
# my_list is now [1, 2, 3]
my_list.sort(reverse=True)
# my_list is now [3, 2, 1]


# reverse(): Reverses the elements of the list in place.
my_list = [1, 2, 3]
my_list.reverse()
# my_list is now [3, 2, 1]
```

08: Collections methods



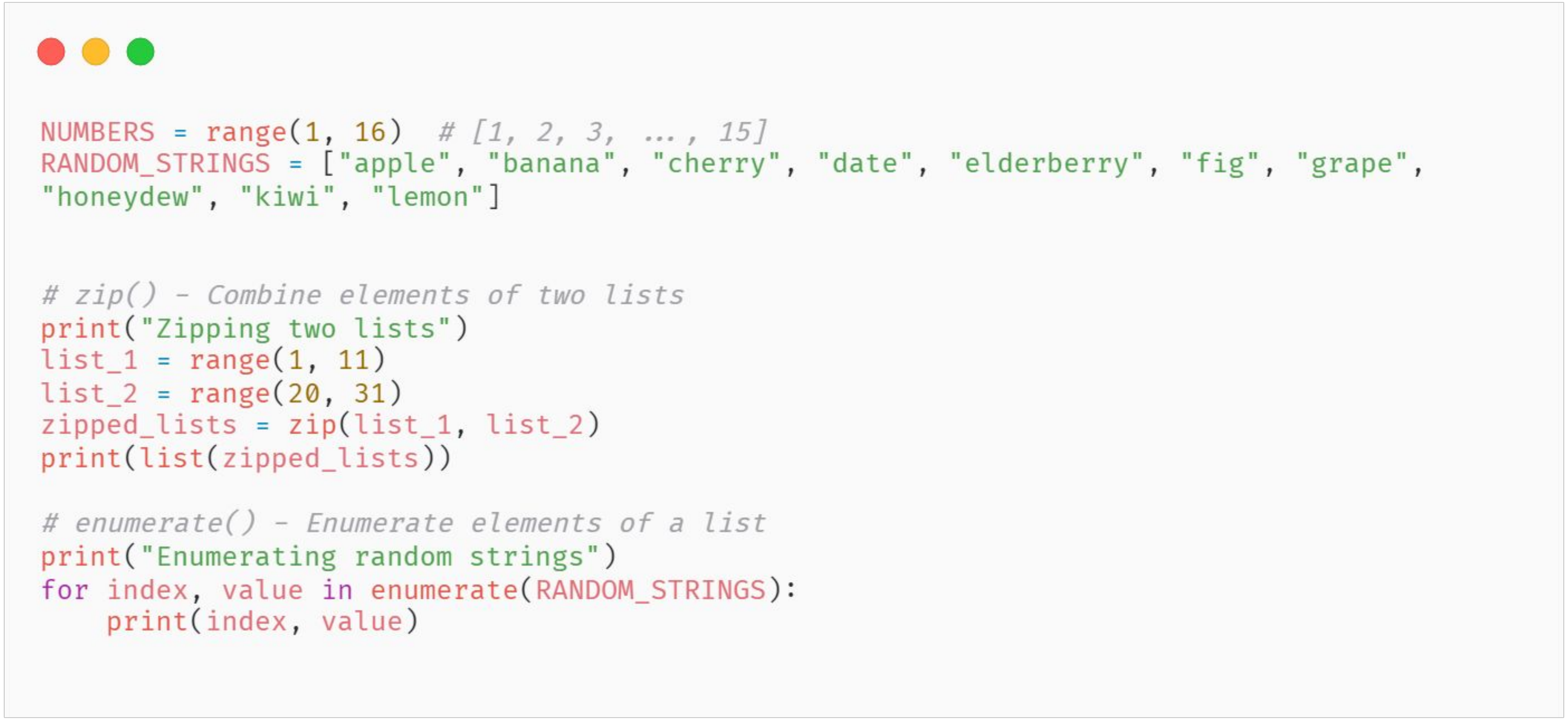
```
# copy(): Returns a shallow copy of the list.
my_list = [1, 2, 3]
new_list = my_list.copy()
# new_list is [1, 2, 3]
```

09: Collections methods



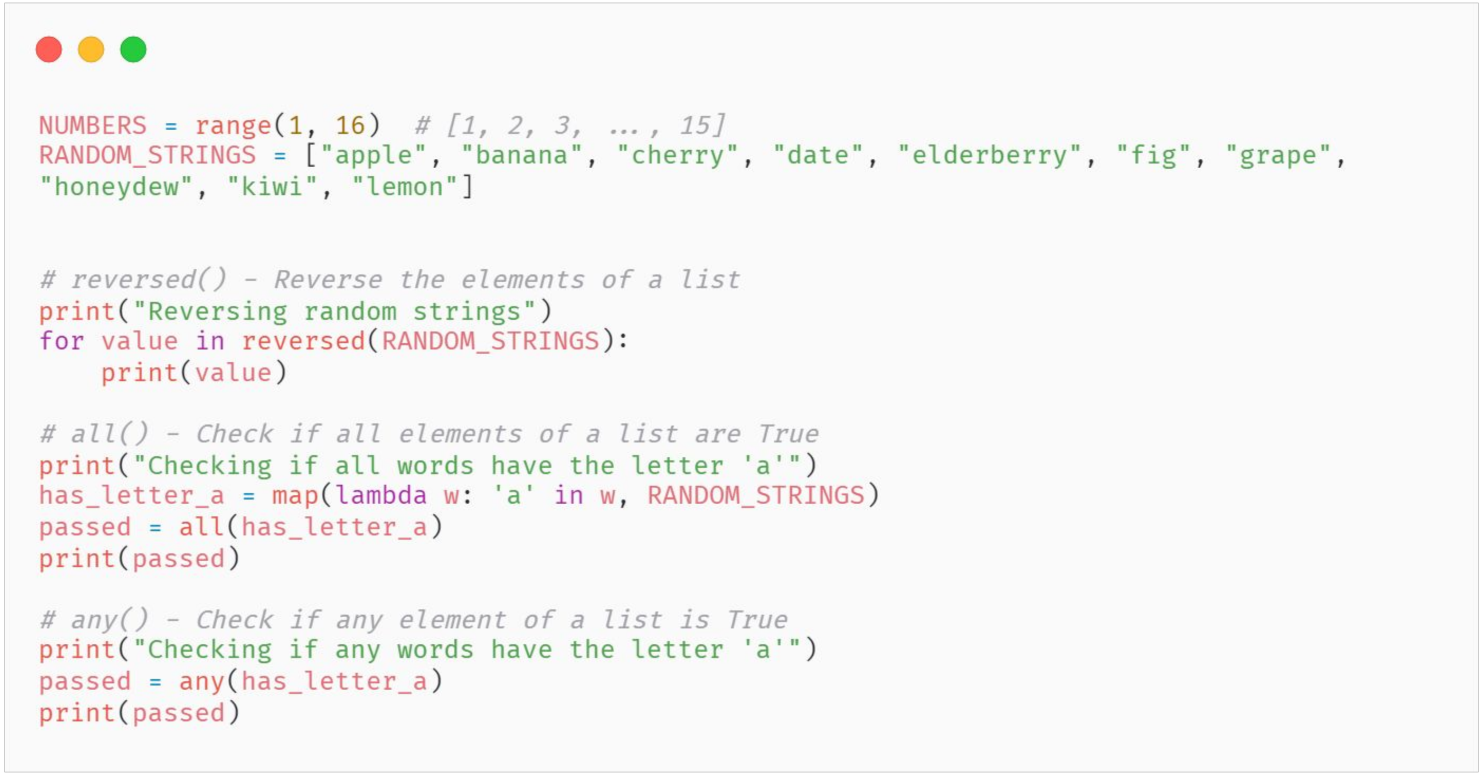
```python
NUMBERS = range(1, 16)  # [1, 2, 3, ..., 15]
RANDOM_STRINGS = ["apple", "banana", "cherry", "date", "elderberry", "fig", "grape",
"honeydew", "kiwi", "lemon"]


# filter() - Filter elements from a list based on a condition
print("Filtering words that start with 'ba'")
prefix = "ba"
filtered_words = list(filter(lambda w: w.startswith(prefix), RANDOM_STRINGS))
print(filtered_words)

# map() - Apply a function to each element of a list
print("Capitalizing filtered words")
filtered_words = list(map(str.capitalize, filtered_words))
print(filtered_words)

# sorted() - Sort elements of a list
print("Sorting filtered words in reverse order")
filtered_words = list(sorted(filtered_words, reverse=True))
print(filtered_words)

# reduce() - Reduce elements of a list to a single value
print("Summing numbers from 1 to 15")
sum_1 = reduce(lambda x, y: x + y, NUMBERS)
print(sum_1)
```

10: Collections methods



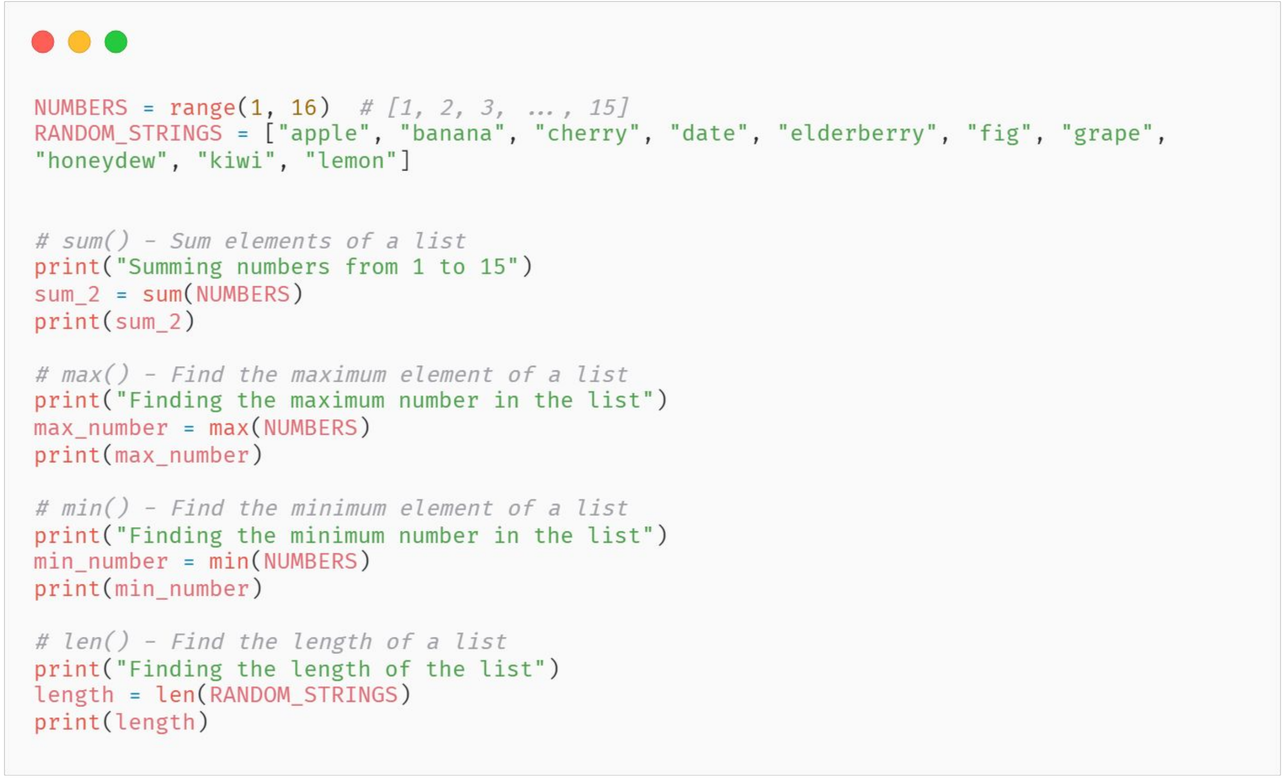
```python
NUMBERS = range(1, 16)  # [1, 2, 3, ..., 15]
RANDOM_STRINGS = ["apple", "banana", "cherry", "date", "elderberry", "fig", "grape",
"honeydew", "kiwi", "lemon"]


# zip() - Combine elements of two lists
print("Zipping two lists")
list_1 = range(1, 11)
list_2 = range(20, 31)
zipped_lists = zip(list_1, list_2)
print(list(zipped_lists))

# enumerate() - Enumerate elements of a list
print("Enumerating random strings")
for index, value in enumerate(RANDOM_STRINGS):
    print(index, value)
```

11: Collections methods



```python
NUMBERS = range(1, 16)  # [1, 2, 3, ..., 15]
RANDOM_STRINGS = ["apple", "banana", "cherry", "date", "elderberry", "fig", "grape",
"honeydew", "kiwi", "lemon"]


# reversed() - Reverse the elements of a list
print("Reversing random strings")
for value in reversed(RANDOM_STRINGS):
    print(value)

# all() - Check if all elements of a list are True
print("Checking if all words have the letter 'a'")
has_letter_a = map(lambda w: 'a' in w, RANDOM_STRINGS)
passed = all(has_letter_a)
print(passed)

# any() - Check if any element of a list is True
print("Checking if any words have the letter 'a'")
passed = any(has_letter_a)
print(passed)
```

12: Collections methods



```python
NUMBERS = range(1, 16)  # [1, 2, 3, ... , 15]
RANDOM_STRINGS = ["apple", "banana", "cherry", "date", "elderberry", "fig", "grape",
"honeydew", "kiwi", "lemon"]


# sum() - Sum elements of a list
print("Summing numbers from 1 to 15")
sum_2 = sum(NUMBERS)
print(sum_2)

# max() - Find the maximum element of a list
print("Finding the maximum number in the list")
max_number = max(NUMBERS)
print(max_number)

# min() - Find the minimum element of a list
print("Finding the minimum number in the list")
min_number = min(NUMBERS)
print(min_number)

# len() - Find the length of a list
print("Finding the length of the list")
length = len(RANDOM_STRINGS)
print(length)
```

13: Additional Resources

- https://www.w3schools.com/python/python_dictionaries.asp
- https://www.geeksforgeeks.org/python-dictionary/