Python Track

Basic data structures
(Lists and Tuples)

Africa To Silicon Valley

Lecture Flow

- Lists
- Tuples

Lists

Africa To Silicon Valley

What are lists?

- Lists are fundamental data structures in Python used to store collections of data.
- They can hold items of any data type, including numbers, strings, and even other lists.
- Lists are ordered, changeable, and allow duplicate values.

Creating lists

- Lists can be created using square **brackets []** and separating items with commas.
- The **list()** constructor can also be used to create lists.

```
# Creating a list using square brackets
fruits = ["apple", "banana", "cherry"]

# Convert iterables to list using the list() constructor
numbers = list((1, 2, 3, 4, 5))
```

List data types

List items can be of any data type

```
• list1 = ["apple", "banana", "cherry"]
• list2 = [1, 5, 7, 9, 3]
• list3 = [True, False, False]
• list4 = ["abc", 34, True, 40, "male"]
```

Accessing List Items

- List items are accessed using their index number, starting from 0.
- Negative indexing can be used to access items from the end of the list.

```
nums = [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

# Accessing the first item

nums[0]              # 10
# Accessing the last item

nums[-1]         # 19
```

Slicing Lists

- Slicing allows extracting a sublist from a list.
- Slicing uses the **colon (:)** to separate start and end indices (inclusive).

```
nums = [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
# Extracting a sublist from index 2 to index 4

nums[2:5]          # [12, 13, 14]

nums[-4 : -1]  ??
```

Africa To Silicon Valley

Modifying Lists

- Lists are mutable, allowing you to change their contents.
- You can modify items using their index or extend the list using **append**() and **insert**().
- You can also remove items using **remove**() and **pop**().

Examples

```
fruits = ["apple", "banana", "cherry"]
# Changing the first item

fruits[0] = "orange"  # fruits = ["orange", "banana", "cherry"]

# Adding an item to the end

fruits.append("mango")  # fruits = ["orange", "banana", "cherry", "mango"]

# Removing an item by value

fruits.remove("cherry")  # fruits = ["orange", "banana", "mango"]

# Removing the last item

removed_item = fruits.pop()  # removed_item = "mango", fruits =
["orange", "banana"]
```

A?SV Africa To Silicon Valley

Common List Operations

- Checking if an item exists: in keyword
- Sorting a list: **sort**() method
- **sorted** (nums , key = myFunction ( ), reverse = True/False)
- Reversing a list: **reverse**() method

Examples

```
fruits = ["orange", "banana"]
# Checking if "apple" exists in the list
if "apple" in fruits:
    print("Yes, apple is in the list")
# Sorting the list in ascending order
fruits.sort()  # fruits = ["banana", "orange"]
# Reversing the sorted list
fruits.reverse()  # fruits = ["orange", "banana"]
```

A2SV
Africa To Silicon Valley

Combining Lists

- Concatenating lists using the + operator or extend() method
- Adding items from one list to another individually

Examples

```
numbers = [1, 2, 3]

fruits = ["orange", "banana"]
# Concatenating lists using '+' operator

new_list = fruits + numbers  # new_list = ["orange", "banana", 1, 2, 3]

# Extending a list using extend() method

fruits.extend(numbers)  # fruits = ["orange", "banana", 1, 2, 3]
```

Traversing Lists

- Iterating through lists using for loops
- Accessing both index and value using enumerate() function

-
  ```
  for index in range(len(nums)):
          print(nums[index])
  ```

-
  ```
  for num in nums:
          print(num)
  ```

-
  ```
  for index, num in enumerate(nums):
          print(index, num)
  ```

List Comprehension

- Creating new lists based on existing lists
- Using expressions and conditions to filter and transform list elements

```
# Creating a list of even numbers from a list of numbers
numbers = [1, 2, 3, 4, 5]
even_numbers = [num for num in numbers if num % 2 == 0]
# even_numbers = [2, 4]
```

A2SV
Africa To Silicon Valley

Why List Comprehension?

```
my_list = [[0]] * 5
my_list = ?  #[[0], [0], [0], [0], [0]]
my_list[0][0] = 1
my_list = ?
```

Africa To Silicon Valley

Why List Comprehension?

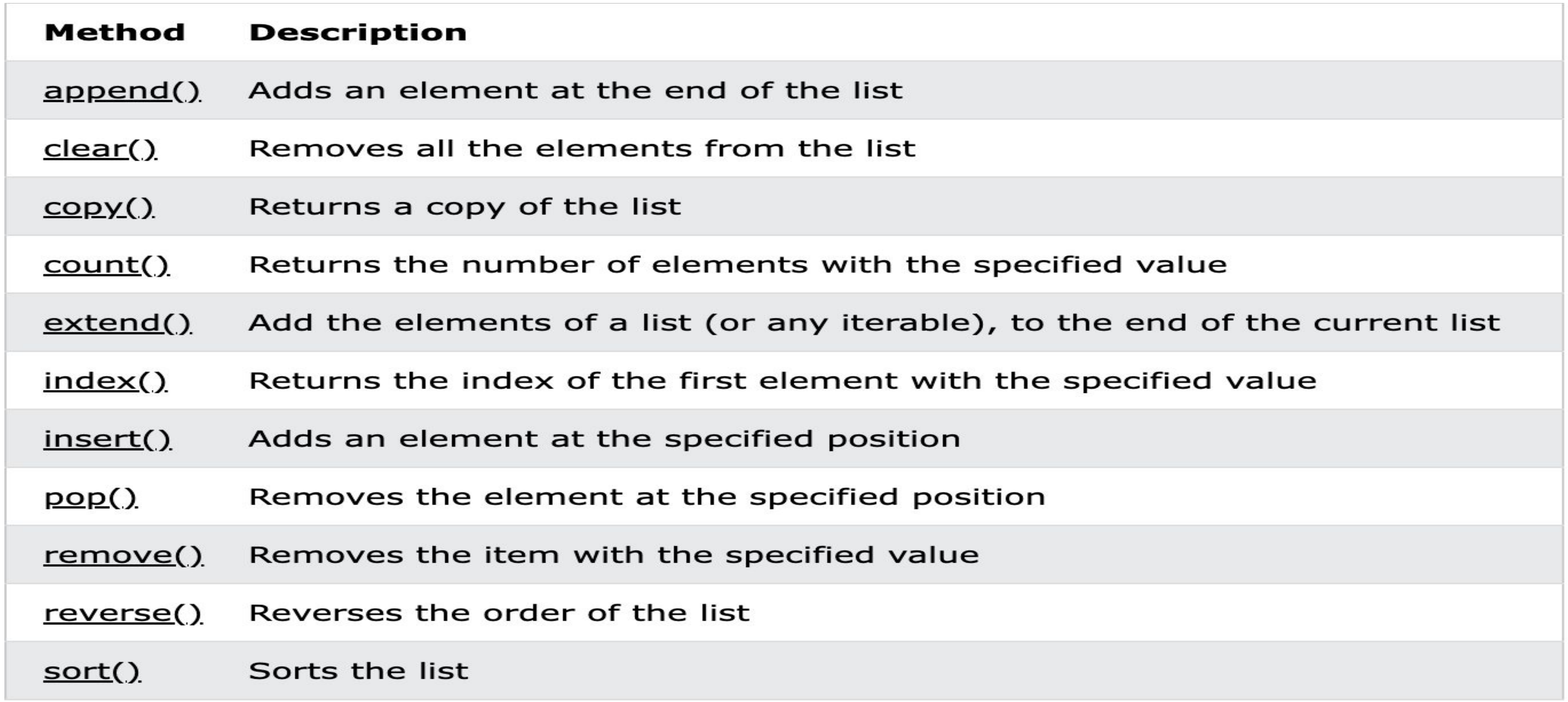
```
my_list = [[0]] * 5

my_list[0][0] = 1

my_list = [[1], [1], [1], [1], [1]] # Why?
```

Africa To Silicon Valley

Other List Methods



Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Tuples

What are Tuples?

- A tuple is a collection which is ordered, allows duplicates and is **unchangeable**. Tuples are also known as Immutable Lists.

- Tuples are written with round brackets.
    - fruits = ("apple", "banana", "cherry")
    - fruit = ("apple",)

A2SV
Africa To Silicon Valley

Creating Tuples

- Tuples are written with round brackets ().
- This is called ‘packing’ a tuple.

```
fruits = ("apple", "banana", "cherry")

fruit = ("apple",) # or just () to create an empty one
```

- The **tuple()** constructor:

```
fruits = tuple(["apple", "banana", "cherry"])

numbers = tuple()
```

A2SV Africa To Silicon Valley

Unpacking tuples

- In Python, we are also allowed to extract the values back into variables. This is called "unpacking".

```
fruits = ("apple", "banana", "cherry")

(green, yellow, red) = fruits

fruits = ("apple", "banana", "cherry", “oranges”, “pineapples”)

green, yellow, *red = fruits
```

A2SV Africa To Silicon Valley

Unpacking tuples

- In Python, we are also allowed to extract the values back into variables. This is called "unpacking".

```
fruits = ("apple", "banana", "cherry")

(green, yellow, red) = fruits

fruits = ("apple", "banana", "cherry", “oranges”, “pineapples”)

(green, yellow, *red) = fruits  #red = [“cherry”,“oranges”, “pineapples”]
```

Africa To Silicon Valley

Tuples

- Is it possible to
    - add an element to a Tuple? How?
    - delete an element?
    - join two tuples?

Tuple Similarities with List

- Similar data types
- Slicing and Indexing
- Similar Iteration

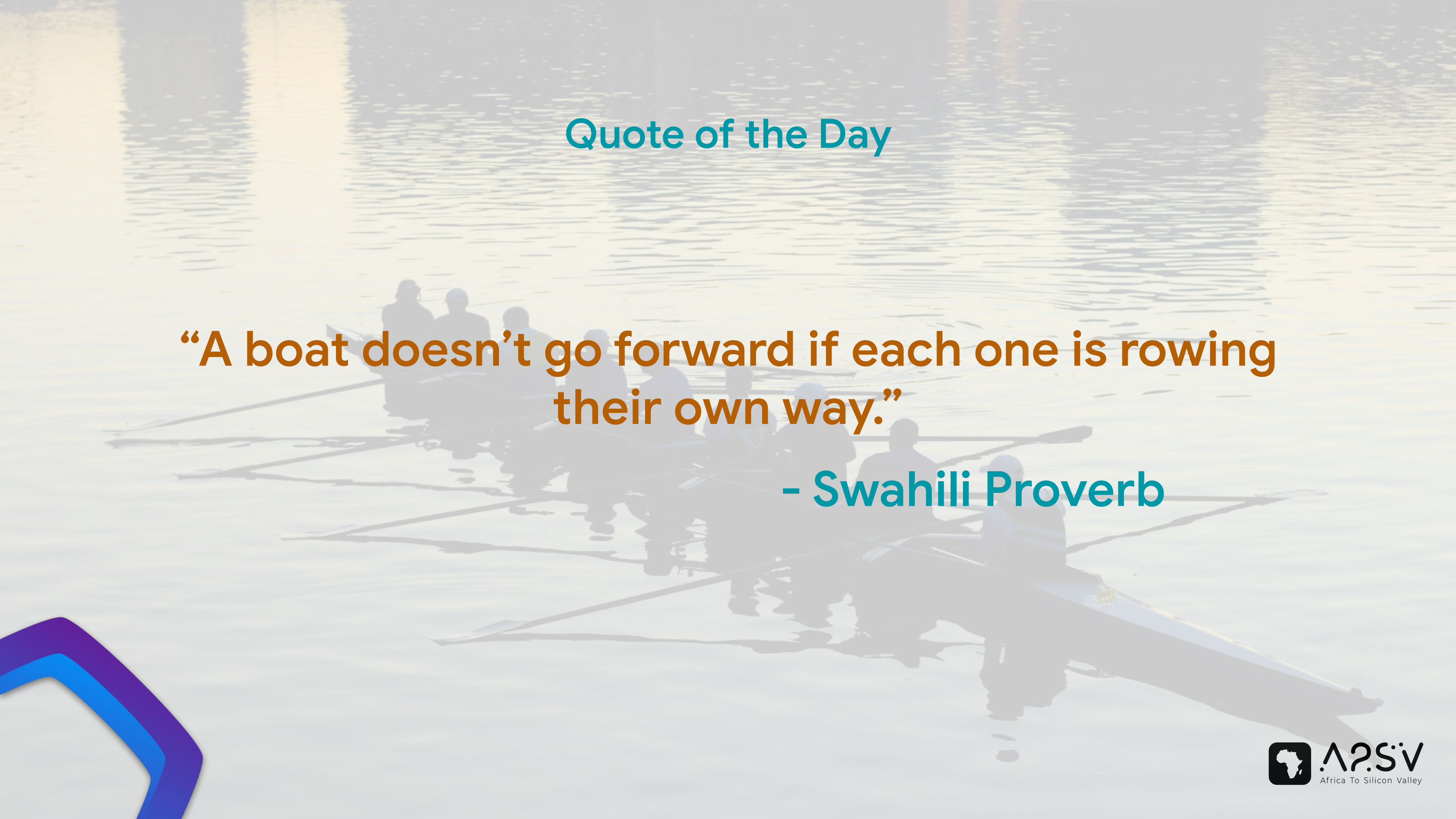
Q: Is it possible to have “Tuple Comprehension” ?

Tuple Methods

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Practice Problems

- List Comprehension
- Runner-up Score
- Nested LIsts
- Lists



Quote of the Day

“A boat doesn’t go forward if each one is rowing their own way.”

- Swahili Proverb

A2SV
Africa To Silicon Valley