Topic 2: Lists

Learning Outcomes:
(a) Identify concepts of lists

Table of Contents

1. **What is a list?**

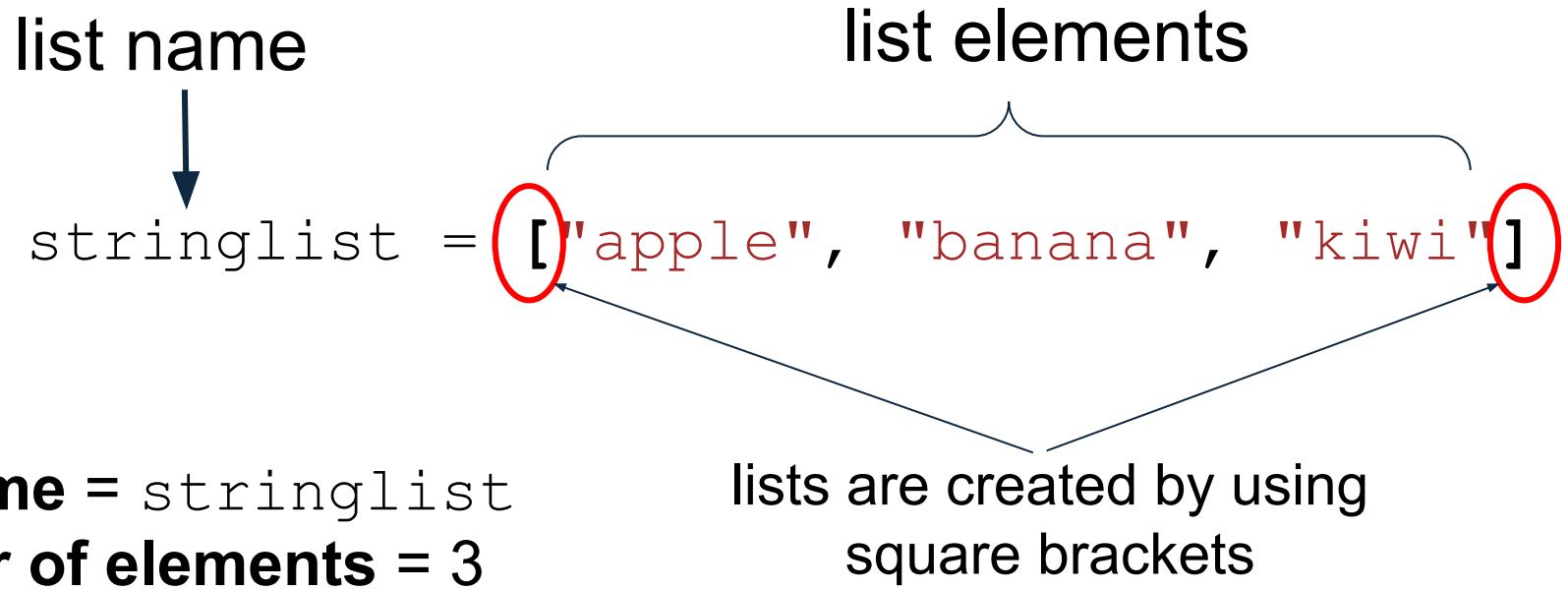
- A **list** is a collection of items stored in a single variable.
- It allows storage of **multiple data types (integers, strings, floats, etc.).**

```
my_list = [1, 2, 3, "apple", 4.5]
```

1. What is a list?

List Syntax

- Lists are defined using **square brackets** [ ].
- Items are **separated by commas**.



List name = `stringlist`
Number of elements = 3
List size = 3

2. Comparison between list and primitive data type

Feature	List	Primitive data type
Stores	multiple values	single value
Data type	can hold different data types	hold only 1 data type
Structure	use [ ] bracket	no bracket
Example	numbers = [1, 3, 5, 7] random = [‘a’, 1, True]	number = 10 name = “Yusuf”

3. What type of data that can be stored in a list?

Any data type can be stored in a list

A. Integers
B. Floats
C. Strings
D. Booleans
E. Mixed Data Type List
F. List of Lists (Nested List)

3. What type of data that can be stored in a list?

Any data type can be stored in a list

A) Integers

Code

```
numbers = [10, 20, 30, 40, 50]

print(numbers)  # Output: [10, 20, 30, 40, 50]
```

3. What type of data that can be stored in a list?

Any data type can be stored in a list

B) Floats

Code

```
decimal_numbers = [1.5, 2.3, 3.7, 4.1]

print(decimal_numbers)  # Output: [1.5, 2.3, 3.7, 4.1]
```

3. What type of data that can be stored in a list?

Any data type can be stored in a list

C) Strings

Code

```
fruits = ["apple", "banana", "cherry", "date"]

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

3. What type of data that can be stored in a list?

Any data type can be stored in a list

D) Booleans

Code

```
status = [True, False, True, False]

print(status)  # Output: [True, False, True, False]
```

3. What type of data that can be stored in a list?

Any data type can be stored in a list

E) Mixed Data Type List

Code

```
mixed_list = [25, "hello", 3.14, True]

print(mixed_list)  # Output: [25, 'hello', 3.14, True]
```

3. What type of data that can be stored in a list?

Any data type can be stored in a list

F) List of Lists (Nested List)

Code

```
nested_list = [[1, 2, 3], ["a", "b", "c"], [True, False]]

print(nested_list)

# Output: [[1, 2, 3], ['a', 'b', 'c'], [True, False]]
```

4. List index

- Refers to a position of an element in a list
- **Starts from the front**, index 0 **(positive index)**, read from left to right
- **Starts from the end to front**, index -1 **(negative index)**, read from right to left(Only in Python)

4. List index (Positive index)

Identify the index number corresponding to its element in a list

Example

```
fruitlist = ["apple", "banana", "kiwi", "durian", "guava",
"orange"]
```

element	"apple"	"banana"	"kiwi"	"durian"	"guava"	"orange"
positive index (left to right)	0	1	2	3	4	5

4. List index (Positive index)

Identify the index number corresponding to its element in a list

Example:

```
cities = ["Kuala Lumpur", "Tokyo", "New York", "Paris",
"Dubai"]

print(cities[0])  # Output: Kuala Lumpur
print(cities[2])  # Output: New York
print(cities[4])  # Output: Dubai
```

4. List index (Positive index)

Identify the index number corresponding to its element in a list

Example:

```
subjects = ["Math", "Science", "History", "English",
"Computer Science"]

print(subjects[0])  # Output: Math
print(subjects[2])  # Output: History
print(subjects[4])  # Output: Computer Science
```

4. List index (Negative index)

Identify the index number corresponding to its element in a list

Example

```
fruitlist = ["apple", "banana", "kiwi", "durian", "guava",
"orange"]
```

element	"apple"	"banana"	"kiwi"	"durian"	"guava"	"orange"
negative index (right to left)	-6	-5	-4	-3	-2	-1

4. List index (Negative index)

Identify the index number corresponding to its element in a list

Example:

```
cities = ["Kuala Lumpur", "Tokyo", "New York", "Paris",
"Dubai"]

print(cities[-1])   # Output: Dubai (last element)
print(cities[-3])   # Output: New York (third last element)
```

4. List index (Negative index)

Identify the index number corresponding to its element in a list

Example:

```
subjects = ["Math", "Science", "History", "English",
"Computer Science"]

print(subjects[-1])   # Output: Computer Science
print(subjects[-3])   # Output: History
print(subjects[-5])   # Output: Math
```

Exercise

1. Identify the list name in the following examples:
   a. `car = ["toyota", "honda", "masserati"]`
   b. `temperature = [1.5, 2.0, -1.0, 3.3, 1.5]`
   c. `variable = ['x', 'y', 'z', 'a', 'b', 'c', 'i' ,'jk',23]`

2. How many elements are in each list above?

3. What is the size of each list?

Topic 2: Lists

Learning Outcomes:
(a) Identify concepts of lists

Table of Contents

Characteristics of a List

✅ **Ordered** – Elements are stored in a specific sequence.
✅ **Mutable** – Can be modified (add, remove, update elements).
✅ **Allows Duplicates** – Can have repeated values.
✅ **Supports Different Data Types** – Can store integers, strings, floats, or even other lists.
✅ **Dynamic Size** – Can grow or shrink as needed.

Characteristic 1	Description
Ordered	• Lists maintain the order of elements as they are inserted. • The order of elements in a list is determined by their indices, **starting from 0**.

Code

```
fruits = ["Apple", "Banana", "Cherry"]
print(fruits[0])  # Output: Apple (order is maintained)
```

Code

```
days = ["Monday", "Tuesday", "Wednesday", "Thursday",
"Friday"]
print(days[0])   # Output: Monday
print(days[4])   # Output: Friday
```

Code

```
scores = [90, 85, 88, 92, 80]
print(scores)   # Output: [90, 85, 88, 92, 80]
```

Characteristic 2	Description
Changeable/ Mutable	• can **modify** the elements after creation by adding, removing, or changing elements.

Code

```
numbers = [10, 20, 30]
numbers[1] = 25  # Modifying the second element
print(numbers)  # Output: [10, 25, 30]
```

Code

```
names = ["Ali", "Sara", "John"]
names[1] = "Aisha"  # Changing "Sara" to "Aisha"
print(names)  # Output: ['Ali', 'Aisha', 'John']
```

Code

```
fruits = ["Apple", "Banana", "Cherry"]
fruits.append("Orange")  # Adding an element
fruits.remove("Banana")  # Removing an element
print(fruits)  # Output: ['Apple', 'Cherry', 'Orange']
```

Characteristic 3	Description
Allow Duplicate	• **Can creating a copy** of an existing list. • Can have duplicate values in the list

Code

```
names = ["Ali", "Sara", "Ali", "John"]
print(names)  # Output: ['Ali', 'Sara', 'Ali', 'John']
```

Code

```
prices = [15.99, 25.50, 15.99, 30.00, 25.50, 45.75]
print(prices)
# Output: [15.99, 25.50, 15.99, 30.00, 25.50, 45.75]
```

Code

```
temperatures = [30.5, 32.0, 31.5, 30.5, 32.0, 31.5]
print(temperatures)
# Output: [30.5, 32.0, 31.5, 30.5, 32.0, 31.5]
```

Characteristic 4	Description
Supports Different Data Types	A list can store multiple data types in a single structure.

Code

```
mixed_list = [5, "Hello", 3.5, True]
print(mixed_list)  # Output: [5, 'Hello', 3.5, True]
```

Code

```
info = ["Ali", 18, 3.5, True]
print(info)  # Output: ['Ali', 18, 3.5, True]
```

Code

```
nested_list = [[1, 2, 3], ["A", "B", "C"], [True, False]]
print(nested_list[1])  # Output: ['A', 'B', 'C']
```

Characteristic 5	Description
Dynamic Size	Lists can grow or shrink by adding or removing elements.

Code

```
numbers = [1, 2, 3, 4, 5]
numbers.append(6)      # Adding an element
numbers.remove(3)      # Removing element 3
print(numbers)         # Output: [1, 2, 4, 5, 6]
```

Code

```
colors = ["Red", "Blue"]
colors.append("Green")  # Adding another color
print(colors)  # Output: ['Red', 'Blue', 'Green']
```

Code

```
colors = ["Red", "Blue"]
colors.remove("Red")  #Remove another color
print(colors)  # Output: ['Blue']
```

Exercise

1. How many elements in the following list:
   a. `list1= ["pc", "laptop", "mobile", "computer"]`
   b. `list2= [1, 4, 3]`
   c. `list3= ["pc", "laptop", "mobile", "computer" ,[1, 4, 3]]`
   d. `list4= []`

2. What is the index number for `"mobile"` in the list `["pc", "laptop", "mobile", "computer" ,[1, 4, 3]]`?

3. What is the index number for `[1, 4, 3]` in the list `["pc", "laptop", "mobile", "computer" ,[1, 4, 3]]`?

Topic 2: Lists

Learning Outcomes:

(f) Identify pre-defined list functions (len( ), min( ), max( ), sum( ) and a method (append( ))

Table of Contents

In this lecture, we'll mainly focus on these four (4) pre defined functions & one (1) method in list:

1. len() function
2. min() function
3. max() function
4. sum() function
5. append() method

1. The len() function

What does it do?

- The **len()** function returns the number of elements in a list (or other sequence types like **dictionary**, **strings** and **tuples**).
- Syntax of len() function

```
len(object)
```

1. The len() function

Example 1: Count elements in a list

Code

```
carlist = ["Saga", "Waja", "Wira", "Persona"]
print("Length of the list:", len(carlist))
```

Output

```
Length of the list:4
```

1. The len() function

Example 2: Count elements in a list of numbers.

Code

```
my_list = [12, 23, 3, 42, 15]
length = len(my_list)
print("Length of the list:", length)
```

Output

```
Length of the list:5
```

1. The len() function

Example 3: Using len() in a conditional statement

Code

```
# List with multiple data types
mixed_list = [10, "hello", 3.14, True]

# Check the length of the list
if len(mixed_list) == 0:
    print("The list is empty.")
else:
    print("The list contains", len(mixed_list), "elements.")
```

Output

```
The list contains 4 elements
```

1. The len() function

Example 4: Using len() in a function to check list length

Code

```
def check_list_length(input_list):
    if len(input_list) == 0:
        return "The list is empty."
    else:
        return f"The list contains {len(input_list)} elements."

list1 = [1, 2, 3, 4, 5]
print(check_list_length(list1))

# Check length of a non-empty list
result = check_list_length(list1)
print(result)
```

Output

```
The list contains 5 elements
```

1. The len() function

Example 5: Count characters in sentence.

Code

```
sentence = "computer science is the gist of life"
print("Number of characters:", len(sentence))
```

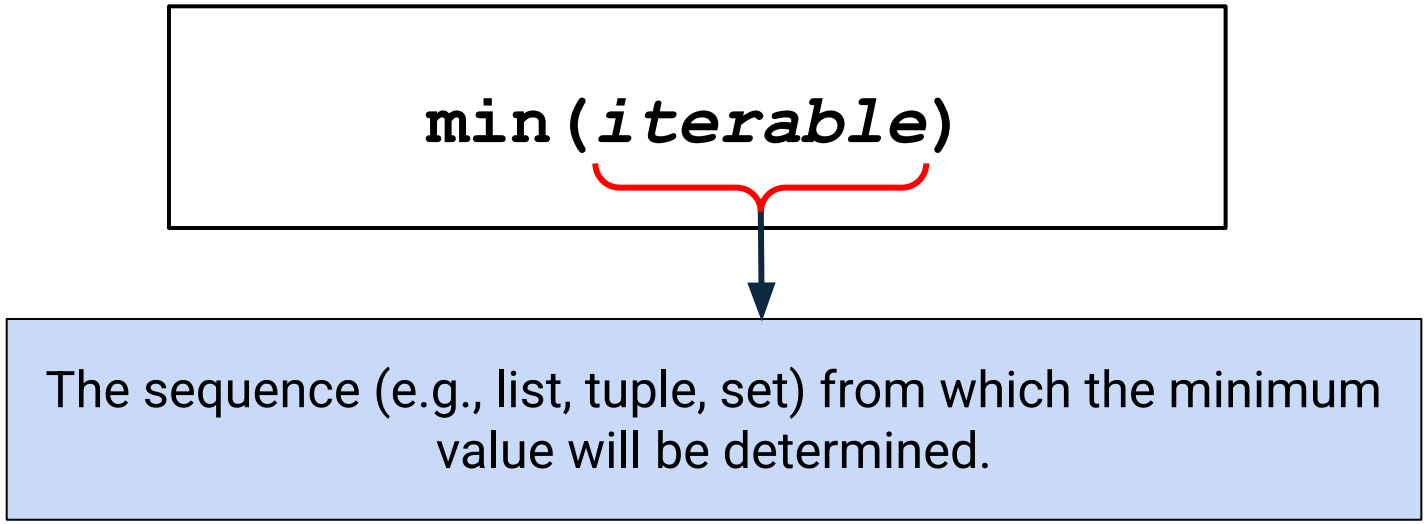
Output

```
Number of characters:36
```

2. The min() function

What does it do?

- The min() function **returns the minimum (smallest) value** from a list.
- Syntax of min() function



2. The min() function

Example 1: Find the minimum in a list of numbers.

Code

```
numbers = [5, 2, 8, 1, 6]
min_value = min(numbers)
print("Minimum value in the list:",min_value)
```

Output

```
Minimum value in the list: 1
```

2. The min() function

Example 2: Find the minimum string (alphabetically)

Code

```
# List of strings
string_list = ["apple", "banana", "cherry", "date"]

# Find the minimum string
min_string = min(string_list)

print("Minimum string:", min_string)
```

Output

```
Minimum string: apple
```

min() method returns the string that comes first in alphabetical order.

2. The min() function

Example 3: Using min() in a function

Code

```
# Example 1: Using min() in a selection (if statement)
numbers = [5, 3, 8, 2, 9]

# Using min() in an if statement to find the minimum value
if len(numbers) > 0:
    min_number = min(numbers)
    print("The minimum number in the list is:", min_number)
else:
    print("The list is empty.")
```

Output

```
The minimum number in the list is: 2
```

2. The min() function

Example 4: Using min() in a function with conditional statement.

Code

```
# Example: Using min() inside a custom function
def find_minimum(numbers):

    if len(numbers) == 0:
        print("The list is empty.")
    else:
       print("The minimum number in the list is:", min(numbers))

# Example usage:
numbers_list = [5, 3, 8, 2, 9]
find_minimum(numbers_list)
```

Output

```
The minimum number in the list is: 2
```

2. The min() function

Example 5: Using min() with mixed data.

Code

```
# Example: Using min() inside a mixed list
list1 = ['a', 'b', 'c', 50]
print(min(list1))
```

Output

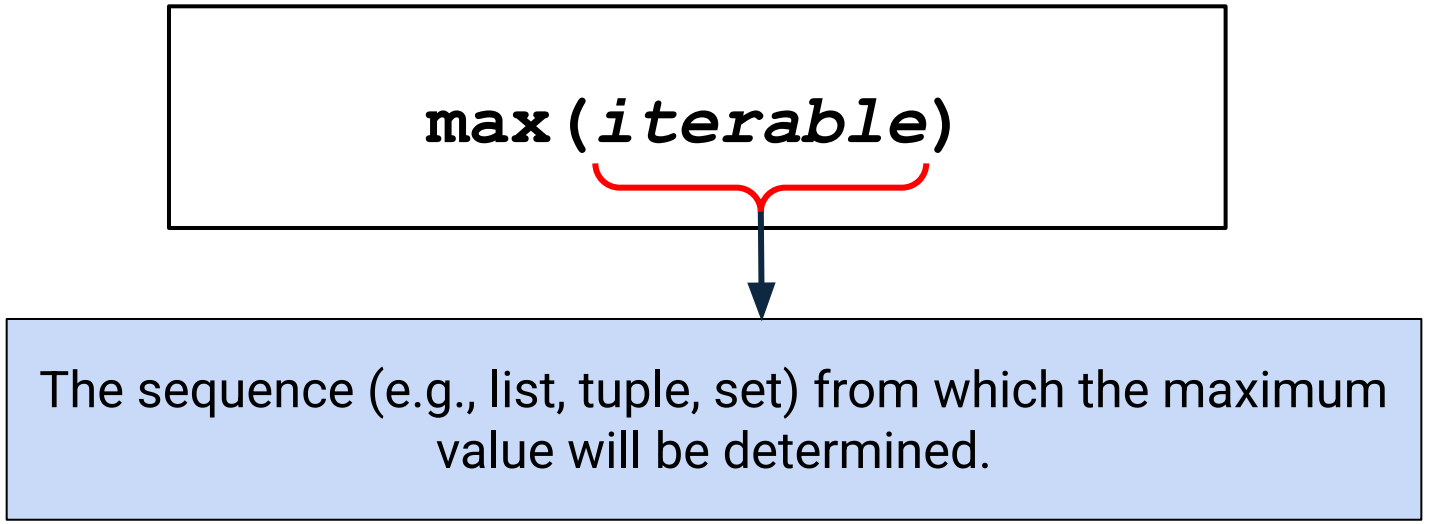
```
TypeError: '<' not supported between instances of
'int' and 'str'
```

System cannot compare different data type

3. The max() function

What does it do?

- The max() function **returns the maximum (largest) value** from a list.
- Syntax of max() function



3. The max() function

Example 1: Find the maximum in a list of numbers

Code

```
numbers = [5, 2, 8, 1, 6]
max_value = max(numbers)
print("Maximum value in the list:",max_value)
```

Output

```
Maximum value in the list: 8
```

3. The max() function

Example 2: Find the maximum string (alphabetically).

Code

```
# List of strings
string_list = ["apple", "banana", "cherry", "date"]

# Find the maximum string
max_string = max(string_list)

print("Maximum string:", max_string)
```

Output

```
Maximum string: date
```

3. The max() function

Example 3: Using max() in a function.

Code

```
def find_maximum(numbers):
    if len(numbers) == 0:
        return None
    else:
        return max(numbers)

# Example usage:
numbers_list = [1, 5, 3, 8, 6, 9]
maximum_number = find_maximum(numbers_list)

if maximum_number is not None:
    print("The maximum number in the list is:", maximum_number)
else:
    print("The list is empty.")
```

Output

```
The maximum number in the list is: 9
```

3. The max() function

Example 4: of max() and min() on list containing string and integer

Code

```
list = [1, "Saga", 3, 4, 5, 6, 7, 8, -1]
max(list)
min(list)
```

The error happen due to max() and min () function not able to process a list that contain different types of data.

Output

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: '>' not supported between instances of
'str' and 'int'
```

TRY THIS

Write a Python program that takes a list of ages as input and calculates the following statistics:

- The number of ages in the list.
- The youngest age in the list.
- The oldest age in the list.

You should use the len(), min(), and max() functions to compute these statistics.

Topic 2: Lists

Learning Outcomes:

(f) Identify pre-defined list functions (len( ), min( ), max( ), sum( ) and a method (append( ))

Table of Contents

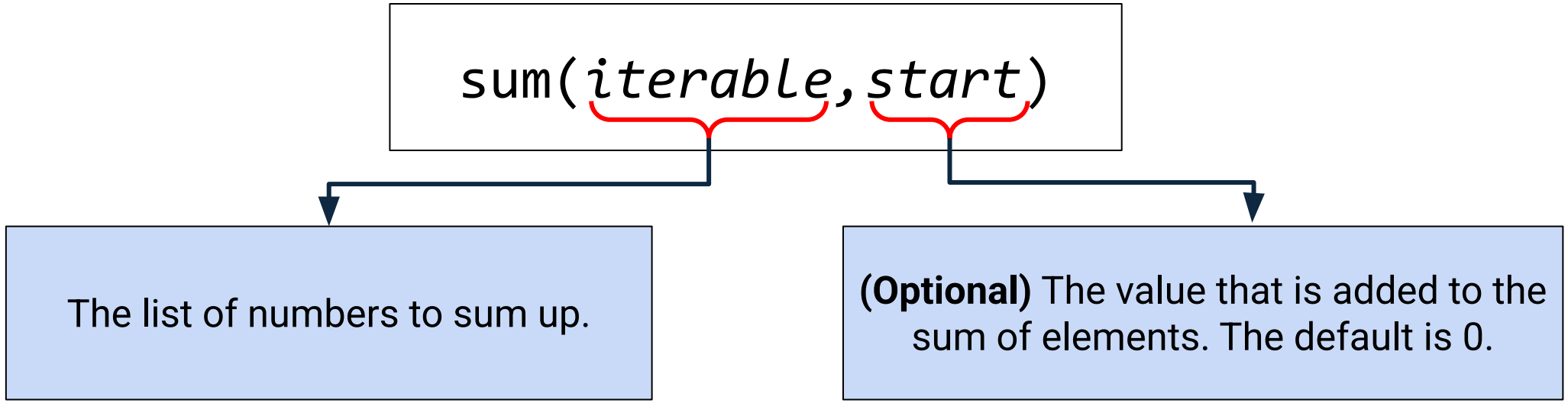
In this lecture, we'll mainly focus on these four (4) pre defined functions & one (1) method in list:

1. len() function
2. min() function
3. max() function
4. sum() function
5. append() method

4. The sum() function

What does it do?

- The sum() function **returns the sum of all the numerical values** in a list.
- Syntax of sum() function



Example 1: Sum of numbers in a list

```
numbers = [1, 2, 3, 4, 5]
total_sum = sum(numbers)
print("Sum of all elements in the list:", total_sum)
```

Output

```
Sum of all elements in the list: 15
```

Example 2: Sum with a starting value.

```
numbers = [1, 2, 3, 4, 5]
total_sum = sum(numbers,100)
print("Sum of all elements in the list:", total_sum)
```

Output

```
Sum of all elements in the list: 115
```

Example 3: sum() function on string list.

```
# List of strings
string_list = ["apple", "kiwi"]

# Find the minimum string
sum_string = sum(string_list)

print(sum_string)
```

Output

```
ERROR!
Traceback (most recent call last):
  File "<main.py>", line 5, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
=== Code Exited With Errors ===
```

Difference between a function and a method

FUNCTION vs METHOD

Function	Description
abs()	Returns the absolute value of a number
all()	Returns True if all items in an iterable object are true
any()	Returns True if any item in an iterable object is true

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

```
fruits = ["apple", "banana", "cherry"]

x = any(fruits)

print(x)
```

how to call/invoke

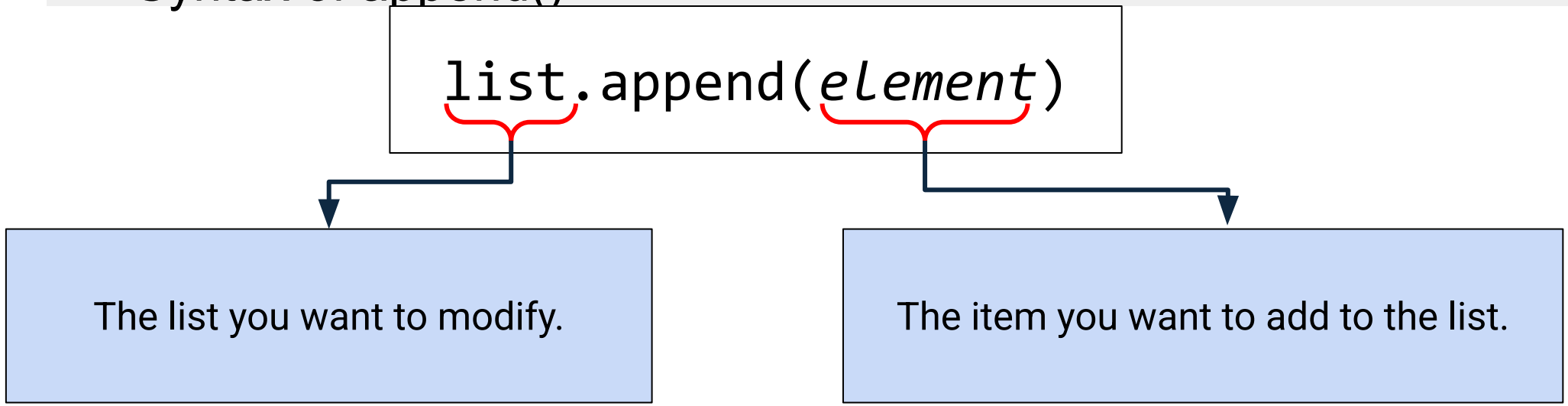
```
fruits = ["apple", "banana", "cherry"]

x = fruits.copy()

print(x)
```

NOTE:
Method is similar to function but attached to an object. Later, you will use this in lists etc.

5. The append( ) method in Python

What does it do?

- The `append()` method in Python is used to **add an element to the end of a list**.
- It modifies the list in place and does not return a new list.
- Syntax of append()



Example 1: Using append( ) with a List.

```
fruits = ["apple", "banana", "cherry"]
fruits.append("orange")
print(fruits)
```

Output

```
['apple', 'banana', 'cherry', 'orange']
```

Example 2: Using `append()` with a Condition

```
numbers_list = [1, 2, 3, 4, 5]
new_number = 6

if new_number not in numbers_list:
    numbers_list.append(new_number)

print(numbers_list)
```

`not in` means that the number is not in the list

Output

```
[1, 2, 3, 4, 5, 6]
```

Example 3: Using append( ) in a Function.

```
def add_item(my_list, item):
    if item:  # Ensures item is not empty or None
        my_list.append(item)

items = ["pen", "pencil"]
add_item(items, "eraser")

print(items)
```

Output

```
['pen', 'pencil', 'eraser']
```

Example 4: Appending Numbers from a Range

```
number_list = []  # Empty list

for i in range(5):
    number_list.append(i)  # Add each number to the list

print(number_list)
```

Output

```
[0, 1, 2, 3, 4]
```

Example 5: Appending even numbers only

```
even_numbers = []

for i in range(10):
    if i % 2 == 0:  # Check if number is even
        even_numbers.append(i)

print(even_numbers)
```

Output

```
[0, 2, 4, 6, 8]
```

Example 6: Apply what you've learned so far. Count the number of positive number and find their sum

```
def sum_count_positiveNumbers(numbers_list):
    positive_numbers = []
    for num in numbers_list:
        if num > 0:
            positive_numbers.append(num)

    sum_of_positives = sum(positive_numbers)
    count_of_positives = len(positive_numbers)

    print("Sum of positive numbers:", sum_of_positives)
    print("Count of positive numbers:", count_of_positives)

# Example usage:
sample_list = [-1, 2, 3, -4, 5]
result = sum_count_positiveNumbers(sample_list )
```

Output

```
Sum of positive numbers: 10
Count of positive numbers: 3
```

TRY THIS

Write a Python program that allows the user to enter a series of numbers. The program should:

- Add each number to a list using the append() method.
- Calculate and display the total sum of all numbers using the sum() function.