Lesson 3: Lists and Functions in Python

Marcin Kurzyna

Lecture Goals

This lecture introduces two key Python concepts: **lists** for storing multiple values, and **functions** for organizing code into reusable parts. By the end of this lecture, students should be able to:

- Create and use lists.
- Write and call functions with parameters and return values.
- Understand how Python organizes data and behavior.

1 Lists

A list is a collection of elements that can store multiple values in a single variable. Lists in Python are flexible and can hold elements of different types, although typically they store similar items.

1.1 Creating Lists

You can create a list like this:

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

1.2 Accessing and Modifying Elements

Lists use zero-based indexing (first element has index 0):

```
print(numbers[0])    # prints 10
numbers[1] = 25      # change second element
print(numbers)       # [10, 25, 30, 40, 50]
```

1.3 Adding and Removing Elements

```
numbers.append(60)    # add to end
numbers.remove(25)    # remove a specific value
print(numbers)
```

1.4 Looping Through Lists

You can use a `for` loop to process all elements:

```
for n in numbers:
    print(n)
```

You can also use the `range()` function to loop by index:

```
for i in range(len(numbers)):
    print(i, numbers[i])
```

2 Example: Average of Numbers

```
numbers = [10, 20, 30, 40, 50]
total = 0
for n in numbers:
    total += n
average = total / len(numbers)
print("Average =", average)
```

3 Functions

A function is a reusable block of code that performs a specific task. Functions help organize code and make programs easier to understand and maintain.

3.1 Defining a Function

General form:

```
def function_name(parameters):
    # function body
    # optional return statement
```

Example:

```
def greet():
    print("Hello!")
```

3.2 Calling a Function

To execute the function:

```
greet()
```

3.3 Functions with Parameters

Functions can accept input values:

```
def greet_user(name):
    print("Hello,", name + "!")
```

Call it like this:

```
greet_user("Alice")
```

3.4 Functions with Return Values

Some functions return results:

```
def add(a, b):
    return a + b
```

Usage:

```
result = add(3, 5)
print(result)
```

3.5 Combining Functions and Lists

Functions can be used to process lists or perform calculations:

```
def sum_list(lst):
    total = 0
    for n in lst:
        total += n
    return total

values = [1, 2, 3, 4, 5]
result = sum_list(values)
print("Sum =", result)
```

Summary

This lecture introduced:

- Lists: structures for storing multiple values.
- Functions: reusable code blocks that can take parameters and return results.
- Using loops with lists to process data collections.

4 Exercises

1. Create a list of 5 integers and print all elements using a `for` loop.
2. Write a program that finds the largest number in a list.
3. Compute the sum and average of numbers in a list using a function.
4. Write a function $\texttt{is}_{e}ven(n)thatreturns\texttt{True}ifanumberiseven.$
5. Write a program that counts how many even numbers are in a list.
6. Create a function $\texttt{reverse}_{l}ist(lst)thatprintstheelementsinreverseorder.$

7. Write a function `max_of_two(a, b)` that returns the larger of two numbers.
8. Combine functions and lists: Write a program that creates a list of numbers 1–10 and prints their squares using a separate function.
9. Challenge: Write a recursive function that calculates the factorial of a number.
10. Challenge: Write a function that checks whether a given word is a palindrome.