ISA 419: Data-Driven Security

03: Python Functions

Fadel M. Megahed, PhD

Professor
Farmer School of Business
Miami University

@FadelMegahed
fmegahed
fmegahed@miamioh.edu
? Automated Scheduler for Office Hours

Spring 2025

Quick Refresher of Last Class

- ✓ Use pseudocode to map out a problem.
- ✓ Python syntax, data types, and data structures.
- ✓ Convert data types using type casting.
- ✓ Manipulate lists and use methods on lists.

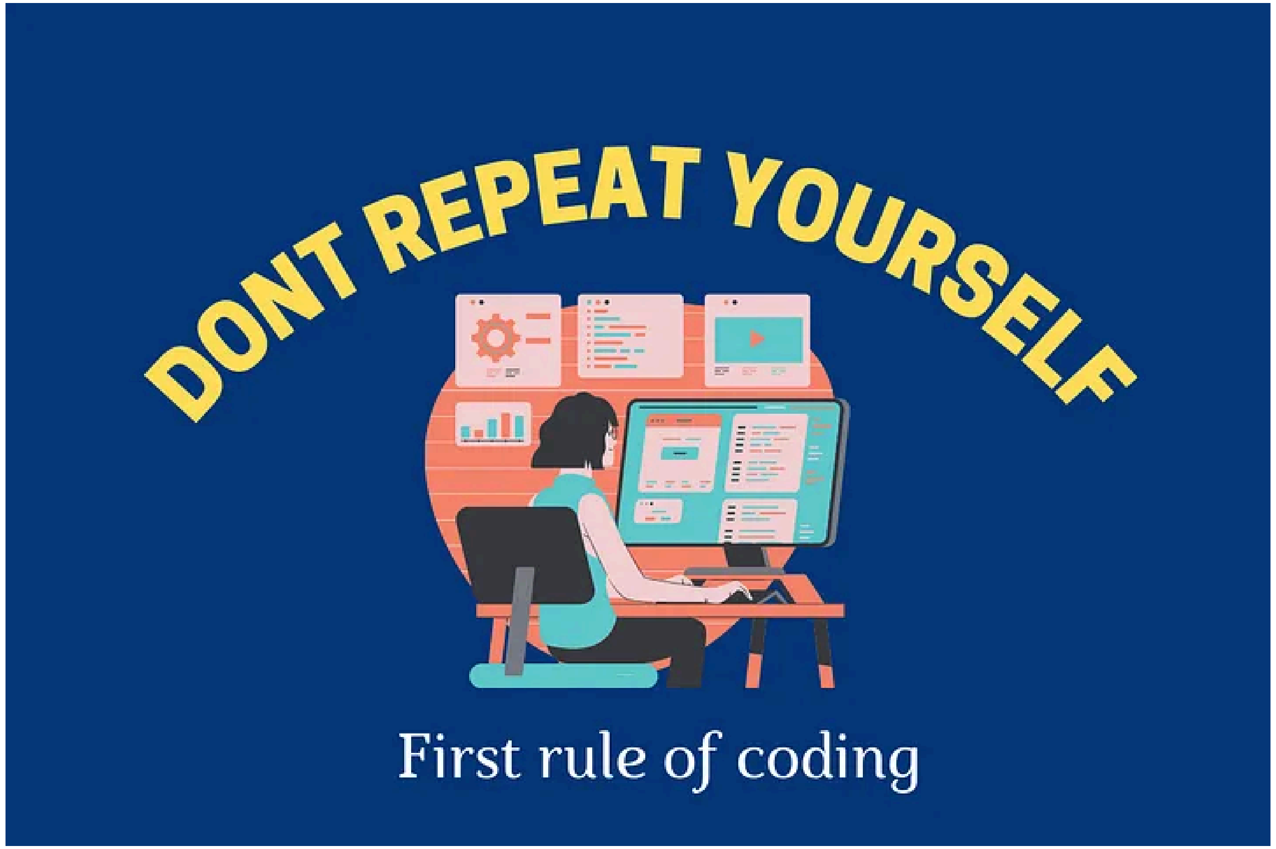
Learning Objectives for Today's Class

- Understand the anatomy of a Python function, use arguments correctly, and construct your first function.
- Utilize built-in and anonymous functions (`map`, `lambda`, `filter`).
- Analyze your second dataset.

The Anatomy of a Python Function

One Motivation for Functions

- **Reusability**: Functions allow you to reuse code.



Source: Sonika Baniya (2021). First rule of coding: DRY (Don't Repeat Yourself).

What is a Function?

- A function is a **block of reusable code** that only runs when called.

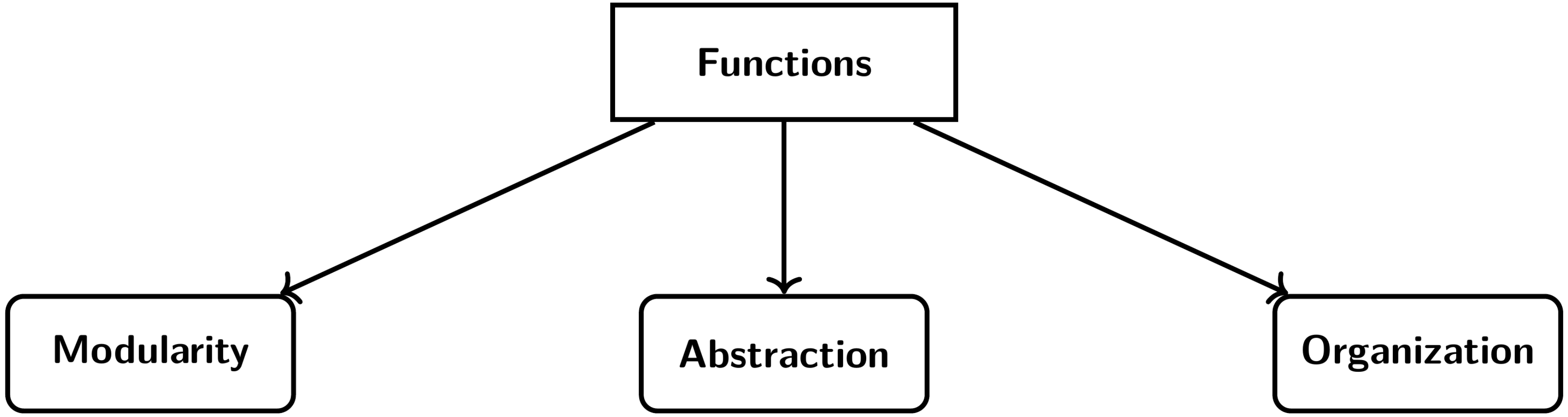


Image Credits: Created by the author (Fadel M. Megahed) for our ISA 419 course.

Recall: Python List Functions

04:00

Task Q1 Q2

- In class02, we introduced the following functions: `len()`, `max()`, `min()`, `sum()`, `sort()`, `index()`, `append()`, `pop()`, and `remove()`.
- For each of the above functions, what do you expect the function to **return**?
- How is the name of the function related to the **action** it performs?

The Anatomy of a Python Function

```
def add_numbers(a, b):                      # A: The function definition

    """ This function sums two numbers """  # B: The docstring

    result = a + b                          # C: The body of the function
    return result                           # C: The body of the function


# Using the function
add_numbers(3, 5)                           # D: Calling the function
```

```
## 8
```

Footnotes:

- Everything **within the function**, post the definition line, **is indented**.
- Vertical spacing here is for clarity, but not required (and likely violates Python's best practies).

The Anatomy of a Python Function (Cont.)

A: The function definition:

- `def` is a **keyword** that tells Python you are defining a function.
- `add_numbers` is the **name** of the function, which should always be followed by parentheses and a colon.
- `a` and `b` are the **parameters** of the function.

B: The docstring:

- Optional: A **docstring** is a string that describes what the function does.

C: The body of the function:

- This is where the function does its work.
- The body is indented.
- Typically includes a `return` statement at the end.

D: Calling the function:

- This is where you **call** the function, i.e., tell Python to execute the code inside your function.

Python Parameters vs Arguments

- **Parameters** are the variables in the function definition.
  - `a` and `b` are the parameters in the `add_numbers` function.
- **Arguments** are the values passed to the function when it is called.
  - `3` and `5` are the arguments in the `add_numbers` function.

```python
def add_numbers(a, b):
    """ This function sums two numbers ""
    result = a + b
    return result


# Using the function
add_numbers(3, 5)
```

```
## 8
```

Python Parameters and Arguments

Python allows for several methods of passing arguments to a function. These include, but are not limited to the following:

- **Positional Arguments**: The arguments are passed to the function in the order in which they are defined.
- **Keyword Arguments**: The arguments are passed to the function with the parameter name.
- **Combination of Positional and Keyword Arguments**: The arguments are passed to the function in the order in which they are defined, **followed by the keyword arguments**.

Class Activity: Modify the `add_numbers` Function

03:00

Modify the function `add_numbers` to take a list of numbers and return the sum of the numbers.

```
# Hints:
# -------
# 1. Change the function name to sum_numbers
#    (so you do not have two functions with the same name).
# 2. Change the parameters to a single parameter named numbers_list.
# 3. Capitalize on the fact that the parameter input is now a list.
```

Functions: Good Practices

- **Function Name:** Choose a descriptive name for your function.
    - The name should describe what the function does.
- **Type Hints:** You can specify the type of the parameters and the return type.
    - This is **not enforced by Python**, but it is a good practice.
    - For example:
        - `def add_numbers(a: int, b: int) -> int:` or
        - `def add_numbers(a: float, b: float) -> float:`.
- **Docstrings:** Always include a docstring to describe what the function does.
    - This is a good practice and is used by Python's built-in `help()` function.
- **Return Statement:** Always include a `return` statement.
    - If you do not include a `return` statement, the function will return `None`.

Built-In and Anonymous Functions in Python (`map`, `lambda`, `filter`)

The map Function

- The `map` function applies a given function to each item of an *iterable* (e.g., `list`). Its synatx is:
    - `map(function, iterable)`.
        - The `function` is the function we want to apply.
        - The `iterable` is what we want to apply the function across.
- The `map` function returns a **map object**, which is an iterator.
- To get the **results**, you must *type convert* the map object to a list.

```python
def square(x: float) -> float:
    """This function squares a number"""
    return x ** 2

# Using the map function
numbers = [1, 2, 3, 4, 5]
map_operation = map(square, numbers)
squared_numbers = list(map_operation)

print('The map operation:', map_operation
```

```
## The map operation:
## <map object at 0x00000236C3582A40>
##
##
##
## The squared numbers:
## [1, 4, 9, 16, 25]
```

The `lambda` Function

- The `lambda` function is an **anonymous function**, defined using the `lambda` keyword:
  - **Anonymous:** They are **not** declared in the standard manner by using the `def` keyword.
  - **Compact:** They allow writing functions in a concise way, often for short-term/throwaway functions.
  - **Single Expression:** The body of a lambda is limited to just one expression. No statements or annotations are allowed; the function body is purely a single expression.

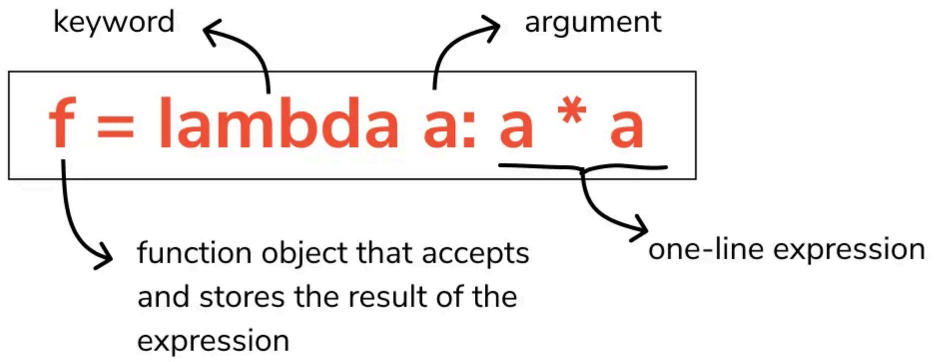

keyword
argument
f = lambda a: a * a
function object that accepts and stores the result of the expression
one-line expression

Image Source: John Vastola (2021). Mastering Lambda Expressions in Python: A Hands-On Guide.

The lambda Function (Cont.)

```python
numbers = [1, 2, 3, 4, 5]

map_operation = map(lambda x: x ** 2, numbers)

squared_numbers = list(map_operation)

print(
  'The map operation:', map_operation, '\n',
  'The squared numbers:', squared_numbers, sep ='\n'
)
```

```
## The map operation:
## <map object at 0x00000236C3532740>
##
##
## The squared numbers:
## [1, 4, 9, 16, 25]
```

The `filter` Function

- The `filter` function constructs an iterator from elements of an iterable for which a function returns `True`.
    - Its syntax is: `filter(function, iterable)`.
- The `filter` function returns a **filter object**, which is an iterator.
- To get the **results**, you must *type convert* the filter object to a list.

```
def is_even(x: int) -> bool:
    """This function checks if a number i
    return x % 2 == 0

# Using the filter function
numbers = [1, 2, 3, 4, 5]
filter_step = filter(is_even, numbers)
even_numbers = list(filter_step)

print('The filter operation:', filter_ste
'The even numbers:', even_numbers, sep ='
```

```
## The filter operation:
## <filter object at 0x00000236C35B74F0>
##
##
## The even numbers:
## [2, 4]
```

Evaluating your Understanding so Far: A Kahoot

Let's evaluate your understanding of the material so far

- Go to Kahoot and enter the game pin shown on screen.
- You will be asked to answer **7 multiple choice questions**.
- You will receive **points** for answering each question **correctly** and **quickly**, i.e., your points are impacted by your speed in addition to obviously answering each question correctly.
- The winner 🏆 (i.e., the one with the most points after the 7 questions) receives a $10 Starbucks ☕ gift card.

Analyzing Your Second Dataset

Analyzing a Simulated Equifax Breach Dataset

15:00

Task | Task 1 | Task 2 | Task 3 | Task 4 | Task 5

- Download the `simulated_equifax_breach_data.csv` file from Canvas.
- Load the dataset into Google Colab.
- Answer the questions in the next tabs.
- You can work in groups of 2-3 students.

Recap

Summary of Main Points

By now, you should be able to do the following:

- Understand the anatomy of a Python function, use arguments correctly, and construct your first function.
- Utilize built-in and anonymous functions (`map`, `lambda`, `filter`).
- Analyze your second dataset.

📝 Review and Clarification 📝

- **Class Notes**: Take some time to revisit your class notes for key insights and concepts.
- **Zoom Recording**: The recording of today's class will be made available on Canvas approximately 3-4 hours after the end of class.
- **Questions**: Please don't hesitate to ask for clarification on any topics discussed in class. It's crucial not to let questions accumulate.