2. Python Functions and Methods

```
– By Srikanth Pinisetti (Data Engineer)
```

Function:

- A function is a block of reusable code that performs a specific task.
- It is defined using the `def` keyword followed by a function name, parentheses `()`, and a colon `:`.
- Functions can accept arguments (input data), perform operations, and return results.
- Functions are not tied to any specific object or class.
- They can be called independently, without the need for any object or class instance.
- Examples of functions include `print()`, `input()`, `range()`, `len()`, `sum()`, etc.

Method:

- A method is a function that is associated with a specific object or class.
- It operates on the data contained within an object of a particular type.
- Methods are called using dot notation (`object.method()`).
- They are defined within a class definition.
- Methods can access and modify the object's attributes.
- Examples of methods include `.upper()`, `.lower()`, `.strip()` for strings, `.append()`, `.pop()` for lists, etc.

Key Differences:

1. **Association with Object or Class**:
   - Functions are not associated with any specific object or class. They are standalone blocks of code.
   - Methods are associated with specific objects or classes. They operate on the data contained within those objects.
2. **How They Are Called**:
   - Functions are called independently by their names, passing arguments if necessary.
   - Methods are called using dot notation on an object or class instance, like `object.method()`.
3. **Definition**:
   - Functions are defined using the `def` keyword outside of any class definition.
   - Methods are defined within a class definition and are typically intended to operate on instances of that class.
4. **Access to Data**:
   - Functions generally operate on data passed to them as arguments.

- Methods have access to the data (attributes) within the object they belong to and can manipulate that data directly.

Example:

let's illustrate the differences between functions and methods with examples:

Function Example:

```
# Function to calculate the area of a rectangle
def calculate_area(length, width):
    return length * width

# Calling the function
area = calculate_area(5, 4)
print("Area:", area)  # Output: Area: 20
```

In this example, `calculate_area()` is a function that takes `length` and `width` as arguments and returns the area of a rectangle. It operates independently and doesn't belong to any specific object or class.

Method Example:

```
# Class definition for a Rectangle
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    # Method to calculate the area of the rectangle
    def calculate_area(self):
        return self.length * self.width

# Creating an instance of the Rectangle class
rectangle = Rectangle(5, 4)

# Calling the method to calculate the area
area = rectangle.calculate_area()
print("Area:", area)  # Output: Area: 20
```

In this example, `calculate_area()` is a method defined within the `Rectangle` class. It operates on the `length` and `width` attributes of the `Rectangle` object (`self`). The method is called using dot notation (`rectangle.calculate_area()`), where `rectangle` is an instance of the `Rectangle` class.

Key Differences:

- **Association with Object/Class**:
  - The `calculate_area()` function is not associated with any specific object or class.

    - The `calculate_area()` method is associated with the `Rectangle` class and operates on instances of that class.
- **How They Are Called**:
    - The function is called by its name (`calculate_area()`), passing arguments explicitly.
    - The method is called using dot notation (`rectangle.calculate_area()`), accessing it through an instance of the class.
- **Definition**:
    - The function is defined outside of any class definition.
    - The method is defined within the class definition using `def` .
- **Access to Data**:
    - The function receives data explicitly through its arguments.
    - The method has access to the data (attributes) within the object it belongs to (`self.length` , `self.width`).