Programming Paradigms

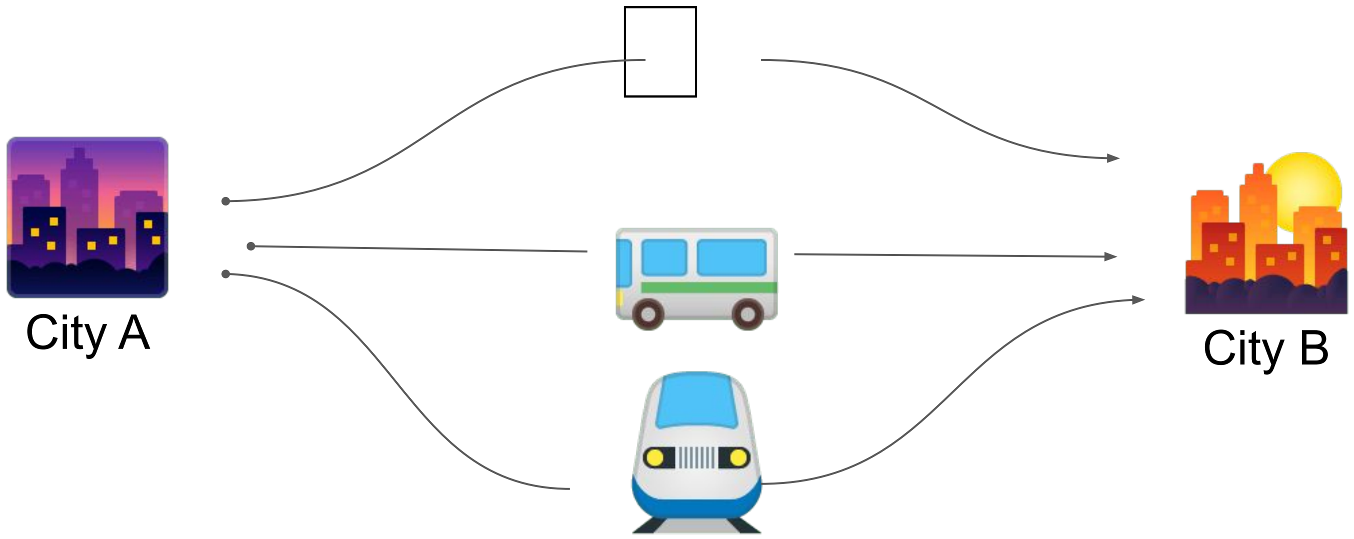
Analytics Vidhya

Selecting the Means of Transport

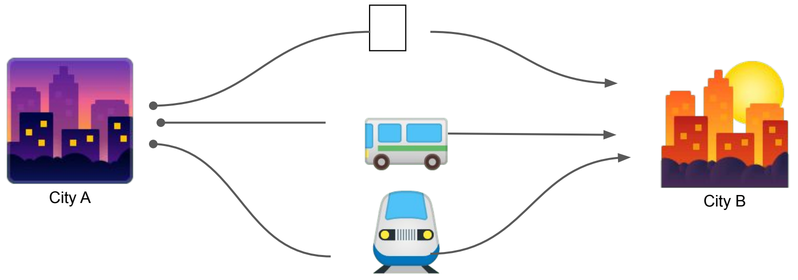
City A

City B

Selecting the Means of Transport



Selecting the Means of Transport



There is no RIGHT or WRONG here

Factors which might influence decision

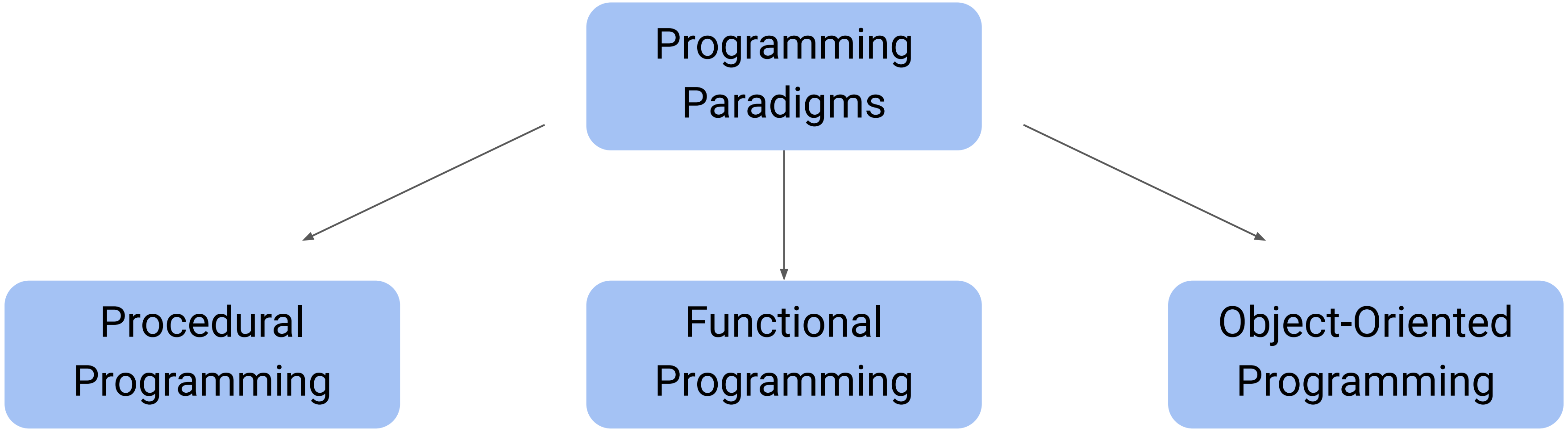
- Price of different transportations
- Frequency of trips
- Closeness of Airport/stations
- Time of departure/ arrival

What are Programming Paradigms?

A programming paradigm is a style or way of programming

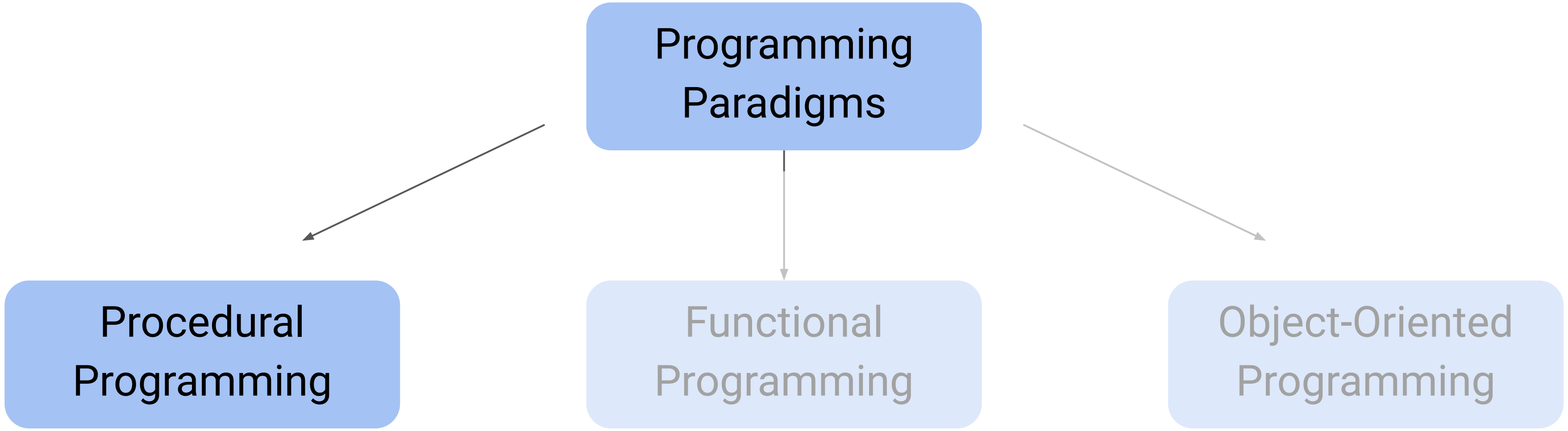
What are Programming Paradigms?

A programming paradigm is a style or way of programming



Procedural Programming

A programming paradigm is a style or way of programming



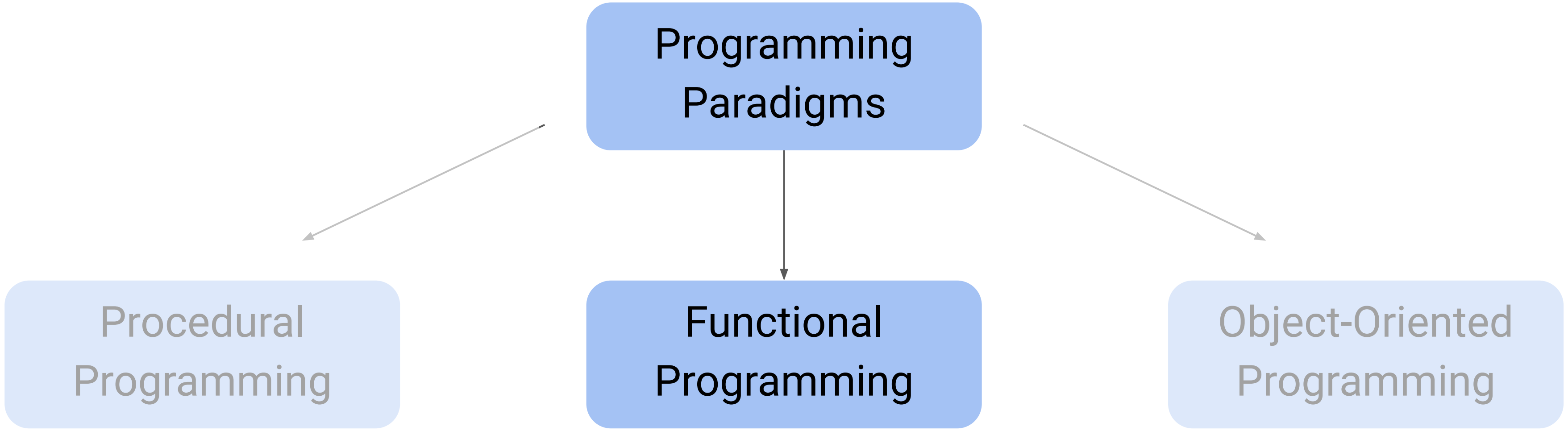
Procedural Programming

- Most basic form of coding based on the concept of ‘*procedural calls*’
- Also called *imperative programming paradigm*
- Code is structured hierarchically into blocks (loops, conditions)
- Difficult to write and maintain large and complex codes

Analytics Vidhya

Functional Programming

A programming paradigm is a style or way of programming



Functional Programming

- Uses Functions as the fundamental building blocks
- Each function performs a singular task
- Efficient and easy to debug
- Statements in programming may not necessarily follow an order
- Follows a top down approach

Analytics Vidhya

Functional Programming

$$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}$$

Functional Programming

$$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}$$

Subtract Function

Multiplication Function

Functional Programming

$$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}$$

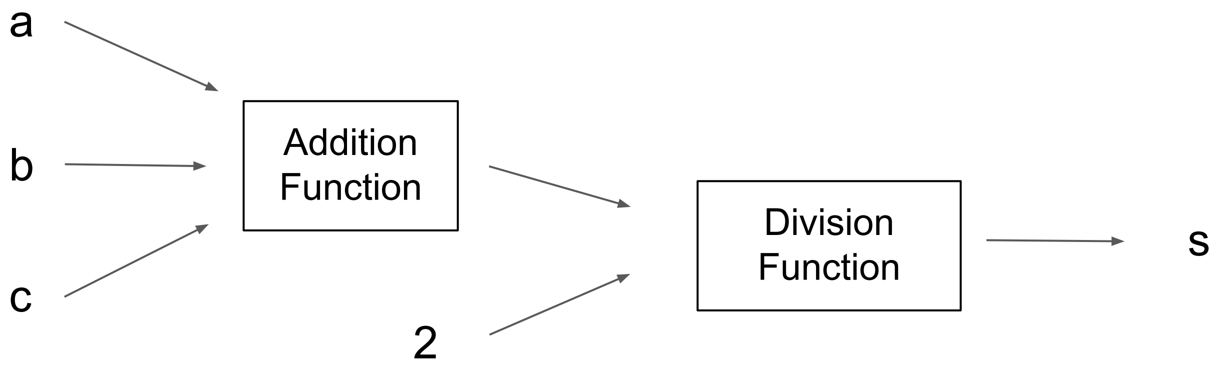
$$s = \frac{a+b+c}{2}$$

Subtract Function	Multiplication Function	Addition Function	Division Function

Functional Programming

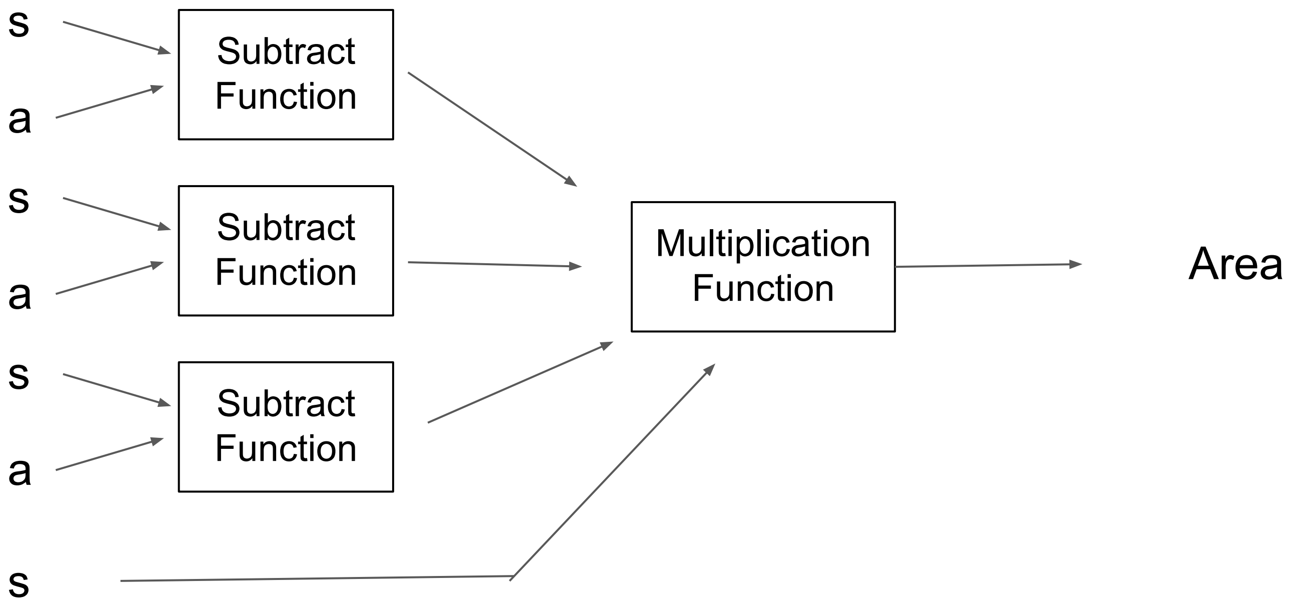
$$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}$$

$$s = \frac{a+b+c}{2}$$



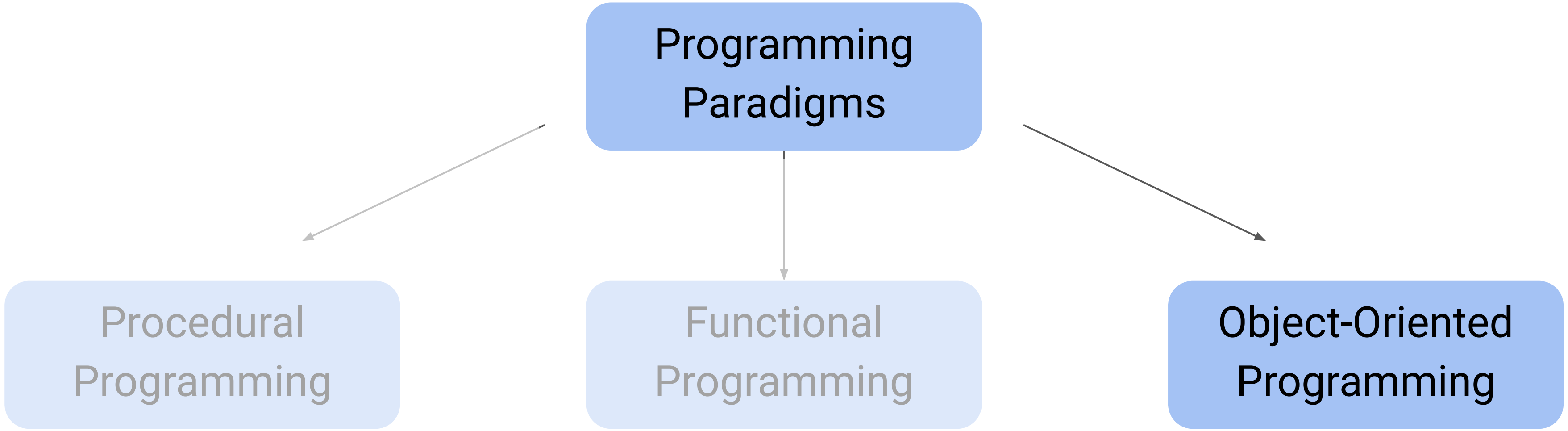
Functional Programming

$$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}$$



Object Oriented Programming

A programming paradigm is a style or way of programming



Object Oriented Programming

- Groups all the data and code within a single structure *(Class)*
- All data is stored as *Classes* and *objects*
- Modular and Organised code
- Easier to reuse code and reduces redundant code blocks
- Follows bottom up approach

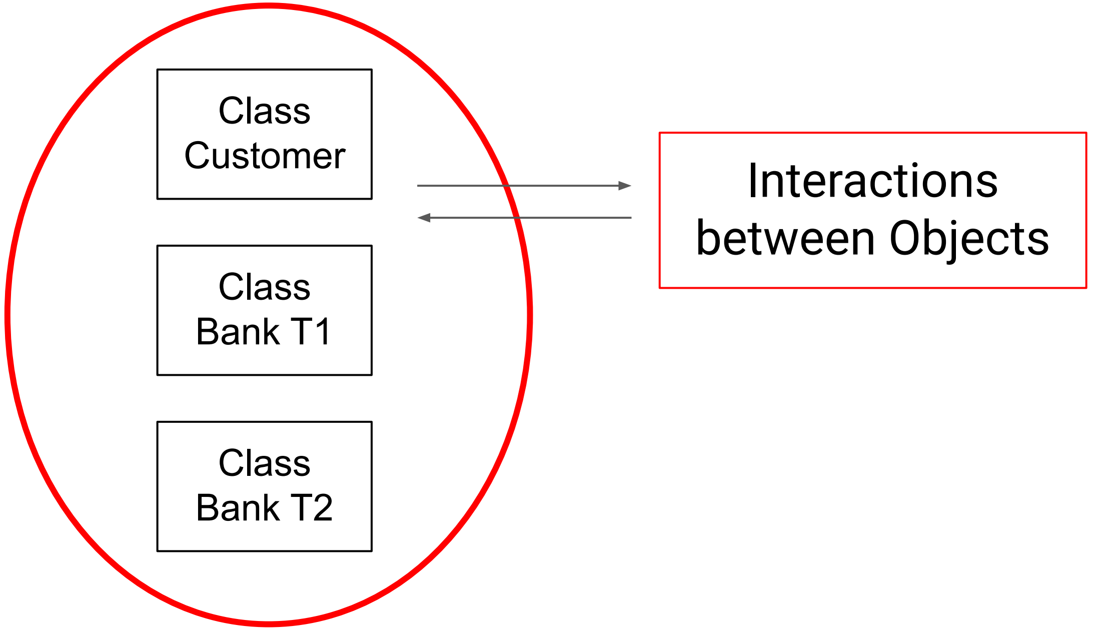
Object Oriented Programming

Class Customer

Class Bank T1

Class Bank T2

Object Oriented Programming



What’s Next?

- Introduction to Functional and Object Oriented Programming
- Deep Dive into Functional Programming
- Deep Dive into Object Oriented Programming

Thank You

Analytics Vidhya

Summary: Functional and Object Oriented

Functional

- Emphasises on use of functions

Object Oriented

- Based on concept of objects

Summary: Functional and Object Oriented

Functional	Object Oriented
Emphasises on use of functions	Based on concept of objects
Statements can be executed in any order	Statements executed in particular order

Summary: Functional and Object Oriented

Functional	Object Oriented
Emphasises on use of functions	Based on concept of objects
Statements can be executed in any order	Statements executed in particular order
Can handle moderately complex program	Can handle very complex program

Summary: Functional and Object Oriented

Functional	Object Oriented
Emphasises on use of functions	Based on concept of objects
Statements can be executed in any order	Statements executed in particular order
Can handle moderately complex program	Can handle very complex programs
Less code flexibility	More flexible code

Summary: Functional and Object Oriented

Functional	Object Oriented
Emphasises on use of functions	Based on concept of objects
Statements can be executed in any order	Statements executed in particular order
Can handle moderately complex program	Can handle very complex programs
Less code flexibility	More flexible code
Lower code reusability	High code reusability

Procedural Programming

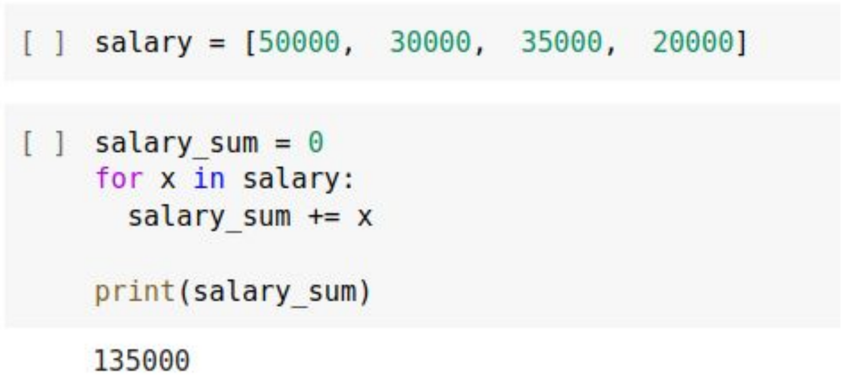
Example: Adding all values in the list ‘salary’ .

Salary = [50000, 30000, 35000, 20000]

Procedural Programming

Example: Adding all values in the list ‘salary’ .

Salary = [50000, 30000, 35000, 20000]



```
[ ] salary = [50000,  30000,  35000,  20000]
```

```
[ ] salary_sum = 0
    for x in salary:
      salary_sum += x

    print(salary_sum)
```

```
135000
```

Functional Programming

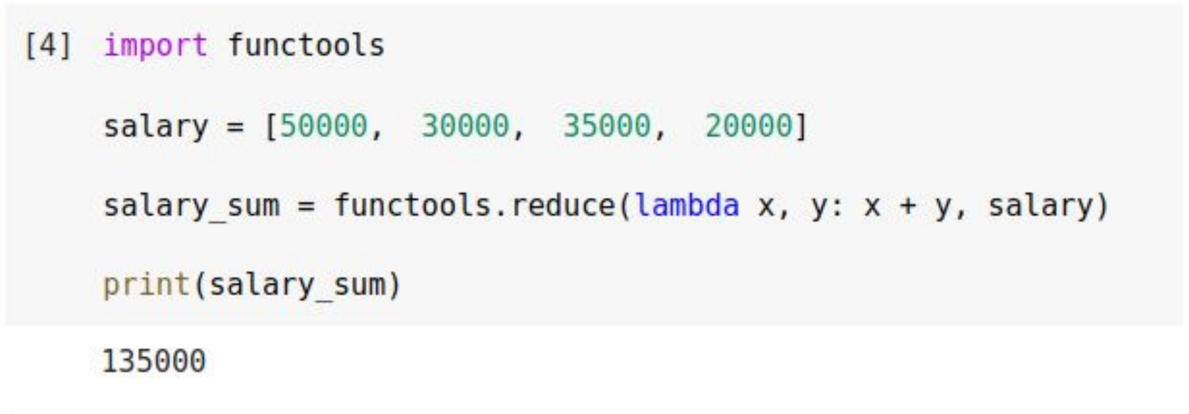
Example: Adding all values in the list ‘salary’ .

Salary = [50000, 30000, 35000, 20000]

Functional Programming

Example: Adding all values in the list 'salary' .

Salary = [50000, 30000, 35000, 20000]



```
[4] import functools

    salary = [50000,  30000,  35000,  20000]

    salary_sum = functools.reduce(lambda x, y: x + y, salary)

    print(salary_sum)

    135000
```

Object Oriented Programming

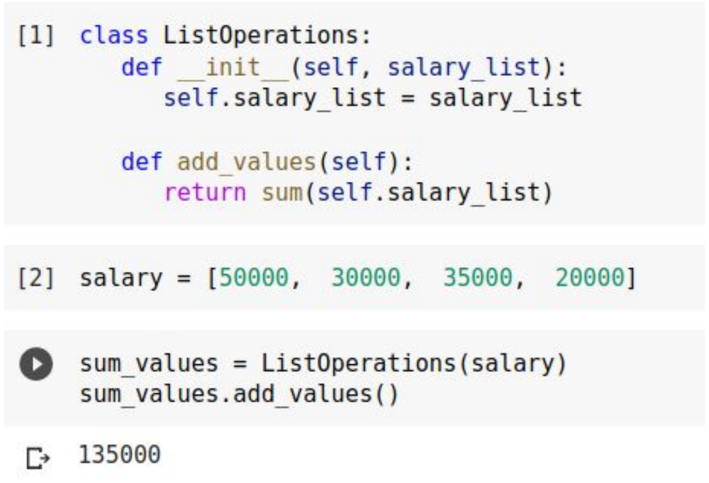
Example: Adding all values in the list ‘salary’ .

Salary = [50000, 30000, 35000, 20000]

Object Oriented Programming

Example: Adding all values in the list 'salary' .

Salary = [50000, 30000, 35000, 20000]



```
[1] class ListOperations:
        def __init__(self, salary_list):
            self.salary_list = salary_list

        def add_values(self):
            return sum(self.salary_list)
```

```
[2] salary = [50000, 30000, 35000, 20000]
```

```
sum_values = ListOperations(salary)
sum_values.add_values()
```

```
135000
```