Functional Programming Concepts

Harit Himanshu
@harittweets

Overview

- Immutability
- Expressions and Statements
- Functions
- Pure and Impure Functions
- Referential Transparency
- Higher-Order Functions

```
x = 5

y = 10


z = x + y // results in 15
```

Immutability in Mathematics

```
x = 5

y = 10

x = x + y
```

Immutability in Programming

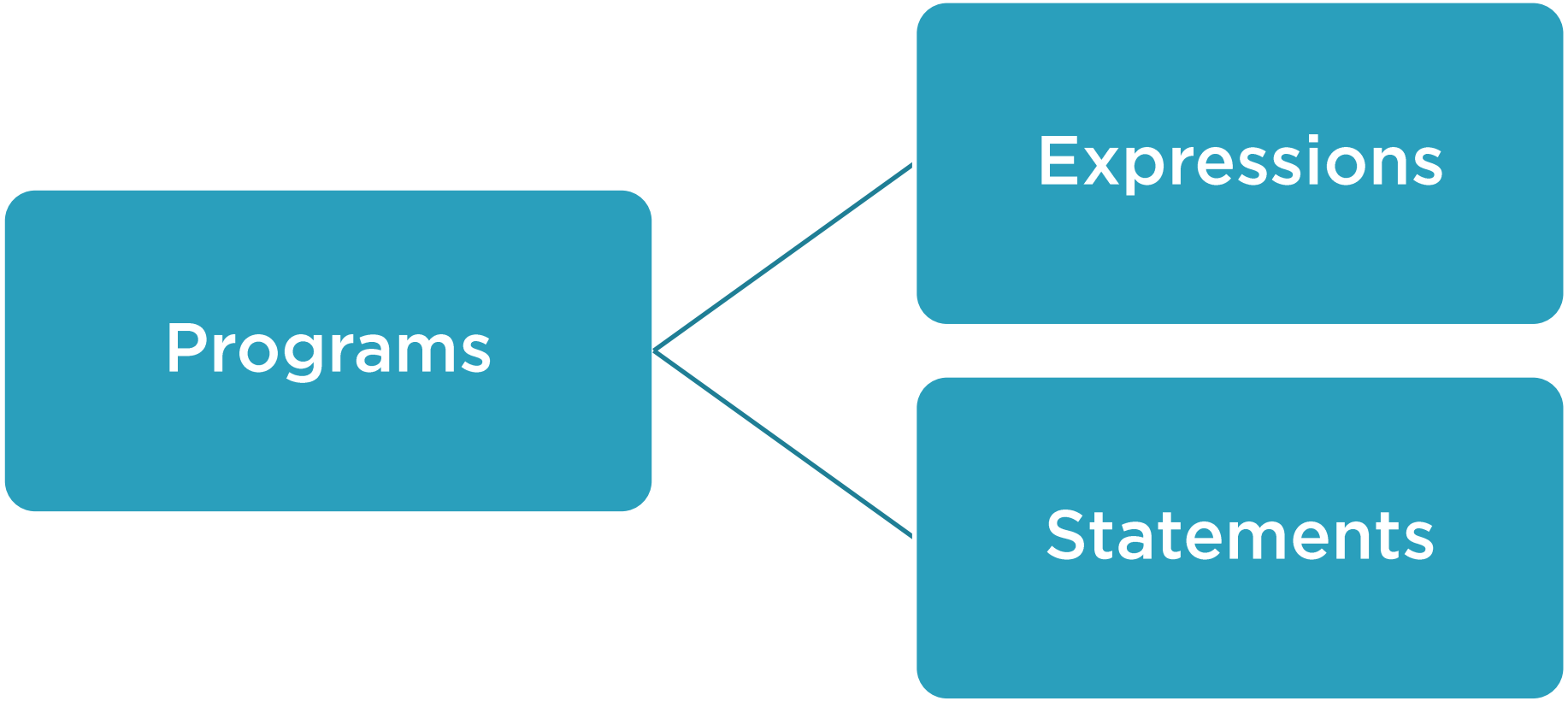
Why care about Immutability?

Easier to reason

No accidental changes

Parallelism

Expression and Statements



An expression is something that yields a value

```
x = 10

y = 20

x + y
```

- Yields 30 as result

```
getSquareOf(number)
```

- Yields the square of the given number

A statement is a line of
code that does some action

```
print("hello world")
```

◀ Prints the text on the console

```
order(pizzas)
```

◀ Submits an order

Expressions vs Statements

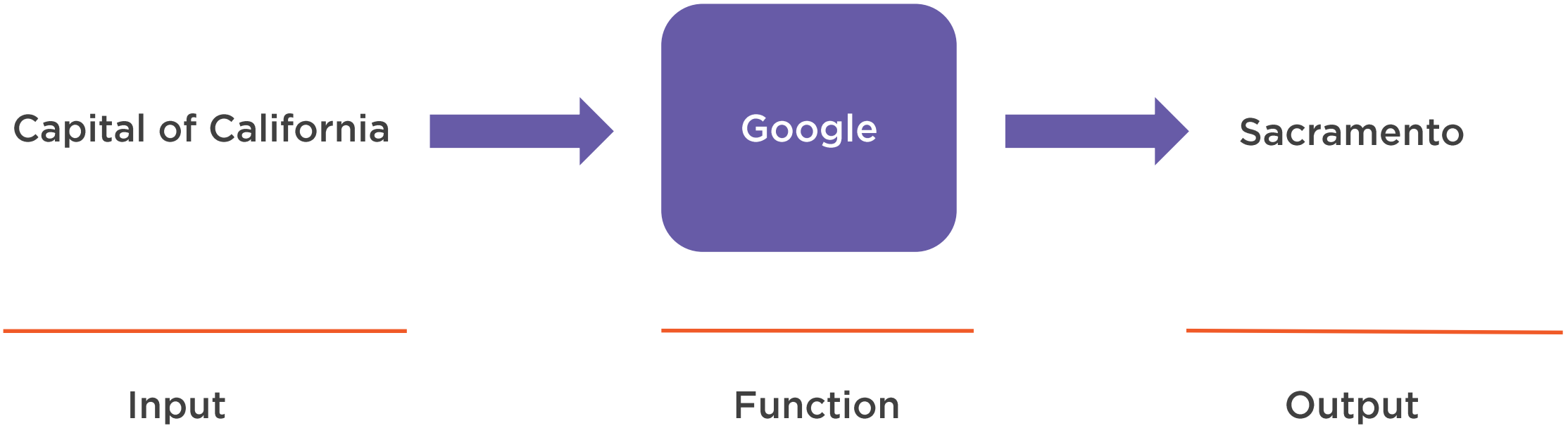
Expressions	Statements
No side effects	May contain side effects

Functions

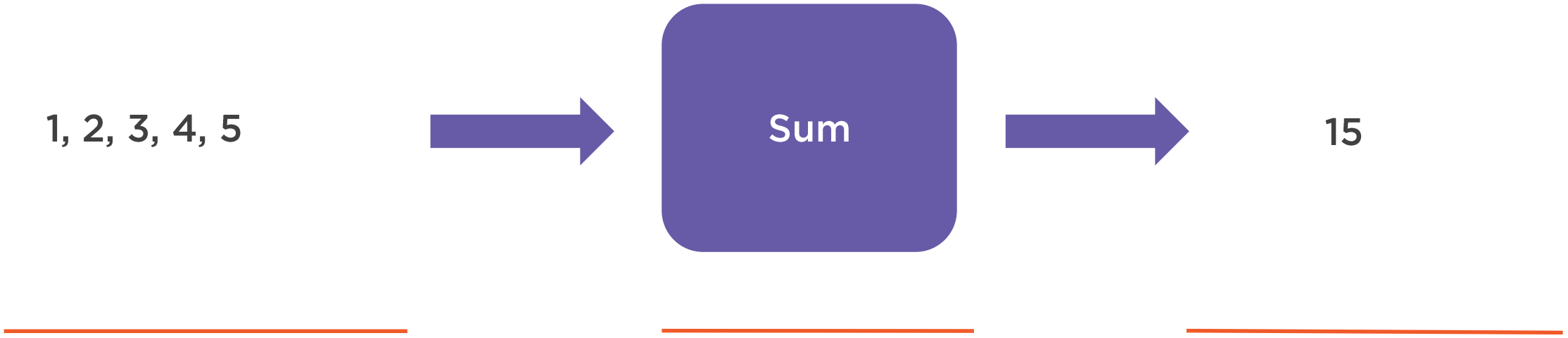
“is a relation between a set of inputs and a set of permissible outputs with the property that each input is related to exactly one output.”

Wikipedia

Function Example



Function Example



More Function Examples

Product

a * b

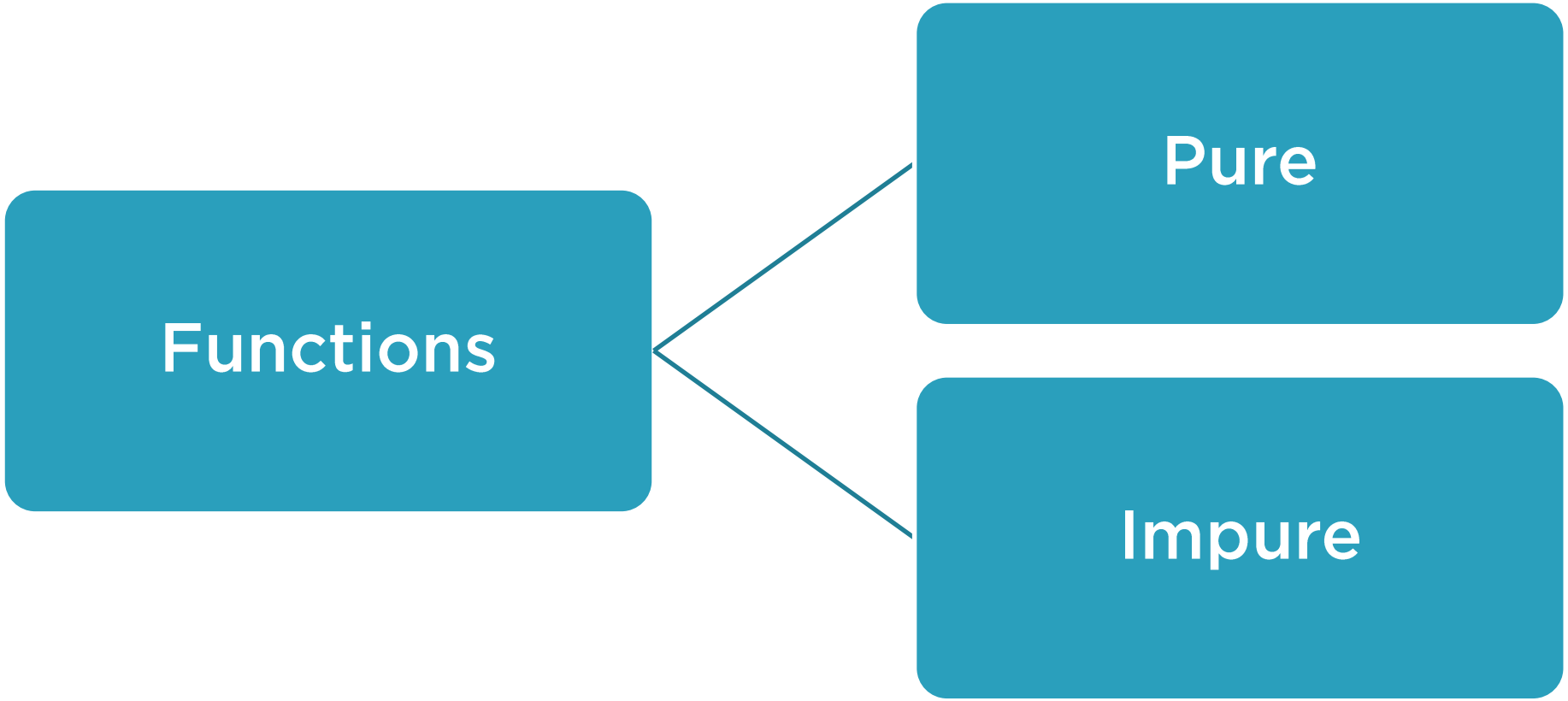
Square

a ^ 2

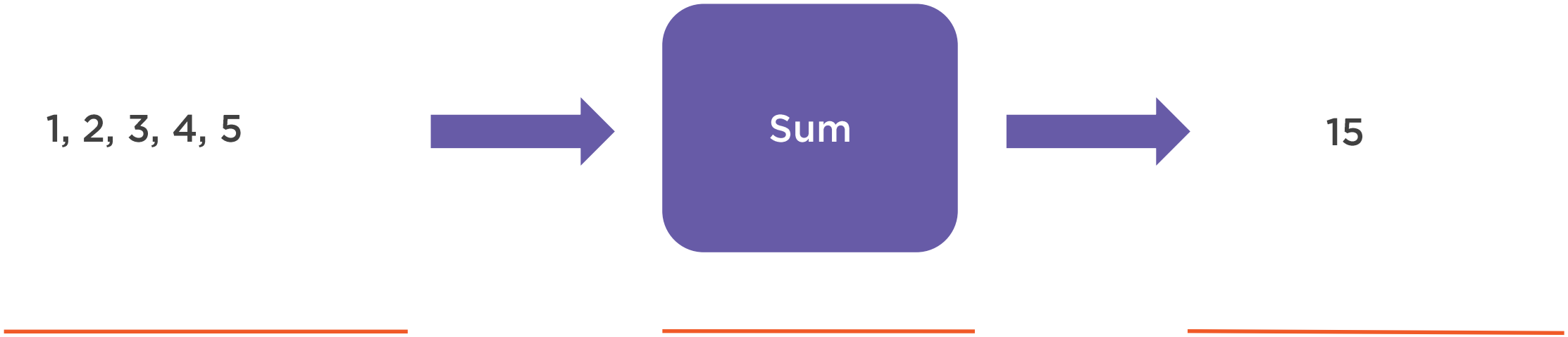
Division

a / b

Types of Functions



Pure Function Example



Pure functions evaluates to same result for a given set of inputs. Period!

Pure Function Examples

Product

a * b

Square

a ^ 2

Division

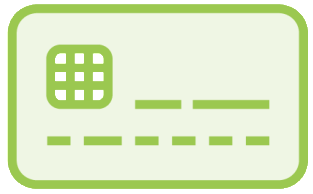
a / b

Impure Function Example

$100

Bank

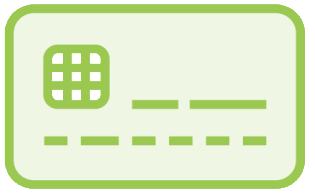
$40



debit(60)

$40

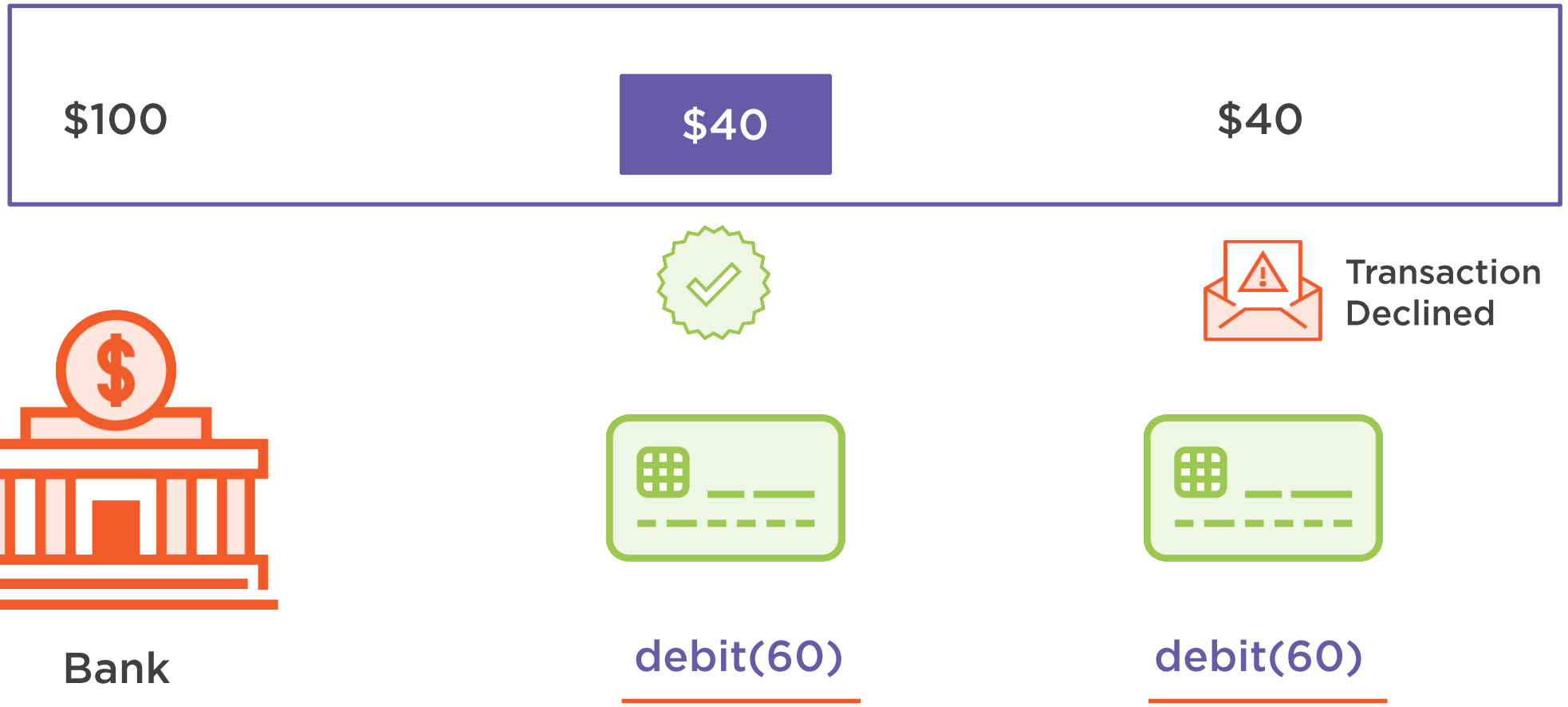
Transaction
Declined



debit(60)

Impure functions do not evaluate to the same output for the given set of inputs.

Impure Function Example



Pure vs Impure Functions

Pure Functions	Impure Functions
Evaluate the same result for the given set of input	Do not guarantee the same result for the given set of inputs
Do not cause the side effects	Cause the side effect in the system
Reliable	Not reliable

Why care about Pure Functions?

Easier to test

Reusable

Parallelize

Example of Side Effects

Making Network Calls

Updating the Database

Writing to File System

Functional programming puts restrictions on how we structure our programs and **not** on what programs we can express

Referential Transparency

Applies to

Expressions

Functions

Referential Transparency

An expression or function is called referentially transparent if it can be replaced with its value, without changing the algorithm, yielding the same output as when they were called without their value replacement.

add(a, b) = return a + b

multiply(a, b) = return a * b

x = add(2, multiply(3, 10))

x = 32

Step1: Replace multiply(3, 10) with 30

x = add(2, 30)

Step2: Replace add(2, 30) with it's value

x = 32

Violating Referential Transparency

Making Network Calls

Updating the Database

Writing to File System

Function Inputs

Numbers

1, 2, 3

Strings

“hello”, “maybe”

Booleans

true, false

Objects

{key: 1, value: “hello”}

Higher Order Functions

First Class Citizens

Functions

(a, b) => a * b

```
case class Fruit(name: String)
```

Higher Order Function Example

Representing a Fruit

```
val apple = Fruit("apple")

val orange = Fruit("orange")

val kiwi = Fruit("kiwi")
```

Higher Order Function Example

All the fruits

```
val fruitBasket = List(

apple, orange, kiwi, apple, orange, kiwi, apple, orange, kiwi

)
```

Higher Order Function Example

The Fruit Basket

```
def getApples(basket: List[Fruit]) = for (fruit <- basket if fruit.name == "apple") yield fruit

def getOranges(basket: List[Fruit]) = for (fruit <- basket if fruit.name == "orange") yield fruit
```

Higher Order Function Example

Get Specific Fruits

```
def getFruits(basket: List[Fruit], filterByFruit: Fruit => Boolean) = for (fruit <- basket if
filterByFruit(fruit)) yield fruit
```

Higher Order Function Example

A higher-order function to the rescue!

Get Fruits Using Higher-Order Function

```
def getApples(basket: List[Fruit]) = getFruits(fruitBasket, (fruit: Fruit)=>
fruit.name == "apple")

def getOranges(basket: List[Fruit])= getFruits(fruitBasket, (fruit: Fruit)=>
fruit.name == "orange")

def getKiwis(basket: List[Fruit])= getFruits(fruitBasket, (fruit: Fruit)=> fruit.name
== "kiwi")
```

Scala Higher Order Functions

map

reduce

filter

fold

Summary

- Immutability
- Expressions and Statements
- Functions
- Pure and Impure Functions
- Referential Transparency
- Higher-Order Functions