Abstraction

Pepe García

2020-04-20

Pure functions

In functional programming we say that functions have some special features.

They're values

Functions can be treated as any other value in your program. They can be assigned to `val`s, returned from methods, taken as parameters...

Purity

They're pure, meaning that they perform no side effects

Idempotence

given the same input, always return the same output

Side effects

- Throwing exceptions
- IO
- mutating variables
- Random

Referential transparency

They're referentially transparent. We can safely substitute any function call with its return value and it won't have any effect in the overall program.

Totality

They're total. Functions should operate on all values from it's input type. If they fail with an input, they're not total.

Totality

They're total. Functions should operate on all values from it's input type. If they fail with an input, they're not total.

When a function is not total, we say it's partial.

Examples

```scala
def sum(a: Int, b: Int): Int = a + b
```