Determinism: A pure function will always produce the same output for the same set of input parameters. It does not depend on any hidden state or external factors.

No side effects: A pure function does not modify any state outside its scope, such as global variables or mutable objects. It does not have any observable effect on the program or the environment.

Referential transparency: A pure function can be replaced with its result value without affecting the correctness of the program. This property allows for equational reasoning and facilitates code understanding and optimization.

By adhering to these principles, pure functions offer several advantages:

Predictability: Pure functions are easier to reason about because their behavior is entirely determined by their inputs. Given the same input, you can expect the same output every time.

Testability: Pure functions are straightforward to test since their behavior is isolated and does not depend on external factors. You can pass different inputs and verify the expected outputs without worrying about hidden state.

Concurrency: Because pure functions do not rely on shared mutable state, they are inherently thread-safe. They can be executed concurrently without concerns about race conditions or data corruption.

Modularity: Pure functions are self-contained units that can be easily composed and combined with other pure functions. This property promotes code modularity and reusability.