Module std::option

This module defines the Option type and its methods to represent and handle an optional value.

Abstraction of a value that may or may not be present. Implemented with a vector of size zero or one because Move bytecode does not have ADTs.

The Option is in an invalid state for the operation attempted. The Option is Some while it should be None.

The Option is in an invalid state for the operation attempted. The Option is None while it should be Some.

Return an empty Option

Return an Option containing e

Return true if t does not hold a value

Return true if t holds a value

Return true if the value in t is equal to e_ref Always returns false if t does not hold a value

Return an immutable reference to the value inside t Aborts if t does not hold a value

Return a reference to the value inside t if it holds one Return default_ref if t does not hold a value

Return the value inside t if it holds one Return default if t does not hold a value

Convert the none option t to a some option by adding e. Aborts if t already holds a value

Convert a some option to a none by removing and returning the value stored inside t Aborts if t does not hold a value

Return a mutable reference to the value inside t Aborts if t does not hold a value

Swap the old value inside t with e and return the old value Aborts if t does not hold a value

Swap the old value inside t with e and return the old value; or if there is no old value, fill it with e. Different from swap(), swap_or_fill() allows for t not holding a value.

Destroys t. If t holds a value, return it. Returns default otherwise

Unpack t and return its contents Aborts if t does not hold a value

Unpack t Aborts if t holds a value

Convert t into a vector of length 1 if it is Some, and an empty vector otherwise

Destroy Option and call the closure f on the value inside if it holds one.

Destroy Option and call the closure f on the value inside if it holds one.

Execute a closure on the value inside t if it holds one.

Execute a closure on the mutable reference to the value inside t if it holds one.

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

Return None if the value is None, otherwise return Option if the predicate f returns true.

Return false if the value is None, otherwise return the result of the predicate f.

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

Struct

Abstraction of a value that may or may not be present. Implemented with a vector of size zero or one because Move bytecode does not have ADTs.

```bash
```

The Option is in an invalid state for the operation attempted. The Option is Some while it should be None.

```bash
```

The Option is in an invalid state for the operation attempted. The Option is None while it should be Some.

```bash
```

Return an empty Option

```bash
```

```bash
```

Return an Option containing e

```bash
```

```bash
```

Return true if t does not hold a value

```bash
```

```bash
```

Return true if t holds a value

```bash
```

```bash
```

```
—

Return true if the value in t is equal to e_ref Always returns false if t does not hold a value

```bash

```
—

```bash

```
—

Return an immutable reference to the value inside t Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Return a reference to the value inside t if it holds one Return default_ref if t does not hold a value

```bash

```
—

```bash

```
—

Return the value inside t if it holds one Return default if t does not hold a value

```bash

```
—

```bash

```
—

Convert the none option t to a some option by adding e. Aborts if t already holds a value

```bash

```
—

```bash

```
—

Convert a some option to a none by removing and returning the value stored inside t Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Return a mutable reference to the value inside t Aborts if t does not hold a value

```bash

```
—
```

```bash
```

Swap the old value inside t with e and return the old value Aborts if t does not hold a value

```bash
```

```bash
```

Swap the old value inside t with e and return the old value; or if there is no old value, fill it with e. Different from swap(), swap_or_fill() allows for t not holding a value.

```bash
```

```bash
```

Destroys t. If t holds a value, return it. Returns default otherwise

```bash
```

```bash
```

Unpack t and return its contents Aborts if t does not hold a value

```bash
```

```bash
```

Unpack t Aborts if t holds a value

```bash
```

```bash
```

Convert t into a vector of length 1 if it is Some, and an empty vector otherwise

```bash
```

```bash
```

Destroy Option and call the closure f on the value inside if it holds one.

```bash
```
—
```bash
```
—
Destroy Option and call the closure f on the value inside if it holds one.
```bash
```
—
```bash
```
—
Execute a closure on the value inside t if it holds one.
```bash
```
—
```bash
```
—
Execute a closure on the mutable reference to the value inside t if it holds one.
```bash
```
—
```bash
```
—
Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).
```bash
```
—
```bash
```
—
If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).
```bash
```
—
```bash
```
—
If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).
```bash
```
—
```bash
```
—

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash
```

```bash
```

Return false if the value is None, otherwise return the result of the predicate f.

```bash
```

```bash
```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash
```

```bash
```

Constants

The Option is in an invalid state for the operation attempted. The Option is Some while it should be None.

```bash
```

The Option is in an invalid state for the operation attempted. The Option is None while it should be Some.

```bash
```

```

—

Return an empty Option

```bash

```

—

```bash

```

—

Return an Option containing e

```bash

```

—

```bash

```

—

Return true if t does not hold a value

```bash

```

—

```bash

```

—

Return true if t holds a value

```bash

```

—

```bash

```

—

Return true if the value in t is equal to e_ref Always returns false if t does not hold a value

```bash

```

—

```bash

```

—

Return an immutable reference to the value inside t Aborts if t does not hold a value

```bash

```

—

```bash

```

—

Return a reference to the value inside t if it holds one Return default_ref if t does not hold a value

```bash

```

—
```

```bash

```

Return the value inside t if it holds one Return default if t does not hold a value

```bash

```

```bash

```

Convert the none option t to a some option by adding e. Aborts if t already holds a value

```bash

```

```bash

```

Convert a some option to a none by removing and returning the value stored inside t Aborts if t does not hold a value

```bash

```

```bash

```

Return a mutable reference to the value inside t Aborts if t does not hold a value

```bash

```

```bash

```

Swap the old value inside t with e and return the old value Aborts if t does not hold a value

```bash

```

```bash

```

Swap the old value inside t with e and return the old value; or if there is no old value, fill it with e. Different from swap(), swap_or_fill() allows for t not holding a value.

```bash

```

```bash

```

Destroys t. If t holds a value, return it. Returns default otherwise

```bash
```
—
```bash
```
—
Unpack t and return its contents Aborts if t does not hold a value
```bash
```
—
```bash
```
—
Unpack t Aborts if t holds a value
```bash
```
—
```bash
```
—
Convert t into a vector of length 1 if it is Some, and an empty vector otherwise
```bash
```
—
```bash
```
—
Destroy Option and call the closure f on the value inside if it holds one.
```bash
```
—
```bash
```
—
Destroy Option and call the closure f on the value inside if it holds one.
```bash
```
—
```bash
```
—
Execute a closure on the value inside t if it holds one.
```bash
```
—
```bash
```
—

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash
```

```bash
```

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a.or(b).

```bash
```

```bash
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```

```bash
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```

```bash
```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t.map(f).

```bash
```

```bash
```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t.map(f).

```bash
```

```bash
```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash
```

```bash

```

Return false if the value is None, otherwise return the result of the predicate f.

```bash

```

```bash

```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash

```

```bash

```

Function

Return an empty Option

```bash

```

```bash

```

Return an Option containing e

```bash

```

```bash

```

Return true if t does not hold a value

```bash

```

```bash

```

Return true if t holds a value

```bash

```

```bash
```

```
—

Return true if the value in t is equal to e_ref Always returns false if t does not hold a value

```bash

```
—

```bash

```
—

Return an immutable reference to the value inside t Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Return a reference to the value inside t if it holds one Return default_ref if t does not hold a value

```bash

```
—

```bash

```
—

Return the value inside t if it holds one Return default if t does not hold a value

```bash

```
—

```bash

```
—

Convert the none option t to a some option by adding e. Aborts if t already holds a value

```bash

```
—

```bash

```
—

Convert a some option to a none by removing and returning the value stored inside t Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Return a mutable reference to the value inside t Aborts if t does not hold a value

```bash

```
—
```

```bash

```

Swap the old value inside t with e and return the old value Aborts if t does not hold a value

```bash

```

```bash

```

Swap the old value inside t with e and return the old value; or if there is no old value, fill it with e. Different from swap(), swap_or_fill() allows for t not holding a value.

```bash

```

```bash

```

Destroys t. If t holds a value, return it. Returns default otherwise

```bash

```

```bash

```

Unpack t and return its contents Aborts if t does not hold a value

```bash

```

```bash

```

Unpack t Aborts if t holds a value

```bash

```

```bash

```

Convert t into a vector of length 1 if it is Some, and an empty vector otherwise

```bash

```

```bash

```

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```
—

```bash

```
—

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```
—

```bash

```
—

Execute a closure on the value inside t if it holds one.

```bash

```
—

```bash

```
—

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash

```
—

```bash

```
—

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash

```
—

```bash

```
—

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```
—

```bash

```
—

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```
—

```bash

```
—

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash
```

```bash
```

Return false if the value is None, otherwise return the result of the predicate f.

```bash
```

```bash
```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash
```

```bash
```

Function

Return an Option containing e

```bash
```

```bash
```

Return true if t does not hold a value

```bash
```

```bash
```

Return true if t holds a value

```bash
```

```bash
```

Return true if the value in t is equal to e_ref Always returns false if t does not hold a value

```bash
```

```bash
```

Return an immutable reference to the value inside t Aborts if t does not hold a value

```bash
```

```bash
```

Return a reference to the value inside t if it holds one Return default_ref if t does not hold a value

```bash
```

```bash
```

Return the value inside t if it holds one Return default if t does not hold a value

```bash
```

```bash
```

Convert the none option t to a some option by adding e. Aborts if t already holds a value

```bash
```

```bash
```

```
—

Convert a some option to a none by removing and returning the value stored inside t Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Return a mutable reference to the value inside t Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Swap the old value inside t with e and return the old value Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Swap the old value inside t with e and return the old value; or if there is no old value, fill it with e. Different from swap(), swap_or_fill() allows for t not holding a value.

```bash

```
—

```bash

```
—

Destroys t. If t holds a value, return it. Returns default otherwise

```bash

```
—

```bash

```
—

Unpack t and return its contents Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Unpack t Aborts if t holds a value

```bash

```

```bash

```

Convert t into a vector of length 1 if it is Some, and an empty vector otherwise

```bash

```

```bash

```

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```

```bash

```

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```

```bash

```

Execute a closure on the value inside t if it holds one.

```bash

```

```bash

```

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash

```

```bash

```

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash

```

```bash

```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).
```

```bash
```
—
```bash
```
—
If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).
```bash
```
—
```bash
```
—
Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).
```bash
```
—
```bash
```
—
Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).
```bash
```
—
```bash
```
—
Return None if the value is None, otherwise return Option if the predicate f returns true.
```bash
```
—
```bash
```
—
Return false if the value is None, otherwise return the result of the predicate f.
```bash
```
—
```bash
```
—
Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.
```bash
```
—

```bash

```

Function

Return true if t does not hold a value

```bash

```

```bash

```

Return true if t holds a value

```bash

```

```bash

```

Return true if the value in t is equal to e_ref Always returns false if t does not hold a value

```bash

```

```bash

```

Return an immutable reference to the value inside t Aborts if t does not hold a value

```bash

```

```bash

```

Return a reference to the value inside t if it holds one Return default_ref if t does not hold a value

```bash

```

```bash

```

Return the value inside t if it holds one Return default if t does not hold a value

```bash

```

```bash

```

Convert the none option t to a some option by adding e. Aborts if t already holds a value

```bash

```
—

```bash

```
—

Convert a some option to a none by removing and returning the value stored inside t Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Return a mutable reference to the value inside t Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Swap the old value inside t with e and return the old value Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Swap the old value inside t with e and return the old value; or if there is no old value, fill it with e. Different from swap(), swap_or_fill() allows for t not holding a value.

```bash

```
—

```bash

```
—

Destroys t. If t holds a value, return it. Returns default otherwise

```bash

```
—

```bash

```
—

Unpack t and return its contents Aborts if t does not hold a value

```bash

```
—

```bash
```

```

—

Unpack t Aborts if t holds a value

```bash

```

—

```bash

```

—

Convert t into a vector of length 1 if it is Some, and an empty vector otherwise

```bash

```

—

```bash

```

—

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```

—

```bash

```

—

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```

—

```bash

```

—

Execute a closure on the value inside t if it holds one.

```bash

```

—

```bash

```

—

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash

```

—

```bash

```

—

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash

```

—
```

```bash
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```

```bash
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```

```bash
```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash
```

```bash
```

Return false if the value is None, otherwise return the result of the predicate f.

```bash
```

```bash
```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash
```

```bash
```

Function

Return true if t holds a value

```bash
```

```bash
```

Return true if the value in t is equal to e_ref Always returns false if t does not hold a value

```bash
```

```bash
```

Return an immutable reference to the value inside t Aborts if t does not hold a value

```bash
```

```bash
```

Return a reference to the value inside t if it holds one Return default_ref if t does not hold a value

```bash
```

```bash
```

Return the value inside t if it holds one Return default if t does not hold a value

```bash
```

```bash
```

Convert the none option t to a some option by adding e. Aborts if t already holds a value

```bash
```

```
—
```bash
```
—
Convert a some option to a none by removing and returning the value stored inside t Aborts if t does not hold a value
```bash
```
—
```bash
```
—
Return a mutable reference to the value inside t Aborts if t does not hold a value
```bash
```
—
```bash
```
—
Swap the old value inside t with e and return the old value Aborts if t does not hold a value
```bash
```
—
```bash
```
—
Swap the old value inside t with e and return the old value; or if there is no old value, fill it with e. Different from swap(), swap_or_fill() allows for t not holding a value.
```bash
```
—
```bash
```
—
Destroys t. If t holds a value, return it. Returns default otherwise
```bash
```
—
```bash
```
—
Unpack t and return its contents Aborts if t does not hold a value
```bash
```
—
```bash
```
—
```

Unpack t Aborts if t holds a value

```bash
```

```bash
```

Convert t into a vector of length 1 if it is Some, and an empty vector otherwise

```bash
```

```bash
```

Destroy Option and call the closure f on the value inside if it holds one.

```bash
```

```bash
```

Destroy Option and call the closure f on the value inside if it holds one.

```bash
```

```bash
```

Execute a closure on the value inside t if it holds one.

```bash
```

```bash
```

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash
```

```bash
```

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash
```

```bash
```

```
—
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```
—
```

```bash
```
—
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```
—
```

```bash
```
—
```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash
```
—
```

```bash
```
—
```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash
```
—
```

```bash
```
—
```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash
```
—
```

```bash
```
—
```

Return false if the value is None, otherwise return the result of the predicate f.

```bash
```
—
```

```bash
```
—
```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash
```
```bash
```
Function

Return true if the value in t is equal to e_ref Always returns false if t does not hold a value

```bash
```
```bash
```
Return an immutable reference to the value inside t Aborts if t does not hold a value

```bash
```
```bash
```
Return a reference to the value inside t if it holds one Return default_ref if t does not hold a value

```bash
```
```bash
```
Return the value inside t if it holds one Return default if t does not hold a value

```bash
```
```bash
```
Convert the none option t to a some option by adding e. Aborts if t already holds a value

```bash
```
```bash
```
Convert a some option to a none by removing and returning the value stored inside t Aborts if t does not hold a value

```bash
```
```bash
```

```
—

Return a mutable reference to the value inside t Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Swap the old value inside t with e and return the old value Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Swap the old value inside t with e and return the old value; or if there is no old value, fill it with e. Different from swap(), swap_or_fill() allows for t not holding a value.

```bash

```
—

```bash

```
—

Destroys t. If t holds a value, return it. Returns default otherwise

```bash

```
—

```bash

```
—

Unpack t and return its contents Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Unpack t Aborts if t holds a value

```bash

```
—

```bash

```
—

Convert t into a vector of length 1 if it is Some, and an empty vector otherwise

```bash

```

```bash

```

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```

```bash

```

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```

```bash

```

Execute a closure on the value inside t if it holds one.

```bash

```

```bash

```

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash

```

```bash

```

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash

```

```bash

```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```

```bash

```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).
```

```bash

```

```bash

```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash

```

```bash

```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash

```

```bash

```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash

```

```bash

```

Return false if the value is None, otherwise return the result of the predicate f.

```bash

```

```bash

```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash

```

```bash

```

Function

Return an immutable reference to the value inside t Aborts if t does not hold a value

```bash
```
—
```bash
```
—
Return a reference to the value inside t if it holds one Return default_ref if t does not hold a value
```bash
```
—
```bash
```
—
Return the value inside t if it holds one Return default if t does not hold a value
```bash
```
—
```bash
```
—
Convert the none option t to a some option by adding e. Aborts if t already holds a value
```bash
```
—
```bash
```
—
Convert a some option to a none by removing and returning the value stored inside t Aborts if t does not hold a value
```bash
```
—
```bash
```
—
Return a mutable reference to the value inside t Aborts if t does not hold a value
```bash
```
—
```bash
```
—
Swap the old value inside t with e and return the old value Aborts if t does not hold a value
```bash
```
—
```bash
```
—

Swap the old value inside t with e and return the old value; or if there is no old value, fill it with e. Different from swap(), swap_or_fill() allows for t not holding a value.

```bash

```

```bash

```

Destroys t. If t holds a value, return it. Returns default otherwise

```bash

```

```bash

```

Unpack t and return its contents Aborts if t does not hold a value

```bash

```

```bash

```

Unpack t Aborts if t holds a value

```bash

```

```bash

```

Convert t into a vector of length 1 if it is Some, and an empty vector otherwise

```bash

```

```bash

```

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```

```bash

```

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```

```bash
```

Execute a closure on the value inside t if it holds one.

```bash
```

```bash
```

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash
```

```bash
```

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash
```

```bash
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```

```bash
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```

```bash
```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash
```
—
```bash
```
—
Return None if the value is None, otherwise return Option if the predicate f returns true.
```bash
```
—
```bash
```
—
Return false if the value is None, otherwise return the result of the predicate f.
```bash
```
—
```bash
```
—
Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.
```bash
```
—
```bash
```
—
Function
Return a reference to the value inside t if it holds one Return default_ref if t does not hold a value
```bash
```
—
```bash
```
—
Return the value inside t if it holds one Return default if t does not hold a value
```bash
```
—
```bash
```
—
Convert the none option t to a some option by adding e. Aborts if t already holds a value
```bash
```

```
—
```bash
```
—
Convert a some option to a none by removing and returning the value stored inside t Aborts if t does not hold a value
```bash
```
—
```bash
```
—
Return a mutable reference to the value inside t Aborts if t does not hold a value
```bash
```
—
```bash
```
—
Swap the old value inside t with e and return the old value Aborts if t does not hold a value
```bash
```
—
```bash
```
—
Swap the old value inside t with e and return the old value; or if there is no old value, fill it with e. Different from swap(), swap_or_fill() allows for t not holding a value.
```bash
```
—
```bash
```
—
Destroys t. If t holds a value, return it. Returns default otherwise
```bash
```
—
```bash
```
—
Unpack t and return its contents Aborts if t does not hold a value
```bash
```
—
```bash
```
—
```

Unpack t Aborts if t holds a value

```bash
```

```bash
```

Convert t into a vector of length 1 if it is Some, and an empty vector otherwise

```bash
```

```bash
```

Destroy Option and call the closure f on the value inside if it holds one.

```bash
```

```bash
```

Destroy Option and call the closure f on the value inside if it holds one.

```bash
```

```bash
```

Execute a closure on the value inside t if it holds one.

```bash
```

```bash
```

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash
```

```bash
```

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash
```

```bash
```

```
—
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```
—
```

```bash
```
—
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```
—
```

```bash
```
—
```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash
```
—
```

```bash
```
—
```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash
```
—
```

```bash
```
—
```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash
```
—
```

```bash
```
—
```

Return false if the value is None, otherwise return the result of the predicate f.

```bash
```
—
```

```bash
```
—
```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash

```
—

```bash

```
—

Function

Return the value inside t if it holds one Return default if t does not hold a value

```bash

```
—

```bash

```
—

Convert the none option t to a some option by adding e. Aborts if t already holds a value

```bash

```
—

```bash

```
—

Convert a some option to a none by removing and returning the value stored inside t Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Return a mutable reference to the value inside t Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Swap the old value inside t with e and return the old value Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Swap the old value inside t with e and return the old value; or if there is no old value, fill it with e. Different from swap(), swap_or_fill() allows for t not holding a value.

```bash

```
—

```bash
```
Destroys t. If t holds a value, return it. Returns default otherwise
```bash
```
```bash
```
Unpack t and return its contents Aborts if t does not hold a value
```bash
```
```bash
```
Unpack t Aborts if t holds a value
```bash
```
```bash
```
Convert t into a vector of length 1 if it is Some, and an empty vector otherwise
```bash
```
```bash
```
Destroy Option and call the closure f on the value inside if it holds one.
```bash
```
```bash
```
Destroy Option and call the closure f on the value inside if it holds one.
```bash
```
```bash
```
Execute a closure on the value inside t if it holds one.
```bash
```

```

```bash

```

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash

```

```bash

```

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash

```

```bash

```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```

```bash

```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```

```bash

```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash

```

```bash

```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash

```

```bash

```
```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash
```

```bash
```

Return false if the value is None, otherwise return the result of the predicate f.

```bash
```

```bash
```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash
```

```bash
```

Function

Convert the none option t to a some option by adding e. Aborts if t already holds a value

```bash
```

```bash
```

Convert a some option to a none by removing and returning the value stored inside t Aborts if t does not hold a value

```bash
```

```bash
```

Return a mutable reference to the value inside t Aborts if t does not hold a value

```bash
```

```bash
```

Swap the old value inside t with e and return the old value Aborts if t does not hold a value

```bash
```
—
```bash
```
—
Swap the old value inside t with e and return the old value; or if there is no old value, fill it with e. Different from swap(), swap_or_fill() allows for t not holding a value.
```bash
```
—
```bash
```
—
Destroys t. If t holds a value, return it. Returns default otherwise
```bash
```
—
```bash
```
—
Unpack t and return its contents Aborts if t does not hold a value
```bash
```
—
```bash
```
—
Unpack t Aborts if t holds a value
```bash
```
—
```bash
```
—
Convert t into a vector of length 1 if it is Some, and an empty vector otherwise
```bash
```
—
```bash
```
—
Destroy Option and call the closure f on the value inside if it holds one.
```bash
```
—
```bash
```

```
—
```

Destroy Option and call the closure f on the value inside if it holds one.

```bash
```
—
```

```bash
```
—
```

Execute a closure on the value inside t if it holds one.

```bash
```
—
```

```bash
```
—
```

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash
```
—
```

```bash
```
—
```

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash
```
—
```

```bash
```
—
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```
—
```

```bash
```
—
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```
—
```

```bash
```
—
```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash
```
—
```
```

```bash
```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash
```

```bash
```

Return false if the value is None, otherwise return the result of the predicate f.

```bash
```

```bash
```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash
```

```bash
```

Function

Convert a some option to a none by removing and returning the value stored inside t Aborts if t does not hold a value

```bash
```

```bash
```

Return a mutable reference to the value inside t Aborts if t does not hold a value

```bash
```

```bash
```

Swap the old value inside t with e and return the old value Aborts if t does not hold a value

```bash
```

```bash
```

Swap the old value inside t with e and return the old value; or if there is no old value, fill it with e. Different from swap(), swap_or_fill() allows for t not holding a value.

```bash
```

```bash
```

Destroys t. If t holds a value, return it. Returns default otherwise

```bash
```

```bash
```

Unpack t and return its contents Aborts if t does not hold a value

```bash
```

```bash
```

Unpack t Aborts if t holds a value

```bash
```

```bash
```

Convert t into a vector of length 1 if it is Some, and an empty vector otherwise

```bash
```

```bash
```

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```
—

```bash

```
—

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```
—

```bash

```
—

Execute a closure on the value inside t if it holds one.

```bash

```
—

```bash

```
—

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash

```
—

```bash

```
—

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash

```
—

```bash

```
—

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```
—

```bash

```
—

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```
—

```bash

```
—

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash
```

```bash
```

Return false if the value is None, otherwise return the result of the predicate f.

```bash
```

```bash
```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash
```

```bash
```

Function

Return a mutable reference to the value inside t Aborts if t does not hold a value

```bash
```

```bash
```

Swap the old value inside t with e and return the old value Aborts if t does not hold a value

```bash
```

```bash
```

Swap the old value inside t with e and return the old value; or if there is no old value, fill it with e. Different from swap(), swap_or_fill() allows for t not holding a value.

```bash
```

```bash
```

Destroys t. If t holds a value, return it. Returns default otherwise

```bash
```

```bash
```

Unpack t and return its contents Aborts if t does not hold a value

```bash
```

```bash
```

Unpack t Aborts if t holds a value

```bash
```

```bash
```

Convert t into a vector of length 1 if it is Some, and an empty vector otherwise

```bash
```

```bash
```

Destroy Option and call the closure f on the value inside if it holds one.

```bash
```

```bash
```
—

Destroy Option and call the closure f on the value inside if it holds one.

```bash
```
—

```bash
```
—

Execute a closure on the value inside t if it holds one.

```bash
```
—

```bash
```
—

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash
```
—

```bash
```
—

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash
```
—

```bash
```
—

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```
—

```bash
```
—

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```
—

```bash
```
—

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash
```

```

```bash

```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash

```

```bash

```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash

```

```bash

```

Return false if the value is None, otherwise return the result of the predicate f.

```bash

```

```bash

```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash

```

```bash

```

Function

Swap the old value inside t with e and return the old value Aborts if t does not hold a value

```bash

```

```bash

```

Swap the old value inside t with e and return the old value; or if there is no old value, fill it with e. Different from swap(), swap_or_fill() allows for t not holding a value.

```bash

```

```bash

```

Destroys t. If t holds a value, return it. Returns default otherwise

```bash

```

```bash

```

Unpack t and return its contents Aborts if t does not hold a value

```bash

```

```bash

```

Unpack t Aborts if t holds a value

```bash

```

```bash

```

Convert t into a vector of length 1 if it is Some, and an empty vector otherwise

```bash

```

```bash

```

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```

```bash

```

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```

```bash

```

Execute a closure on the value inside t if it holds one.
```

```bash

```
—

```bash

```
—

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash

```
—

```bash

```
—

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash

```
—

```bash

```
—

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```
—

```bash

```
—

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```
—

```bash

```
—

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash

```
—

```bash

```
—

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash

```
—

```bash
```

```
—

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash

```
—

```bash

```
—

Return false if the value is None, otherwise return the result of the predicate f.

```bash

```
—

```bash

```
—

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash

```
—

```bash

```
—

Function

Swap the old value inside t with e and return the old value; or if there is no old value, fill it with e. Different from swap(), swap_or_fill() allows for t not holding a value.

```bash

```
—

```bash

```
—

Destroys t. If t holds a value, return it. Returns default otherwise

```bash

```
—

```bash

```
—

Unpack t and return its contents Aborts if t does not hold a value

```bash

```
—

```bash

```
—

Unpack t Aborts if t holds a value

```bash

```
—

```bash

```
—

Convert t into a vector of length 1 if it is Some, and an empty vector otherwise

```bash

```
—

```bash

```
—

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```
—

```bash

```
—

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```
—

```bash

```
—

Execute a closure on the value inside t if it holds one.

```bash

```
—

```bash

```
—

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash

```
—

```bash

```
—

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash

```
—
```

```bash
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```

```bash
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```

```bash
```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash
```

```bash
```

Return false if the value is None, otherwise return the result of the predicate f.

```bash
```

```bash
```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash
```

```bash
```

Function

Destroys t. If t holds a value, return it. Returns default otherwise

```bash
```

```bash
```

Unpack t and return its contents Aborts if t does not hold a value

```bash
```

```bash
```

Unpack t Aborts if t holds a value

```bash
```

```bash
```

Convert t into a vector of length 1 if it is Some, and an empty vector otherwise

```bash
```

```bash
```

Destroy Option and call the closure f on the value inside if it holds one.

```bash
```

```bash
```

Destroy Option and call the closure f on the value inside if it holds one.

```bash
```

```

```bash

```

Execute a closure on the value inside t if it holds one.

```bash

```

```bash

```

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash

```

```bash

```

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash

```

```bash

```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```

```bash

```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```

```bash

```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash

```

```bash

```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to
```

Rust's t. map (f).

```bash
```

```bash
```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash
```

```bash
```

Return false if the value is None, otherwise return the result of the predicate f.

```bash
```

```bash
```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash
```

```bash
```

Function

Unpack t and return its contents Aborts if t does not hold a value

```bash
```

```bash
```

Unpack t Aborts if t holds a value

```bash
```

```bash
```

Convert t into a vector of length 1 if it is Some, and an empty vector otherwise

```bash

```
—

```bash

```
—

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```
—

```bash

```
—

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```
—

```bash

```
—

Execute a closure on the value inside t if it holds one.

```bash

```
—

```bash

```
—

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash

```
—

```bash

```
—

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash

```
—

```bash

```
—

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```
—

```bash

```
—

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```

```bash
```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash
```

```bash
```

Return false if the value is None, otherwise return the result of the predicate f.

```bash
```

```bash
```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash
```

```bash
```

Function

Unpack t Aborts if t holds a value

```bash
```

```bash
```

Convert t into a vector of length 1 if it is Some, and an empty vector otherwise

```bash
```

```bash
```

Destroy Option and call the closure f on the value inside if it holds one.

```bash
```

```bash
```

Destroy Option and call the closure f on the value inside if it holds one.

```bash
```

```bash
```

Execute a closure on the value inside t if it holds one.

```bash
```

```bash
```

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash
```

```bash
```

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash
```

```bash
```

```
—
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```
—
```

```bash
```
—
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```
—
```

```bash
```
—
```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash
```
—
```

```bash
```
—
```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash
```
—
```

```bash
```
—
```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash
```
—
```

```bash
```
—
```

Return false if the value is None, otherwise return the result of the predicate f.

```bash
```
—
```

```bash
```
—
```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash
```
—
```bash
```
—

Function

Convert t into a vector of length 1 if it is Some, and an empty vector otherwise

```bash
```
—
```bash
```
—

Destroy Option and call the closure f on the value inside if it holds one.

```bash
```
—
```bash
```
—

Destroy Option and call the closure f on the value inside if it holds one.

```bash
```
—
```bash
```
—

Execute a closure on the value inside t if it holds one.

```bash
```
—
```bash
```
—

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash
```
—
```bash
```
—

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash
```
—
```bash
```

```
—
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```
—
```

```bash
```
—
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```
—
```

```bash
```
—
```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash
```
—
```

```bash
```
—
```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash
```
—
```

```bash
```
—
```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash
```
—
```

```bash
```
—
```

Return false if the value is None, otherwise return the result of the predicate f.

```bash
```
—
```

```bash
```
—
```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash

```
—

```bash

```
—

Macro function

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```
—

```bash

```
—

Destroy Option and call the closure f on the value inside if it holds one.

```bash

```
—

```bash

```
—

Execute a closure on the value inside t if it holds one.

```bash

```
—

```bash

```
—

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash

```
—

```bash

```
—

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash

```
—

```bash

```
—

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```
—

```bash
```

```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```

```bash

```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash

```

```bash

```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash

```

```bash

```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash

```

```bash

```

Return false if the value is None, otherwise return the result of the predicate f.

```bash

```

```bash

```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash

```

```bash

```

Macro function
```

Destroy Option and call the closure f on the value inside if it holds one.

```bash
```

```bash
```

Execute a closure on the value inside t if it holds one.

```bash
```

```bash
```

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash
```

```bash
```

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash
```

```bash
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```

```bash
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```

```bash
```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

```
—

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash

```
—

```bash

```
—

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash

```
—

```bash

```
—

Return false if the value is None, otherwise return the result of the predicate f.

```bash

```
—

```bash

```
—

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash

```
—

```bash

```
—

Macro function

Execute a closure on the value inside t if it holds one.

```bash

```
—

```bash

```
—

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash

```
—

```bash

```

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash

```

```bash

```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```

```bash

```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```

```bash

```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash

```

```bash

```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash

```

```bash

```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash

```

```bash

```

Return false if the value is None, otherwise return the result of the predicate f.

```bash

```

```bash

```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash

```

```bash

```

Macro function

Execute a closure on the mutable reference to the value inside t if it holds one.

```bash

```

```bash

```

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash

```

```bash

```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```

```bash

```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```

```bash

```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash

```
```

```bash

```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash

```

```bash

```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash

```

```bash

```

Return false if the value is None, otherwise return the result of the predicate f.

```bash

```

```bash

```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash

```

```bash

```

Macro function

Select the first Some value from the two options, or None if both are None. Equivalent to Rust's a. or (b).

```bash

```

```bash

```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```

```bash
```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash
```

```bash
```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash
```

```bash
```

Return false if the value is None, otherwise return the result of the predicate f.

```bash
```

```bash
```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash
```

```bash
```

Macro function

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```

```bash

```

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```

```bash

```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash

```

```bash

```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash

```

```bash

```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash

```

```bash

```

Return false if the value is None, otherwise return the result of the predicate f.

```bash

```

```bash

```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash

```

```bash

```

Macro function

If the value is Some, call the closure f on it. Otherwise, return None. Equivalent to Rust's t.and_then(f).

```bash

```

```bash

```

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash

```

```bash

```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash

```

```bash

```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash

```

```bash

```

Return false if the value is None, otherwise return the result of the predicate f.

```bash

```

```bash

```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash

```

```bash

```

Macro function

Map an Option to Option by applying a function to a contained value. Equivalent to Rust's t. map (f).

```bash

```

```bash

```

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash

```

```bash

```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash

```

```bash

```

Return false if the value is None, otherwise return the result of the predicate f.

```bash

```

```bash

```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash

```

```bash

```

Macro function

Map an Option value to Option by applying a function to a contained value by reference. Original Option is preserved. Equivalent to Rust's t. map (f).

```bash
```

```bash
```

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash
```

```bash
```

Return false if the value is None, otherwise return the result of the predicate f.

```bash
```

```bash
```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash
```

```bash
```

Macro function

Return None if the value is None, otherwise return Option if the predicate f returns true.

```bash
```

```bash
```

Return false if the value is None, otherwise return the result of the predicate f.

```bash
```

```bash
```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash

```

```bash

```

Macro function

Return false if the value is None, otherwise return the result of the predicate f.

```bash

```

```bash

```

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash

```

```bash

```

Macro function

Destroy Option and return the value inside if it holds one, or default otherwise. Equivalent to Rust's t.unwrap_or(default).

Note: this function is a more efficient version of destroy_with_default , as it does not evaluate the default value unless necessary. The destroy_with_default function should be deprecated in favor of this function.

```bash

```

```bash

```