Mojo traits

What each standard library trait does, when you need it, and where it bites

Mojo language reference
v1.0.0b2

keyword type built-in "string" number operator @decorator True/False/None # comment

LIFECYCLE

AnyType the root; every type conforms automatically — `builtin/anytype.mojo`

no requirements

ImplicitlyDeletable has an implicit destructor, called at last use — `builtin/anytype.mojo`

`__del__(deinit self, /)`	provided
`comptime __del__is_trivial: Bool`	compile-time flag

Automatically added to every eligible type (all fields are also **ImplicitlyDeletable**). When the type is trivial, Mojo skips the destructor.

Movable transfer ownership; enables the ^ operator — `builtin/value.mojo`

`__init__(out self, *, deinit move: Self)`	provided
`comptime __move_ctor_is_trivial: Bool`	compile-time flag

Required to store a type in **List**, **Optional**, or **Variant**, or to return it by move.

Copyable explicit copy — `builtin/value.mojo`

Refines: Movable

`__init__(out self, *, copy: Self)`	provided
`copy(self) -> Self`	provided
`comptime __copy_ctor_is_trivial: Bool`	compile-time flag

The copy constructor is synthesized if all fields are **Copyable**.

ImplicitlyCopyable (MARKER) lets copies be inserted implicitly — `builtin/value.mojo`

Refines: Copyable < Movable · no new requirements

Can mask logical errors and hide code reasoning. Prefer **Movable** or **Copyable**.

Defaultable create with no arguments — `builtin/value.mojo`

`__init__(out self)`

Reach for this when you need generic default-construction without arguments.

RegisterPassable (MARKER) stored in registers, not memory; no stable address or identity — `builtin/value.mojo`

Refines: Movable · no requirements (marker)

No stable address: you can't take the address of **self** in read-convention methods. **Identifiable** is meaningless for these types.

Moved trivially; all fields must also conform.

TrivialRegisterPassable (MARKER) register-passable, copyable by moving bits, no side effects — `builtin/value.mojo`

Refines: ImplicitlyCopyable < Copyable < Movable, ImplicitlyDeletable, RegisterPassable · no requirements (marker)

A type whose values are treated as basic bit patterns. No constructors or destructors needed. All fields must also conform.

FORMAT

Writable format itself as text; works with print(), String(), format strings — `format/__init__.mojo`

`write_to(self, mut writer: Some[Writer])`	provided
`write_repr_to(self, mut writer: Some[Writer])`	provided

If all fields conform, you inherit both methods through reflection.

Writer a destination for a custom output target — `format/__init__.mojo`

String, FileHandle, FileDescriptor conform

`write_string(mut self, string: StringSlice)`	
`write[*Ts: Writable](mut self, *args: *Ts)`	provided

Use for loggers, network streams, string builders, etc.

TESTING

Strategy produces random inputs for property-based tests — `testing/prop/strategy/__init__.mojo`

Refines: ImplicitlyDeletable, Movable

`Value: Copyable & ImplicitlyDeletable`	associated type
`value(mut self, mut rng: Rng) raises -> Self.Value`	

Allows strategies to carry and advance state between draws. **value()** draws one sample from the random number generator.

ACCELERATOR TRAITS

DevicePassable marks a type as passable to an accelerator — `builtin/device_passable.mojo`

`comptime device_type: AnyType`	the on-device type
`_to_device_type(self, mut enc, ...)`	DeviceContext hook

A host type implements this so it can be handed to a GPU or other accelerator. **DeviceContext** calls the conversion hook to turn host into device at kernel launch.

DeviceTypeEncoder encodes host values into device layout — `builtin/device_passable.mojo`

`target() -> _TargetType`	device target
`encode_device_ptr(mut self, ...)`	required
`encode[T](mut self, value, dst)`	provided (+ fields, tuple, array)

Encodes a value's fields into the accelerator's data layout.

COMPARE & HASH

Equatable equality; enables == and != — `builtin/comparable.mojo`

`__eq__(self, other: Self) -> Bool`	provided
`__ne__(self, other: Self) -> Bool`	provided

Don't use with floating-point values (use **isclose()**). **NaN != NaN**.

Mojo provides a fieldwise default. Override for caches, internal metadata, and custom behavior.

Comparable ordered comparison; < > ≤ ≥, and sort() — `builtin/comparable.mojo`

Refines: Equatable

`__lt__(self, rhs: Self) -> Bool`	
`__gt__(self, rhs: Self) -> Bool`	provided
`__le__(self, rhs: Self) -> Bool`	provided
`__ge__(self, rhs: Self) -> Bool`	provided

Implement **__lt__()** unless it's expensive. If so, override all four.

Hashable produces a hash; needed for Dict keys and Set elements — `hashlib/hash.mojo`

KeyElement = Hashable + Equatable + Movable

`__hash__(self, mut hasher: Some[Hasher])`	provided

Hasher implements a hash algorithm (the algorithm, not a hashable type) — `hashlib/hasher.mojo`

`__init__(out self)`
`_update_with_bytes(mut self, data: Span[Byte, _])`
`_update_with_simd(mut self, value: SIMD[_, _])`
`update(mut self, value: Some[Hashable])`
`finish(var self) -> UInt64`

Hashers remain alive after finalization. All three update methods are required.

Identifiable identity; same-object test, enables is / is not — `builtin/identifiable.mojo`

`__is__(self, rhs: Self) -> Bool`	
`__isnot__(self, rhs: Self) -> Bool`	provided

Excludes register-passable types, which don't have stable addresses.

CONVERT

Boolable, Intable, Floatable convert with Bool(), Int(), Float64() — `builtin/bool.mojo` · `builtin/int.mojo` · `builtin/floatable.mojo`

`__bool__(self) -> Bool`	Boolable
`__int__(self) -> Int`	Intable
`__int__(self) raises -> Int`	IntableRaising
`__float__(self) -> Float64`	Floatable
`__float__(self) raises -> Float64`	FloatableRaising

If the method raises, use the Raising variant.

Boolable unlocks if / while / and / or usage.

MATH

Absable, Powable, Roundable unary math operators — `math/math.mojo`

`__abs__(self) -> Self`	**Absable** · abs()
`__pow__(self, exp: Self) -> Self`	**Powable** · pow(), **
`__round__(self) -> Self`	**Roundable** · round()
`__round__(self, ndigits: Int) -> Self`	**Roundable** · round(), precision

Ceilable, Floorable, Truncable round toward a bound — `math/math.mojo`

`__ceil__(self) -> Self`	**Ceilable** · ceil()
`__floor__(self) -> Self`	**Floorable** · floor()
`__trunc__(self) -> Self`	**Truncable** · trunc()

CeilDivable, CeilDivableRaising ceiling division (rounds up instead of down) — `math/math.mojo`

`__ceildiv__(self, denominator: Self) -> Self`	CeilDivable
`__ceildiv__(self, denominator: Self) raises -> Self`	CeilDivableRaising

DivModable combined division and modulo; enables divmod() — `math/math.mojo`

Refines: ImplicitlyCopyable < Copyable < Movable

`__divmod__(self, denominator: Self) -> Tuple[Self, Self]`

Math outlier. The tuple is (quotient, remainder).

ITERATE

Sized, SizedRaising has a length; enables len() — `builtin/len.mojo`

`__len__(self) -> Int`	Sized
`__len__(self) raises -> Int`	SizedRaising

Iterable iterate by borrowing — `iter/__init__.mojo`

`IteratorType[iterable_mut: Bool, //, iterable_origin: Origin[mut=iterable_mut]]: Iterator`	associated type
`__iter__(ref self) -> Self.IteratorType[origin_of(self)]`	

Parameterized on mutability and origin. Yields references tied to the source lifetime.

IterableOwned iterate by consuming and owning — `iter/__init__.mojo`

`IteratorOwnedType: Iterator`	associated type
`__iter__(var self) -> Self.IteratorOwnedType`	

No origin tracking.

Iterator produces elements one at a time; the for-loop workhorse — `iter/__init__.mojo`

Refines: ImplicitlyDeletable, Movable

`Element: Movable`	associated type
`__next__(mut self) raises StopIteration -> Self.Element`	
`bounds(self) -> Tuple[Int, Optional[Int]]`	provided
`nth(var self, n: Int) -> Optional[Self.Element]`	provided

Requires an **Iterable** on the collection, and **Iterator** on the iterator. Don't rely on **bounds()** for safety checks. It's a hint.

Typed raises (**StopIteration**).

INTEROP

PathLike represents a file system path — `os/pathlike.mojo`

Path conforms

`__fspath__(self) -> String`

ConvertibleToPython can be sent to Python — `python/conversions.mojo`

Refines: ImplicitlyDeletable

`to_python_object(var self) raises -> PythonObject`

ConvertibleFromPython can be created from a Python object — `python/conversions.mojo`

Refines: Copyable < Movable, ImplicitlyDeletable

`__init__(out self, *, py: PythonObject) raises`