

# Felt252dict

Cairo currently supports a dictionary type called `Felt252Dict` that maps keys of type `felt252` to other simple types, namely `u8`, `u16`, `u32`, `u64`, `u128`, `felt252` and `nullable`. The `nullable` type enables the dictionary to contain more complex types. By default, the value `0`, or value logically equivalent to `0`, is returned for non-existing keys. Here is a simple usage example for a dictionary: `let mut dict = Felt252DictTrait::new(); dict.insert(10, 110); dict.insert(11, 111); let val10 = dict[10]; // 110 let val11 = dict[11]; // 111 let val12 = dict[12]; // 0 dict.insert(10, 120); let val10 = dict[10]; // 120`

## Dictionary Entry

Reading from a dictionary creates a copy of the value and thus requires the value to implement the `Copy` trait. However, dictionaries also support a special type of read access, using the `dictionaryentry` method, which is useful for in-place updates of the dictionary. To create an entry, call `entry(key)`, which returns the current value of the given key and a `Felt252DictEntry` object. The `Felt252DictEntry` object is a temporary owner of the dictionary and can only be destructed by calling `finalize(new_value)` on it. This ensures that the dictionary is not used while it is being modified. `let mut dict = Felt252DictTrait::new(); dict.insert(10, 110); let (val10, entry) = dict.entry(10); let new_val = do_some_calculation(val10); entry.finalize(new_val); let val10 = dict[10]; // new_val`

## Dictionary Destruction

Dictionaries are automatically destroyed when they go out of scope. However, the destruction process of a dictionary is not trivial, and the `Destruct` trait is implemented for `Felt252Dict`. This means that `Destruct` must be derived for any type that contains a dictionary. For more information, see the [Destructors](#) section.

## [derive(Destruct)]

```
struct MyStruct {  
    dict: Felt252Dict,  
}
```

[9.1.10 Array types](#) ð§9.2 Linear Types