In Place Construction

std::variant has two in_place helpers which this lesson elaborates on. Let's read more about them below!

WE'LL COVER THE FOLLOWING

- std::variant and in_place helpers

std::variant and in_place helpers

`std::variant` has two `in_place` helpers that you can use:

• `std::in_place_type` - used to specify which type you want to change/set in the variant

• `std::in_place_index` - used to specify which index you want to change/set. Types are enumerated from 0.

- In a variant `std::variant<int, float, std::string>` - `int` has the index **0**, `float` has index **1** and the `string` has index of **2**. The index is the same value as returned from `variant::index` method.

Fortunately, you don't always have to use the helpers to create a variant. It's smart enough to recognize if it can be constructed from the passed single parameter:

// this constructs the second/float:

```
std::variant<int, float, std::string> intFloatString { 10.5f };
```

For variant we need the helpers for at least two cases:

• ambiguity - to distinguish which type should be created where several could match

• efficient complex type creation (similar to optional)

You are now familiar with initialising `std::variant`, the next lesson will discuss the ambiguities you might encounter during it.