Returning std::optional

Let's learn the different ways to return an optional.

If you return an optional from a function, then it's very convenient to return just std::nullopt or the computed value.

```
std::optional<std::string> TryParse(Input input) {
    if (input.valid())
        return input.asString();

    return std::nullopt;
}

// use:
auto oStr = TryParse(Input{...});
```

In the above example you can see that the function returns std::string
computed from input.asString() and it's wrapped in optional. If the value is unavailable, then you can return std::nullopt.

Due to mandatory copy elision the optional object - oStr will be created at the caller site

Alternatively, you can also try with non-standardised Named Returned Value Optimisation. This happens when you create an object at the beginning of a function and then return it.

```
std::optional<std::string> TryParseNrvo(Input input) {
   std::optional<std::string> oOut; // empty

if (input.valid())
   oOut = input.asString();

return oOut;
}

// use:
auto oStr = TryParseNrvo(Input{...});
```

In the second example, the oStr object should also be created at the caller side.

Play with the sample, which includes extra logging to check the addresses of the optional variable.

```
#include <iostream>
                                                                                          6
#include <string>
#include <optional>
std::optional<std::string> TryParse(std::string input) {
   if (!input.empty())
        return input;
    return std::nullopt;
}
std::optional<std::string> TryParseNrvo(std::string input) {
    std::optional<std::string> oOut; // empty
    std::cout << &oOut << '\n';
    if (!input.empty())
       oOut = input;
    return oOut;
}
int main()
    auto oOut = TryParse("Hello");
    std::cout << &oOut << '\n';
    auto oOutNrvo = TryParseNrvo("Hello");
    std::cout << &oOutNrvo << '\n';</pre>
}
```

Another thing to note is that returning an optional inside {} braces isn't always a good idea. We'll learn shortly why that is the case.