https://youtu.be/Y-SSHBtvPHo

Polymorphism

Polymorphism is means "assuming many forms".

In the context of object-oriented programming, **polymorphism**) describes a paradigm in which a function may behave differently depending on how it is called. In particular, the function will perform differently based on its inputs.

Polymorphism can be achieved in two ways in C++: overloading and overriding. In this exercise we will focus on overloading.

Overloading

In C++, you can write two (or more) versions of a function with the same name. This is called **"overloading"**. Overloading requires that we leave the function name the same, but we modify the function signature. For example, we might define the same function name with multiple different configurations of input arguments.

This example of `class Date` overloads:

```
#include <ctime>

class Date {
public:
    Date(int day, int month, int year) : day_(day), month_(month), year_(year) {}
    Date(int day, int month) : day_(day), month_(month)  // automatically sets the Date to the current ye
    {
        time_t t = time(NULL);
        tm* timePtr = localtime(&t);
        year_ = timePtr->tm_year;
    }

private:
    int day_;
    int month_;
    int year_;
};
```

Instructions

Overloading can happen outside of an object-oriented context, too. In this exercise, you will practice overloading a normal function that is not a class member.

1. Create a function `hello()` that outputs, "Hello, World!"
2. Create a `class Human`.
3. Overload `hello()` by creating a function `hello(Human human)`. This function should output, "Hello, Human!"
4. Create 2 more classes and use those classes to further overload the `hello()` function.