

# Polymorphic Functions

---

What we really want are **polymorphic functions** like this:

```
// A polymorphic function
void greet(const Person& p)    // any kind of Person
{
    cout << "Hello, I'm " << p.toString() << endl;
}
```

This function is polymorphic because the formal parameter is a **reference to a base class**. (Note, **not** a base class object.) You can **pass any kind of** **Person**, such as a **Student** or an **Employee** object and it will behave appropriately.

*Polymorphic functions should operate on references or pointers to a base class. Functions should **never use pass-by-value** with base class objects.*



This course content is offered under a [CC Attribution Non-Commercial](#) license. Content in this course can be considered under this license unless otherwise noted.