CS 162P
Polymorphism

Polymorphism means taking multiple forms. While it can refer to functions, in this document the focus is on polymorphism in classes and inheritance.

Polymorphism

In the following set of classes, a base or parent class is defined for an NPC. This class is then used to create two child classes, ShopKeeper and Guide. If you are not familiar with the term NPC, this is a non-player character in a role-playing game, i.e., a created individual that exists to interact with the actual game players. Each class simply consists of a method that returns a class specific string.

Example Code

```
class NPC:
    def action(self):
        return "Do nothing"

    def base(self):
        return "NPC"

class ShopKeeper(NPC):
    def action(self):
        return "Sell something."

class Guide(NPC):
    def action(self):
        return "Give directions."


def main():
    npcs = [NPC(), ShopKeeper(), Guide()]

    for item in npcs:
        print(type(item), end=" ")
        print(item.base() + " " + item.action())
```

output

```
<class '__main__.NPC'> NPC Do nothing
<class '__main__.ShopKeeper'> NPC Sell something.
<class '__main__.Guide'> NPC Give directions.
```

The resulting output is exactly what would be expected.

An object of type NPC displays the result of calling the action method defined in the class NPC, and objects of type ShopKeeper and Guide display the results of calling their overriding definitions of action.

For the base method that is not overridden, the original definition in NPC is returned.