Python Polymorphism

The word "polymorphism" means "many forms", and in programming it refers to methods/functions/operators with the same name that can be executed on many objects or classes.

Function Polymorphism

An example of a Python function that can be used on different objects is the len() function.

String

For strings len() returns the number of characters:

Example

```
x = "Hello World!"

print(len(x))
```

Tuple

For tuples len() returns the number of items in the tuple:

Example

```
mytuple = ("apple", "banana", "cherry")


print(len(mytuple))
```

Dictionary

For dictionaries len() returns the number of key/value pairs in the dictionary:

Example

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(len(thisdict))
```

Class Polymorphism

Polymorphism is often used in Class methods, where we can have multiple classes with the same method name.

For example, say we have three classes: Car, Boat, and Plane, and they all have a method called **move():**

Example

Different classes with the same method:

```
class Car:
  def __init__(self, brand, model):
    self.brand = brand
    self.model = model
  def move(self):
    print("Drive!")


class Boat:
  def __init__(self, brand, model):
```

```
    self.brand = brand
    self.model = model
  def move(self):
    print("Sail!")
class Plane:
  def __init__(self, brand, model):
    self.brand = brand
    self.model = model
  def move(self):
    print("Fly!")
car1 = Car("Ford", "Mustang")       #Create a Car class
boat1 = Boat("Ibiza", "Touring 20") #Create a Boat class
plane1 = Plane("Boeing", "747")     #Create a Plane class
for x in (car1, boat1, plane1):
  x.move()
```

Inheritance Class Polymorphism

What about classes with child classes with the same name? Can we use polymorphism there?

Yes. If we use the example above and make a parent class called Vehicle, and make Car, Boat, Plane child classes of Vehicle, the child classes inherits the Vehicle methods, but can override them:

Example

Create a class called Vehicle and make Car, Boat, Plane child classes of Vehicle:

```
class Vehicle:
  def __init__(self, brand, model):
    self.brand = brand
    self.model = model

  def move(self):
    print("Move!")

class Car(Vehicle):
  pass

class Boat(Vehicle):
  def move(self):
    print("Sail!")

class Plane(Vehicle):
  def move(self):
    print("Fly!")

car1 = Car("Ford", "Mustang") #Create a Car object
boat1 = Boat("Ibiza", "Touring 20") #Create a Boat object
plane1 = Plane("Boeing", "747") #Create a Plane object
for x in (car1, boat1, plane1):
  print(x.brand)
  print(x.model)
  x.move()
```