Python Polymorphism

❮ Previous Next ❯

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

Get your own Python Server

```
x = "Hello World!"

print(len(x))
```

Try it Yourself »

Tuple

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

Example

Try it Yourself »

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))
```

Try it Yourself »

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:

```

  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 object
boat1 = Boat("Ibiza", "Touring 20") #Create a Boat object
plane1 = Plane("Boeing", "747")     #Create a Plane object

for x in (car1, boat1, plane1):
  x.move()
```

Try it Yourself »

Look at the for loop at the end. Because of polymorphism we can execute the same method for all three classes.

Inheritance Class Polymorphism

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

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()
```

Try it Yourself »

Child classes inherits the properties and methods from the parent class.

In the example above you can see that the `Car` class is empty, but it inherits `brand`, `model`, and `move()` from `Vehicle`.