Python - Polimorfismo

O termo "polimorfismo" refere-se a uma função ou método que assume formas diferentes em contextos diferentes. Como Python é uma linguagem de tipo dinâmico, o polimorfismo em Python é facilmente implementado.

Se um método em uma classe pai for substituído por uma lógica de negócios diferente em suas diferentes classes filhas, o método da classe base será um método polimórfico.

Exemplo

Como exemplo de polimorfismo dado a seguir, temos **shape** que é uma classe abstrata. É usado como pai por duas classes círculo e retângulo. Ambas as classes substituem o método draw() do pai de maneiras diferentes.

```
from abc import ABC, abstractmethod
class shape(ABC):
   @abstractmethod
   def draw(self):
      "Abstract method"
      return

class circle(shape):
   def draw(self):
      super().draw()
      print ("Draw a circle")
      return

class rectangle(shape):
   def draw(self):
      super().draw()
      print ("Draw a rectangle")
      return

shapes = [circle(), rectangle()]
for shp in shapes:
   shp.draw()
```

Saída

Quando você executa este código, ele produzirá a seguinte saída -

```
Draw a circle
Draw a rectangle
```

A variável **shp** primeiro se refere ao objeto círculo e chama o método draw() da classe círculo. Na próxima iteração, ele se refere ao objeto retângulo e chama o método draw() da classe retângulo. Portanto, o método draw() na classe de forma é polimórfico.