INFORMATORIO

Taller de Programación Web

Clases y Objetos

Subsecretaría de Empleo
Chaco Gobierno de todos

Ministerio de Producción, Industria y Empleo
Chaco Gobierno de todos

CHACO
Gobierno de todos

Polimorfismo

En programación orientada a objetos ***se denomina polimorfismo a la capacidad que tienen los objetos de una clase de responder al mismo mensaje o evento en función de los parámetros utilizados durante su invocación***. Un objeto polimórfico es una entidad que puede contener valores de diferentes tipos durante la ejecución del programa.
Esto se puede conseguir a través de la herencia: un objeto de una clase derivada es al mismo tiempo un objeto de la clase padre, de forma que allí donde se requiere un objeto de la clase padre también se puede utilizar uno de la clase hija.

Python, al ser de tipado dinámico, no impone restricciones a los tipos que se le pueden pasar a una función, por ejemplo, más allá de que el objeto se comporte como se espera: si se va a llamar a un método ***f()*** del objeto pasado como parámetro, por ejemplo, evidentemente el objeto tendrá que contar con ese método.
En ocasiones también se utiliza el término polimorfismo para referirse a la sobrecarga de métodos, término que se define como la capacidad del lenguaje de determinar qué método ejecutar de entre varios métodos con igual nombre según el tipo o número de los parámetros que se le pasa. En Python no existe sobrecarga de métodos (el último método sobreescribiría la implementación de los anteriores), aunque se puede conseguir un comportamiento similar recurriendo a funciones con valores por defecto para los parámetros o a la sintaxis ****params*** o *****params***, o bien usando decoradores.

Sobrecarga de métodos

La sobrecarga de métodos es también es conocida por Overriding Methods, le permite sustituir un método proveniente de la Clase Base, en la Clase Derivada debe definir un método con la misma forma (es decir, mismo nombre de método y mismo número de parámetros que como está definido en la Clase Base).

Código

```
Definir una clase con sus métodos y atributos implementando sobrecarga de métodos.

class Persona():
      def __init__(self, cedula):
          self.cedula = cedula

      def mensaje(self):
          print("Mensaje desde la clase Persona")

class Obrero(Persona):
      def __init__(self, cedula):
          Persona.__init__(self, cedula)
          self.__especialista = 1

      def mensaje(self):
          print("Mensaje desde la clase Obrero")

obrero_planta = Obrero(12345)
obrero_planta.mensaje()
```

Resultado

```
Mensaje desde la clase Obrero
```

Lo que se logra definiendo el método ***mensaje()*** en la Clase Derivada (***Obrero***) se conoce como Método Overriding haciendo que cuando se cree el objeto (en este caso ***obrero_planta***) y se llame al método ***mensaje()***, este será tomado de la propia clase y no de la Clase Base ***Persona***). Si comenta o borra el método ***mensaje()*** de la clase ***Obrero*** (Clase Derivada) y corre nuevamente el código, el método llamado será el ***mensaje()*** de la Clase Base ***Persona***.

Sobrecarga de Operadores

La sobrecarga de operadores es también es conocida por Overloading Operators, trata básicamente de lo mismo que la sobrecarga de métodos pero pertenece en esencia al ámbito de los operadores aritméticos, binarios, de comparación y lógicos.

Código

```
Definir una clase con sus métodos y atributos implementando sobrecarga de operadores.

class Punto:
      def __init__(self,x = 0,y = 0):
          self.x = x
          self.y = y

      def __add__(self,other):
          x = self.x + other.x
          y = self.y + other.y
          return x, y

punto1 = Punto(4,6)
punto2 = Punto(1,-2)

print(punto1 + punto2)
```

Resultado

```
(5, 4)
```