Trabajo de Investigación

Asignatura: Programación Orientada a Objetos

Universidad Nacional de San Juan

Facultad de Ciencias Exactas, Físicas y Naturales

Carrera: Licenciatura en Ciencias de la Computación

Integrantes:

- Lahoz Nicolas
- Gonzalez Martin

Profesores:

- Mondre Marcelo
- Margita Masanet
- Manuel Ortega

2022

Trabajo de Investigación Listas Sobrecarga de Operadores

González Martin, Lahoz Nicolas
Universidad Nacional de San Juan, Argentina

1. Introducción

Dada las pautas del trabajo en cuestión, se decidió profundizar una investigación que se centre en la utilización del método __call__ predefinido por Python. Para esto se trabajó con la creación de una clase y dicha clase se instanció con varios objetos y con la creación de una lista, con el objetivo de demostrar la utilidad de este método e introducir la función *callable* la cual nos permitirá visualizar con mayor claridad el funcionamiento del método propiamente dicho. Cabe aclarar que este método es sumamente importante cuando se necesita realizar la "llamada" a un objeto en particular, pero se desarrollará más adelante.

2. Palabras Claves

Python, Programación, Lista, Función, Objetos.

3. Desarrollo

Como se aclaró previamente, el método predefinido *__call__* se encarga de, de cierta manera, convertir al objeto creado en una función, esto quiere decir que el objeto podrá ser llamado sin un pasaje de parámetros y se obtendrán los parámetros que previamente habían sido enviados cuando se realizó la creación del mismo.

Puede resultar un poco complejo visualizar como actúa este operador, por eso se decidió incluir la explicación del funcionamiento de la función *callable* la cual realiza lo siguiente: dada una cierta lista de n componentes cualesquiera, se podrá realizar un bucle a esta lista y realizar un *print* utilizando esta función con el índice de la componente de dicha lista. Lo que hará la función será retornar *True* cuando el objeto sea "invocable" y *False* cuando el mismo no lo sea. Para que estos objetos sean invocables, la clase contenedora de ellos debe contener la función *__call__*.

A continuación, se mostrarán ejemplos del código en cuestión:

```
class Sueldo:
    __horas=0
    __pago=200

    def __init__(self, horas, pago=200):
        self.__horas = horas
        self.__pago= pago


if __name__=="__main__":

    obj = Sueldo(12)
    obj1 = Sueldo(8)
    obj2 = Sueldo(6)

    lista = [
            obj,
            Sueldo,
            obj1,
            obj2
        ]

    for i in lista:
            print(callable(i))
```

Resultados en consola:

```
False
True
False
False
```

Como se puede observar en el siguiente código el cual no contiene el método *__call__*, si se ejecuta este programa se obtendrán cuatro valores: False, True, False, False. Esto sucede porque *obj, obj1 y obj2* no son invocables, pero la clase *Sueldo* si lo es. Para que estos tres objetos sean invocables se debe incluir el método, lo cual quedaría de la siguiente manera

```
class Sueldo:
    __horas=0
    __pago=200

    def __init__(self, horas, pago=200):
        self.__horas = horas
        self.__pago= pago
    def __call__(self):
        return self.__horas, self.__pago,
self.__horas*self.__pago


if __name__=="__main__":

    obj = Sueldo(12)
    print(obj())
    obj1 = Sueldo(8)
    print(obj1())
    obj2 = Sueldo(6)
    print(obj2())


    lista = [
            obj,
            Sueldo,
            obj1,
            obj2
        ]


    for i in lista:
            print(callable(i))
```

Resultados en consola:

```
(12, 200, 2400)
(8, 200, 1600)
(6, 200, 1200)
True
True
True
True
```

Como se puede observar ahora que está el método, se obtiene en consola gracias a la función *callable* cuatro True, ya que los tres objetos son invocables, y también se puede observar que al invocar con *print* dichos objetos se obtienen los tres datos: horas, pago por hora y sueldo.

Conclusiones

En conclusión, se puede decir que la funcionalidad que ofrece el método *__call__* se hace muy útil a la hora de obtener el retorno de una instancia de la clase dentro de una lista. También se puede obtener el retorno de valores booleanos en específico de la instancia de la clase por medio de la función *callable()* pudiendo así determinar si el objeto es invocable o no

Referencias

[1] PythonDocs: https://docs.python.org/3/library/functions.html?highlight=callable#callable

[2] Sitio oficial de funcionalidades de Python: https://python-intermedio.readthedocs.io/es/latest/

Dirección de Contacto del Autor/es:

Gonzalez Martin
San juan
Argentina

Lahoz Nicolas
San juan
Argentina