Programación en Python y C++

Programación Orientada a Objetos

Rafael De Luna Loredo

Facultad de Ciencias, UASLP

Indice

Programación Orientada a Objetos

Es un paradigma de programación que nos permite representar mejor conceptos de la realidad mediante objetos, que no es mas que una manera de llamar a una especie de estructura de datos pero que no trabaja como tal solo con datos sino con conceptos propios de un objeto real como un carro, como puede ser el estado de la batería, nivel del tanque, la velocidad actual o velocidad máxima, de igual manera no solo estados sino también acciones como puede ser el avanzar, pitar, encender o abrir cajuela.
Entonces la POO es una manera de interpretar objetos del mundo real en términos de programación.

Elementos clave

- Las acciones o comportamientos se llaman métodos
- Se basan en una plantilla llamada clase
- Los datos o información que contiene se llaman atributos
- Los métodos de una clase son propios de esa clase
- Los atributos y métodos pueden ser públicos, privados o heredados

Clases

Son una plantilla sobre la que se construye un objeto, como si del plano de un edificio se tratara o del diseño de un carro, puede resultar algo confuso pero resumiendo, el objeto es cuando ya puedes interactuar con el, en el caso del edificio es cuando ya lo ves construido.

Constructor

Es un método que construye el objeto, normalmente se utiliza para darle un estado inicial a los atributos de un objeto pero esto no es obligatorio.

destructor

Es un método que destruye el objeto, se utiliza para realizar algunas acciones, por ejemplo en el caso de un carro, si el motor falla se enviara la alerta al tablero.

Miembros de una clase

Hacen referencia a los componentes de una clase, por ejemplo en un carro, tenemos espejos, motor, llantas, asientos, puertas y más. En una clase tendríamos esos mismos componentes pero haciendo referencia mediante un tipo de dato.

Público

Nos dice que cualquiera puede hacer uso de la clase o de los métodos de la misma, un modo de verlo sería como tener una casa sin seguro, cualquiera podría entrar.

Privado

Cuando una clase es privada quiere decir que solo la clase puede hacer uso de sus métodos, aquí sería como tener una casa con seguro y solo quien tenga llave podrá entrar

Amigo

Aplica cuando queremos que otra clase o una función tenga acceso a los miembros de una clase, en el caso de la casa, sería como darle una llave a un amigo o familiar para que pueda entrar.

Protegido

Es parecido al nivel amigo, sin embargo aplica para herencia de clases que veremos más adelante, sirve para dar acceso a los miembros privados de una clase a sus clases derivadas, es como cuando tus papás te dan permiso para algo.

Clase Padre

Es la clase hija principal o de la que las demás heredan y que tiene características comunes a todas las subclases o clases hijas, un ejemplo sería una clase vehículo como clase principal y las clases carro y motocicleta como subclases

Clase Hija

Es la clase que hereda los métodos y atributos de la clase padre, tienen muchas características en común pero es necesario diferenciar una subclase de otra, como por ejemplo una motocicleta y un carro, como características comunes ambos son vehículos a motor y tienen ruedas, pero no son iguales en tamaño ni en capacidad

Polimorfismo

Es la capacidad de una clase base de actuar como una de sus clases derivadas, es decir puede adoptar múltiples formas.

Clases abstractas

Son clases de las clases no se pueden declarar instancias, es decir solo son una plantilla, ya que solo sirven de base para poder crear las subclases u objeto de sus clases derivadas, un ejemplo sería la clase vehículo, si declaramos una clase vehículo no podríamos saber de que tipo de vehículo se trata.

Métodos virtuales

Son métodos de una clase abstracta pero que su definición o redefinición se hace en las clases heredadas. Una clase abstracta contiene por lo menos un método virtual.

Interfaces

Una interfaz es una clase abstracta que solo tiene métodos virtuales, son muy útiles para declarar funcionalidades generales .

Referencias I

[1] Bhasin, H.
Python Basics, 3 ed.
David Pallai, Mercury Learning and Information, 2019.

[2] Brassard, G., and Bratley, P.
Algorithmics: Theory anda Practice, 31 ed.
Prentice Hall, 1988.

[3] CAIRÓ, O., and Guardati, S.
Algorithms, 3 ed.
McGraw-Hill Interamericana, 2006.

[4] Deitel, P., and Deitel, H.
C++ Como Programar, 9 ed.
Pearson Educación de México, 2014.

Referencias II

[5] Downey, B., A.
Think Python, 2 ed.
OŔeylly, 2016.

[6] Jaworski, M., and Ziadé, T.
Expert Python Programming, 3 ed.
Packt Publishing, 2019.

[7] Laakmann, M., G.
Cracking the Coding Interview, 6 ed.
CarrerCup, 2016.

[8] Matthes, E.
Python Crash Course, 2 ed.
No Star Press, 2019.

Referencias III

[9] Ramalho, L.
Fluent Python, 1 ed.
OReylly, 2014.

[10] Reek, K.
Pointers On C.
Addison-Wesley Longman, 1997.

[11] Sedgewick, R., and Wayne, K.
Algorithms, 4 ed.
Pearson Education, 2011.

[12] Stroustrup, B.
The C++ Programming Language, 4 ed.
Pearson Education, 2013.

Referencias IV

[13] Stroustrup, B.
Pricnciples and Practice Using C++, 2 ed.
Pearson Education, 2014.

[14] unknown.
Cplusplus.