TIPOS DE DATOS EN PYTHON

**Python es un lenguaje fuerte y dinámicamente tipado, con conteo de referencias.** Todos los objetos que se crean y se usan tienen un tipo.

**Un lenguaje de programación es fuertemente tipado si no se permiten violaciones de los tipos de datos**, es decir, dado una variable de un tipo concreto, no se puede usar como si fuera una variable de otro tipo distinto a menos que se haga una conversión(cast). No hay una única definición de este término.

**Un lenguaje de programación es dinámicamente tipado si una misma variable puede tomar valores de distinto tipo en distintos momentos**. La mayoría de los lenguajes de tipado dinámico son interpretados, como Python o Ruby.

* **En Python todos los datos son objetos y son tipo referencia.**
* Una variable tipo referencia guarda su dirección en la pila, pero el objeto propiamente dicho se almacena en el heap (memoria dinámica)
* **Una clase es un tipo referencia, de ahí que los objetos instancias de esa clase se almacenan en el montículo** (heap).

TIPOS DE DATOS MUTABLES E INMUTABLES

Los tipos de datos inmutables son los más sencillos de utilizar en programación (suelen ser los tipos de datos simples: String, Integer, Boolean, etc. Estos datos son los que mayor uso hacen de la memoria.

**Es decir, una variable está compuesta por dos espacios de memoria. El primero se encuentra en la pila y contiene la dirección de memoria del segundo. El segundo se encuentra en el montículo y contiene el valor de la variable.**

**Conclusión: cada vez que una variable, que apunta a un valor inmutable, «cambia» su valor, la misma apuntará a una nueva ubicación de memoria.** La anterior celda de memoria la liberara el garbage collector

Los tipos de datos mutables son los más “complejos” de utilizar en programación (suelen ser las estructuras de datos como: dict, list, clases, etc.), no solo son más complejos porque son estructuras datos que almacenan datos inmutables o mutables, sino que suelen complicarse con el tema de los punteros; y **hacen menos uso de la memoria porque se escriben una sola vez y se reutilizan.**

Hay que decir que los tipos de datos mutables están diseñados así, porque copiar una estructura de datos entera tardaría mucho e implicaría utilizar mucha memoria para seguramente no aprovechar la copia.

Conteo de referencias:

Imagen que contiene Diagrama

Descripción generada automáticamente

Pasaje por valor:

Diagrama

Descripción generada automáticamente

Pasaje por referencia:

Diagrama

Descripción generada automáticamente

CLASES EN PYTHON

**Una clase describe un conjunto de objetos, de características similares**. Es como una **plantilla** que describe cómo deben ser las instancias de dicha clase, de forma que cuando creamos una instancia ésta tendrá exactamente los mismos métodos y atributos que los que tiene la clase. Una clase es un nuevo tipo de dato.

Contiene:

* **otros datos** (que pueden ser de cualquier tipo), denominados formalmente: **atributos.**
* **funciones**, que operan sobre esos datos, denominadas formalmente: funciones miembro o **métodos.**

Las clases pueden contener funciones. A éstas se les denomina MÉTODOS. La forma de crearlos en Python es en la declaración de la clase

class NuevaClase:

def metodo1(self,[parámetros]):

codigo\_metodo1

**Donde self Es el primer parámetro de cualquier método (que no sea método de clase). Hace referencia al objeto que envía el mensaje, Nunca se pasa como parámetro cuando se llama a un método. Es un parámetro implícito.**

Un objeto es un agregado de datos y de métodos que permiten manipular dichos datos, y un programa en Python es un conjunto de sentencias que interaccionan con los objetos de la aplicación a través de mensajes.

Los datos y métodos contenidos en una clase se llaman miembros de la clase y se accede a ellos siempre mediante el operador "."

El control de acceso a miembros de una clase se hace de tres formas:

* **Público: Puede ser accedido desde cualquier código**, inclusive fuera de la clase. Es un atributo o método sin ningún decorador.
* **Protegido: Desde una clase sólo puede accederse a miembros protegidos de objetos de esa misma clase**, también pueden accederlo las subclases también sin hacer uso de métodos, es decir en forma directa, usando el operador «.». Un atributo o método precedido por un símbolo underscore “\_”, es un miembro protegido.
* **Privado: Sólo puede ser accedido desde el código de la clase a la que pertenece.** Un atributo o método precedido por doble símbolo underscore “\_\_”, es un miembro privado. Es te tipo de visibilidad, garantiza el encapsulamiento de la clase.

Tipo Abstracto de Datos

Generalmente los datos miembros o atributos de una clase han de definirse privados, es decir, ocultos a otras clases y programas, siendo posible el acceso a ellos solo a través de los métodos públicos de la clase.

Tipo de dato abstracto: Definir un tipo abstracto de datos implica:

1. definir un tipo de datos
2. definir un conjunto de operaciones abstractas sobre objetos de ese tipo
3. el encapsulamiento completo. Esto implica, que se pueden manipular objetos de ese tipo exclusivamente a través del uso de las operaciones definidas para él.

Se puede trabajar con tipos de datos abstractos completos solo cuando se trabaje con Lenguajes Orientados a Objetos, donde se encapsulan en un solo módulo, los datos y los procesos que actúan sobre los mismos, dando lugar a un tipo de datos abstracto, «las clases».

CREACION DE OBJETOS

Toda clase tiene un método predeterminado especial, llamado constructor y que tiene como función inicializar los atributos del objeto.

* La creación de un objeto se hace a través de la invocación del método constructor unObjeto = Punto()
* La invocación al método constructor de la clase genera espacio en memoria (heap)para almacenar un objeto de clase, y devuelve una referencia a dicha ubicación de memoria.
* Los objetos se almacenan en memoria dentro del heap o montículo, ya que pertenecen a los tipos de datos mutables (aunque se los puede hacer inmutables)

Un constructor es un método que es invocado por el intérprete al momento de creación de un objeto y se encarga de llevar a cabo todas las tareas de inicialización de los datos miembro del objeto.

El constructor, en la clase, tiene el nombre reservado \_\_init\_\_, y recibe como primer parámetro, una referencia al objeto que invoca el método (self), puede contener otros parámetros formales, que normalmente, representan los valores con los que se inicializarán los atributos.

El constructor no devuelve ningún valor explícito. No es obligatorio definir un constructor para cada clase, y en caso de que no se defina ninguno el intérprete creará uno por nosotros sin parámetros ni instrucciones. (constructor por defecto u omisión).

EL constructor por defecto u omisión asigna valores por defecto a los datos miembros del objeto. Se puede definir solo un constructor para cada clase. Si se define un constructor, este ocultará al constructor provisto por el lenguaje.

**cuando se crea una instancia de una clase, se ejecutan dos métodos: \_\_new\_\_ e \_\_init\_\_. El método \_\_new\_\_ es un método de clase, que necesariamente pide el espacio de memoria, y devuelve la referencia que luego es usada en todos los métodos de la clase (que no sea métodos de clase). Generalmente no reescribiremos el método \_\_new\_\_ porque su función no puede ser reemplazada, puede ser extendida.**

LISTAS PYTHON:

Las listas en Python son estructuras de datos secuenciales, que vienen incluidas con el Core del lenguaje (no es necesario importar paquetes para poder usarlos), y que permiten almacenar en su interior referencias a objetos.

El acceso a los componentes de la lista se puede hacer de dos formas:

1. Accediendo componente a componente, como si fuera un arreglo, el primer componente es indexado desde 0.
2. Obteniendo un objeto de la lista a través de un iterador

Organización de los programas en Python

* Los módulos en Python son una forma de agrupar funciones, métodos, clases y datos, que se relacionan.
* Un módulo es un archivo conteniendo definiciones y declaraciones de Python.
* El nombre del archivo es el nombre del módulo con el sufijo .py agregado.
* Dentro de un módulo, el nombre de este (como una cadena) está disponible en el valor de la variable global \_\_name\_\_.
* Python viene con una biblioteca de módulos estándar, descripta en un documento separado, la Referencia de la Biblioteca de Python (de aquí en más, “Referencia de la Biblioteca”).
* Algunos módulos se integran en el intérprete; estos proveen acceso a operaciones que no son parte del núcleo o Core del lenguaje pero que sin embargo están integrados, tanto por eficiencia como para proveer acceso a primitivas del sistema operativo, como llamadas al sistema.
* El conjunto de tales módulos es una opción de configuración que depende de la plataforma subyacente.

En Python, cada uno los archivos .py se denominan módulos. Estos módulos, a la vez, pueden formar parte de paquetes. Un paquete, es una carpeta que contiene archivos .py.

* La importación de módulos debe realizarse al comienzo del documento, en orden alfabético de paquetes y módulos.
* Primero deben importarse los módulos propios de Python. Luego, losmódulos de terceros y finalmente, los módulos propios de laaplicación.
* Entre cada bloque de imports, debe dejarse una línea en blanco.

ARREGLOS NUMPY

Una tabla n-dimensional es un tipo especial de variable capaz de almacenar en su interior y de manera ordenada uno o varios datos de un determinado tipo.

El paquete NumPy, provee una estructura de datos arreglo n-dimensional. NumPy es un paquete externo de Python, se debe agregar a los paquetes y módulos provistos por el Core del lenguaje. La clase más importante definida en NumPy es un arreglo n-dimensional, la clase a la que pertenecen los objetos es ndarray, debe quedar claro que los objetos de esta clase no son arreglos propiamente dicho. Este objeto arreglo es una colección de ítems todos del mismo tipo, que puede accederse usando un subíndice desde la posición 0 hasta la dimensión menos uno.

Cada ítem del arreglo ndarray tiene el mismo tamaño en memoria, todos los ítems son del tipo descripto por el objeto tipo de dato (data-type), al objeto tipo de datos que determina el tipo escalar almacenado en el arreglo, se lo denomina dtype. Se puede acceder a cualquier item del arreglo usando un subíndice, el elemento accedido es un tipo escalar de Python.

Una de las formas de creación de un arreglo del tipo ndarray, es utilizando la función empty, para crear un arreglo de referencias nulas. Para crear el arreglo de un tipo de datos específico, el tipo de datos debe establecerse en el parámetro dtype.

ARCHIVOS CSV

Los archivos CSV (del inglés comma-separated values) son un tipo de documento en formato libre, sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas o punto y coma, o cualquier otro delimitador, ya que la coma es el separador decimal: 1,0,0,0, y las filas por saltos de línea.

REFERENCIA SELF

* En Python todo método de una clase (que no sea método de clase), recibe un primer parámetro formal que es una referencia al objeto que recibe el mensaje, luego si existen irán el resto de los parámetros formales.
* Por convención, los programadores Python, decidieron llamar a dicho parámetro como self.
* Python hace obligatorio el uso de la referencia implícita en el interior de los métodos de la clase para acceder a los atributos y otros métodos.

FUNCIONES Y METODOS CON PARAMETROS POR DEFECTO

* En Python no existe la sobrecarga de funciones y métodos como se la conoce en otros lenguajes como C# y Java.
* Existen los parámetros por defecto, que hacen que a una función se la pueda invocar de distintas maneras.
* El intérprete es el encargado de identificar qué parámetros formales toman valor por defecto y cuales no, en función de la cantidad de parámetros actuales en la invocación de la función o método.

Valores por defecto en constructores:

* Como ya se pudo observar en los ejemplos anteriores, los constructores pueden generarse con valores por defecto.
* Ésta forma de trabajo permite asegurarse que los objetos de una clase tengan una inicialización en todos sus atributos, aunque el programador no pase los valores iniciales al instanciar un objeto.

DATOS MIEMBRO ESTATICOS Y FUNCIONES MIEMBRO ESTATICAS

* Cada objeto de una clase tiene su propia copia de todos los datos miembros de la clase a la que pertenece (datos miembro no estáticos).
* Un dato miembro de una clase se puede declarar estático, se declara sin decoradores y con el valor inicial que tendrá para todos los miembros de la clase. El programador es responsable de accederlo exclusivamente con funciones miembro estáticas, o a través del nombre de la clase.
* A un miembro estático se le asigna una zona fija de almacenamiento, al igual que una variable global, pero de clase.
* Un dato miembro estático es, por lo tanto, una variable de clase según el Paradigma de la Orientación a Objetos. Una variable de clase representa información “propia de la clase” (es decir, un atributo de la clase, no un atributo de un objeto específico).
* Un dato miembro estático es compartido por todas las instancias (objetos) de una clase y existe, incluso cuando ningún objeto de esta clase existe.
* Las funciones miembros estáticas sólo pueden acceder a otras funciones y datos miembros estáticos.
* Las funciones miembros estáticas representan los métodos de clase propios de los lenguajes puros orientados a objetos, por cuanto los métodos de clase manipulan solamente las variables de clase.
* Los métodos estáticos o métodos de clases se pueden ejecutar, simplemente haciendo referencia a la clase que lo contienen, no es necesario instanciar la clase para poder hacer uso del dato miembro o de la función miembro de clase.
* En Python, para declarar una función como método de clase, se utiliza el decorador @classmethod.
* Un método de clase recibe la clase (cls) como primer argumento implícito, al igual que un método de instancia recibe la instancia a través de self.

SOBRECARGA DE OPERADORES

La sobrecarga de operadores significa simplemente, capturar las operaciones básicas incluidas en el lenguaje estándar de Python, como lo son + (suma), - (resta), \* (multiplicación), / (división), operadores de comparación > (mayor que), < (menor que), == (igual), etc., en las clases definidas por el programador, de modo que tengan el mismo significado que lo tienen para los tipos predefinidos del lenguaje.

Ideas clave en la sobrecarga de operadores en Python:

* La sobrecarga de operadores permite que las clases intercepten el comportamiento de las operaciones normales de Python.
* La sobrecarga de operadores se implementa al proporcionar métodos especialmente nombrados en una clase.
* La sobrecarga de operadores hace que las instancias de las clases definidas por el programador se comporten como los tipos integrados del lenguaje.

Python, ofrece la sobrecarga por derecha (o reversa), para los operadores como + (suma), -(resta), \* (multiplicación), / (división), % (módulo), etc., a los que denomina \_\_radd\_\_,\_\_rsub\_\_, \_\_rmul\_\_, \_\_rdiv\_\_, \_\_rmod\_\_, respectivamente. Estos operadores permiten que las operaciones que son conmutativas lo sigan siendo, y que las operaciones que involucren valores de distintas clases se puedan llevar a cabo.

DESTRUCCION DE OBJEROS DE UNA CLASE

Destructores y recolección de basura:

El lenguaje de programación Python, utiliza el mecanismo de recolección de basura (GarbageCollector), para obtener el espacio de objetos que ya no están referenciados. El principio fundamental que utiliza se basa en el ciclo de vida de los objetos, si un objeto deja de estar referenciado, es candidato a la recolección de basura. Las clases pueden proveer un método para la liberación de recursos que ya no se ocupen, a este método se los denomina destructor.

Si el programador no provee el código de un destructor, la clase tendrá un destructor por defecto u omisión. El destructor es invocado automáticamente por el Garbage Collector (recolector de basura) para liberar los recursos utilizados por el objeto que se va a destruir.

El programador puede escribir código específico para la liberación de recursos, este código se ubicará en el destructor de la clase, que tiene el nombre reservado \_\_del\_\_.

CONTEO DE REFERENCIAS- COLECTOR DE BASURA

A diferencia de muchos otros lenguajes, Python no necesariamente libera la administración de memoria al sistema operativo.

Tiene un asignador de objetos dedicado para objetos de menos de 512 bytes, que mantiene algunos fragmentos de memoria ya asignada para su uso posterior.

El garbage collector estándar de Python tiene dos componentes:

* El recolector de basura por conteo de referencias
* El recolector de basura generacional, conocido como el módulo gc

El algoritmo de conteo de referencias es increíblemente eficiente, pero no puede detectar referencias cíclicas. Es por eso por lo que Python tiene un algoritmo suplementario llamado GC generacional cíclico, que trata específicamente con las referencias cíclicas.

El módulo de conteo de referencias es fundamental para Python y no se puede deshabilitar, mientras que el GC cíclico es opcional y se puede invocar manualmente.

Si el contador de referencias llega a cero, automáticamente se invoca a la función deliberación de memoria para el objeto específico

Para eliminar algo de la memoria, se debe asignar un nuevo valor a una variable o salir de un bloque de código. En Python, el bloque de código más popular es una función. Aquí es donde ocurre la mayor parte de la recolección de basura. Esa es otra razón para mantener las funciones pequeñas y simples.

Se puede conocer la cantidad de referencias de un objeto usando la función sys.getrefecount

COLECTOR GENERACIONAL

¿Por qué se necesita un recolector de basura adicional cuando se tiene el conteo de referencias?

Desafortunadamente, el conteo clásico de referencias tiene un problema fundamental: no puede detectar referencias cíclicas. Una referencia cíclica ocurre cuando uno o más objetos están haciendo referencia entre sí.

Diagrama

Descripción generada automáticamente