INTRODUCCIÓN A LA O.O:
Programación Imperativa: los algoritmos se describen en base a procesos.
Programación modular: el problema se divide en módulos autónomos que se pueden programar, verificar y modificar individualmente.

Tipo de Datos Abstracto:

1. Un tipo de datos definido por el programador
2. Un conjunto de operaciones abstractas sobre objetos de ese tipo
3. Encapsulamiento de los objetos de ese tipo, de tal manera que el usuario final del tipo no pueda manipular esos objetos excepto a través del uso de las operaciones definidas.

Características del Diseño O.O:

- La programación orientada a objetos encara la resolución de cada problema desde la óptica del objeto.
- El objeto combina los datos con los procedimientos u operaciones que actúan sobre dichos datos.
- Los objetos interactúan entre sí enviando mensajes
- Los métodos son muy similares a los procedimientos de programación imperativa tradicional y los mensajes se podrían pensar como invocaciones a esos procedimientos.
- El programador orientado a objetos puede agrupar características comunes de un conjunto de objetos en clases, a través de un proceso de abstracción.
- Los descendientes de estas clases se construyen por medio del mecanismo de subclasificación, permitiendo que sean heredados los métodos programados anteriormente y debiendo programar solamente las diferencias.
- La Programación Orientada a Objetos no se puede desligar de todo el paradigma de orientación a objetos.
- El principio fundamental del paradigma de programación orientada a objetos es construir un sistema de software en base a las entidades de un modelo elaborado a partir de un proceso de abstracción y clasificación.
- Para hacer buena POO hay que desarrollar todo el sistema utilizando el paradigma empezando por un análisis y un diseño orientados a objetos.
- En general, el desarrollo de software implica, desde una visión orientada a objetos tiene una serie de etapas para hacer: requerimientos, análisis, diseño, implementación, prueba.

UML

Es un lenguaje que permite la visualización, especificación y documentación de sistemas orientados a objetos, no es una metodología sino una notación, que aglutina distintos enfoques de orientación a objetos, UML 2 define trece diagramas para describir distintas perspectivas del sistema.

ELEMENTOS BÁSICOS DE POO:
Objetos: Un objeto es una unidad atómica que encapsula estado y comportamiento, la encapsulación en un objeto permite una alta cohesión y un bajo acoplamiento.
Los objetos son entidades que tienen atributos y comportamiento particular.
Atributos de un objeto: Los atributos describen la abstracción de características individuales que posee un objeto.
Comportamientos: Los comportamientos de un objeto representan las operaciones que pueden ser realizadas por un objeto.

Objeto = Estado + Comportamiento + Identidad

- El estado agrupa los valores instantáneos de todos los atributos de un objeto, evoluciona con el tiempo.
- El comportamiento describe las acciones y reacciones de ese objeto.

- Las acciones u operaciones de un objeto se desencadenan como consecuencia de un estímulo externo, representado en forma de un mensaje enviado por otro objeto. El estado y el comportamiento están relacionados.
- La identidad permite distinguir los objetos de forma no ambigua, independientemente de su estado. Esto permite distinguir dos objetos en los que todos los valores de los atributos son idénticos.

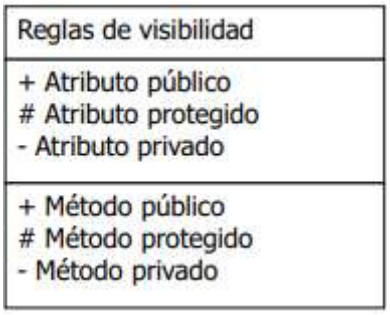
Constructor

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.
CLASES:
Una clase abstrae las características de un conjunto de objetos con comportamientos similares.
La encapsulación de una clase permite la cohesión y presenta distintas ventajas básicas:

- Se protegen los datos de accesos indebidos
- El acoplamiento entre las clases se disminuye
- Favorece la modularidad y el mantenimiento

Una clase es una descripción de un conjunto de objetos, ya que consta de comportamientos y atributos que resumen las características comunes del conjunto.
La posibilidad de definir clases es una de las ventajas de la orientación a objetos; definir clases significa colocar código reutilizable en un depósito común en lugar de redefinirlo cada vez que se necesite, cada objeto es instancia de una clase.

Encapsulamiento de una clase — Visibilidad
-Público:
Accesible desde cualquier código, incluso fuera de la clase.
Se define sin ningun decorador.
-**Protegido:**
Accesible desde la misma clase y sus subclases.
Se puede acceder directamente usando el operador ..
Se indica con un simbolo underscore (_).
-Privado:
Solo accesible desde el codigo de la clase a la que pertenece.
Se indica con doble simbolo underscore (__).
Garantiza el encapsulamiento de la clase.



Métodos y mensajes
Los objetos tienen la posibilidad de actuar, la acción sucede cuando un objeto recibe un mensaje, que es una solicitud que pide al objeto que se comporte de manera determinada.

- Cada objeto recibe, interpreta y responde a mensajes enviados por otros objetos.
- Los comportamientos u operaciones que caracterizan un conjunto de objetos residen en la clase y se llaman métodos.
- Los métodos son el código que se ejecuta para responder a un mensaje, y el mensaje es la llamada o invocación a un método.

Variables de instancia
Las variables de instancia o miembros dato se usan para guardar los atributos de un objeto particular.

Variables de clase
son aquellos atributos que tienen el mismo valor para cada objeto de la clase. Representa un área de memoria compartida por todos los objetos de la clase.

HERENCIA:
Generalización: consiste en factorizar los elementos comunes (atributos, métodos y restricciones) de un conjunto de clases en una clase más general llamada súper clase.

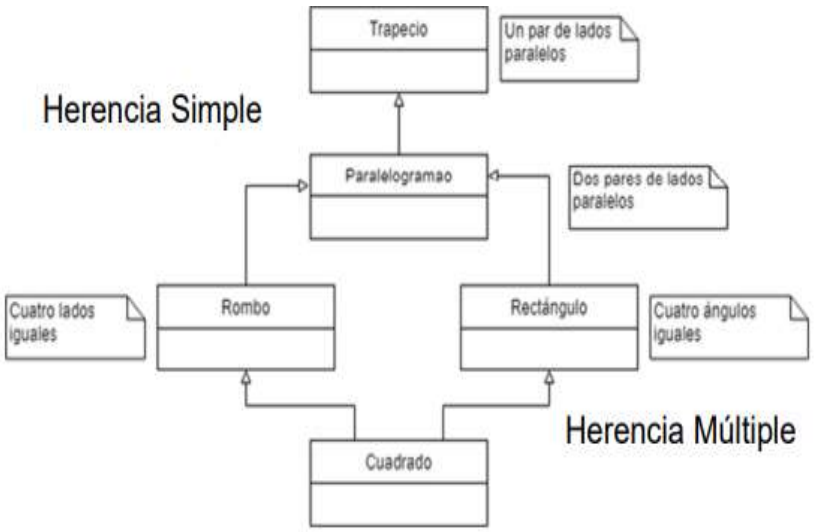
Especialización: permite capturar las particularidades de un conjunto de objetos no discriminadas por las clases ya identificadas.

- Las clases se ordenan según una jerarquía, una súper clase es una abstracción de sus subclases.
- Los árboles de clases se determinan partiendo de las hojas mientras que los niveles superiores son abstracciones construidas para ordenar y comprender.
- Una subclase identifica el comportamiento de un conjunto de objetos que hereda las características de la clase padre y adicionan algunas específicas que ésta no posee.

Clase abstracta: Una clase es abstracta cuando no existe un objeto que sea instancia directa de ella, pero sí existe una subclase de ella que es instanciable.

Clase concreta: Una clase concreta es aquella que es instanciable, es decir que existe al menos un objeto de esa clase.

Tipos de herencia:



RELACIONES ENTRE CLASES:
Las relaciones muestran el acoplamiento de las clases, es el número de clases con las que una clase concreta está relacionada. Una clase está acoplada si sus objetos lo están. Un objeto está acoplado con otro si uno de ellos actúa sobre el otro.
ASOCIACIÓN:
Es una conexión entre dos clases, la comunicación puede ser tanto uni como bi-direccional, una asociación puede tener nombre y se representa por una línea continua entre las clases asociadas, la asociación no es contenida por las clases, ni subordinada a las clases asociadas.

Cardinalidad o multiplicidad

Los valores de multiplicidad más comunes son:

- **1** uno y sólo uno
- **0..1** cero o uno (0 es una relación opcional)
- **m..n** de m a n (m y n enteros naturales)
- * de cero a varios
- **0..*** de cero a varios
- **1..*** de uno a varios

Clase asociación: Una dupla de objetos, instancias de cada una de las clases que participan en la asociación, se relaciona con una única instancia de la clase asociación. No importa la multiplicidad en ambos extremos.

Clase que modela la asociación: Para una dupla de objetos existe más de objeto asociado a la clase asociada.

AGREGACIÓN:
Reutilización y extensión: Se denomina reutilización al uso de clases u objetos desarrollados y probados en un determinado contexto, para incorporar esa funcionalidad en una aplicación diferente a la de origen.
La extensión se basa en aprovechar las clases desarrolladas para una aplicación, utilizándolas para la construcción de nuevas clases, en la misma u otra aplicación.

- La agregación consiste en definir como atributos de una clase a objetos de otras clases ya definidas.
- La agregación es una relación no simétrica en la que una de las clases cumple un papel predominante respecto de la otra.
- La agregación declara una dirección a la relación todo/parte.

Agregación (propiamente dicha): Este tipo de relación se presenta en aplicaciones en las cuales un objeto contiene como partes a objetos de otras clases, pero de tal modo que la destrucción del objeto continente no implica la destrucción de sus partes.
Los tiempos de vida de los objetos continente y contenido no están tan estrechamente acoplados, de modo que se pueden crear y destruir instancias de cada clase independientemente.

Composición: La forma más simple de reutilizar una clase es simplemente haciendo una nueva clase que la contenga. Esta técnica se llama composición. Las composiciones generan una relación de existencia entre el todo y cada una de sus partes.

- Un objeto de una clase contiene como partes a objetos de otras clases y estas partes están físicamente contenidas por el agregado.
- Los objetos agregados no tienen sentido fuera del objeto resultante. - Los tiempos de vida de los objetos continente y contenido están estrechamente acoplados
- La destrucción del objeto continente implica la destrucción de sus partes

CONCEPTOS CLAVE:
Encapsulamiento:

- Término formal que describe al conjunto de métodos y de datos de un objeto de manera tal que el acceso a los datos se permite solamente a través de los métodos propios de la clase a la que pertenece el objeto.
- La comunicación entre los distintos objetos se realiza solamente a través de mensajes explícitos.

Abstracción:

- La orientación a objetos fomenta que los programadores y usuarios piensen en las aplicaciones en términos abstractos.

- A partir de un conjunto de objetos, se piensa en los comportamientos comunes de los mismos para situarlos en superclases, las cuales constituyen un depósito para elementos comunes y reutilizables.

Polimorfismo:

- Capacidad que tienen objetos de clases diferentes, relacionados mediante la herencia, a responder de forma distinta a una misma llamada de un método.
- Fomenta la extensibilidad del software
- Software escrito para invocar comportamiento polimórfico se escribe en forma independiente del tipo de los objetos a los cuales los mensajes son enviados
- Nuevos tipos de objetos, que pudieran responder a mensajes existentes, pueden ser agregados en dicho sistema sin modificar el sistema base.

Persistencia:

- designa la capacidad de un objeto de trascender el tiempo o el espacio
- un objeto persistente conserva su estado en un sistema de almacenamiento permanente
- el objeto puede ser reconstruido por otro proceso y se comportará exactamente como en el proceso inicial

UNIDAD 2

Tipos de Datos Mutables
Los tipos de datos mutables son aquellos que pueden ser modificados después de su creación. Esto significa que puedes cambiar su contenido sin necesidad de crear un nuevo objeto.
incluyen: listas diccionarios, conjuntos

Tipos de Datos Inmutables
Los tipos de datos inmutables son aquellos que no pueden ser modificados una vez creados. Cualquier cambio requiere la creación de un nuevo objeto. Ejemplos incluyen: CADENAS, TUPLAS, DATOS NUMERICOS

Reference counting: Si a dos o más variables distintas se les asigna el mismo valor estas tomaran la misma dirección de memoria para guardarlo. Python contara cuantas variables hay apuntando a esa dirección de memoria en cada momento

El manejo de parametros en Python se realiza de la siguiente manera:

Por referencia (mutables): Los objetos como listas y diccionarios se pasan por referencia, lo que significa que cualquier modificaciónon realizada dentro de una función afectara al objeto original.
Por valor (inmutables): Los objetos como cadenas y tuplas se pasan por valor, lo que implica que cualquier cambio dentro de la función no afectara al objeto original.

Listas en Python
Las listas en Python son estructuras de datos secuenciales, que vienen incluidas con el core del lenguaje, y que permiten almacenar en su interior referencias a objetos.

Organización de los Programas en Python

Modulos: Son una forma de agrupar funciones, métodos, clases y datos relacionados.

Definición: Un modulo es un archivo que contiene definiciones y declaraciones en Python.
Nombre del Modulo: El nombre del archivo del modulo termina con el sufijo .py.
Variable Global __name__: Dentro de un módulo, el nombre del mismo está disponible como una cadena en la variable global __name__.
Esto permite estructurar el código de manera más organizada y reutilizable.

Paquetes en Python

Modulos: Los archivos con extensión .py se denominan modulos y pueden formar parte de paquetes.
Paquete: Es una carpeta que contiene uno o más archivos .py.
Archivo de inicio: Para que una carpeta sea considerada un paquete, debe incluir un archivo llamado __init__.py. Contenido del archivo: Este archivo no necesita contener instrucciones y puede estar vacío. Los paquetes permiten organizar y estructurar mejor el código en proyectos grandes, facilitando la reutilización y el mantenimiento.

Arreglos NumPy

El paquete NumPy proporciona una estructura de datos conocida como arreglo n-dimensional. Esta estructura se representa mediante la clase ndarray, que es fundamental en NumPy. Los arreglos son colecciones de elementos del mismo tipo, accesibles mediante subíndices que van desde la posición 0 hasta la dimensión menos uno.

Caracteristicas de los Arreglos NumPy:

-Homogeneidad: Todos los elementos del arreglo tienen el mismo tamaño en memoria y son del mismo tipo de dato, determinado por el atributo dtype.
-Acceso a elementos: Se puede acceder a cualquier elemento del arreglo utilizando un subíndice, y el elemento accedido es un tipo escalar de Python.
-Estructura: Los arreglos pueden tener multiples dimensiones, lo que permite representar datos de manera más compleja y eficiente.

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.

Apertura del flujo de datos

```
archivo = open('librosPOO.csv')
reader = csv.reader(archivo,delimiter=';')
```

Lectura de líneas

```
for fila in reader:
    ...
```

Cierre del flujo de datos

```
archivo.close()
```

Referencia self

Este parámetro actúa como una referencia al objeto que invoca el método, permitiendo el acceso a sus atributos y otros métodos dentro de la clase.

Por convención, se utiliza el nombre self para este parámetro, aunque se puede nombrar de otra manera.
Python exige el uso de esta referencia implícita en el interior de los métodos de la clase para interactuar con los atributos y métodos del objeto.

Funciones Miembro Estáticas

Las funciones miembro estáticas en Python son métodos que solo pueden acceder a otras funciones y datos miembros estáticos. Representan métodos de clase que manipulan exclusivamente variables de clase, sin necesidad de instanciar la clase. Se declaran usando el decorador @classmethod, y reciben la clase como primer argumento implícito. Esto permite ejecutar los métodos simplemente haciendo referencia a la clase.

Sobrecarga de Operadores
La sobrecarga de operadores en Python permite a las clases capturar operaciones básicas como suma (+), resta (-), multiplicación (*), y división (/). Esto se logra implementando métodos especialmente nombrados en la clase, permitiendo que las instancias de estas clases se comporten como los tipos integrados del lenguaje. De esta manera, las clases pueden interceptar el comportamiento de las operaciones normales de Python.

Garbage collector
El garbage collector se encarga de liberar los espacios de memoria que ya no son necesarios
El garbage collector estándar de Python tiene dos componentes:

- El recolector de basura por conteo de referencias (cuando las referencias llegan a 0 se libera la memoria asignada a la variable)
- El recolector de basura generacional que se encarga de solucionar los problemas causados por referencias cíclicas, conocido como gc

Tipos de Polimorfismo
-Polimorfismo: Es la capacidad de objetos de diferentes clases para responder de manera distinta a la misma llamada de un método.

-Polimorfismo de subtipo: Se basa en la ligadura dinámica y la herencia, permitiendo que una clase hija sobrescriba métodos de la clase padre.

-Polimorfismo en general: Hace que el código sea flexible y reutilizable, permitiendo el uso de diferentes tipos de objetos de manera intercambiable.

UNIDAD 3

Excepciones
Definición: Son eventos que pueden alterar el flujo normal de un programa debido a errores o situaciones excepcionales. En Python, se manejan mediante bloques try-except, permitiendo que el programa continúe ejecutándose o maneje el error de manera controlada.
-Assert: Permite al programador verificar condiciones en el código. Si la condición es falsa, se lanza una excepción AssertionError.

-Raise: Se utiliza para lanzar manualmente una excepción en el código, permitiendo al programador indicar errores o condiciones especiales.

-Bloque try-except-else-finally: Estructura que permite capturar y manejar excepciones. Los bloques else y finally son opcionales, y el bloque try puede manejar varias excepciones.

Testing
Definición: Es el proceso de evaluar un sistema o sus componentes para verificar que cumplen con los requisitos y funcionan correctamente. Las pruebas unitarias son una forma de testing que se centra en validar el funcionamiento de las unidades más pequeñas del código.

Listas Enlazadas
Definición: Estructuras de datos que consisten en nodos, donde cada nodo contiene un valor y una referencia al siguiente nodo en la secuencia. Las listas enlazadas permiten la inserción y eliminación eficiente de elementos.

Interfaces
Definición: Son contratos que definen un conjunto de métodos que una clase debe implementar. Las interfaces permiten la creación de sistemas más flexibles y reutilizables, ya que diferentes clases pueden implementar la misma interfaz de diversas maneras.

Persistencia de Objetos
Definición: Es el proceso de almacenar el estado de un objeto en un formato que permita su recuperación posterior. Los archivos JSON son una forma común de lograr la persistencia, ya que permiten serializar y deserializar objetos de manera sencilla.

Archivos JSON
JSON (JavaScript Object Notation) es un formato de intercambio de datos basado en texto, utilizado para el intercambio de datos en la web. Es una alternativa a XML, consumiendo menos ancho de banda y recursos. Se puede codificar y decodificar en Python usando la librería JSON.

Jerarquía de computadoras
La computadora virtual que un programador utiliza cuando decide hacer un programa en algún lenguaje de alto nivel está formada, de hecho, por una jerarquía de computadoras virtuales. De las cuales la ultima es el hardware.
Cada capa es una máquina virtual que abstrae a las máquinas del nivel inferior.
Las máquinas, en su nivel, «interpretan» sus instrucciones particulares, utilizando servicios de su capa inferior para implementarlas.

Programación Orientada a Eventos
Son modelos o patrones de programación en el que la estructura y la ejecución de los programas van determinados por los sucesos o acciones que ocurren en el sistema, definidos por el usuario o por el propio sistema.
Con los lenguajes orientados a eventos se pueden realizar en poco tiempo aplicaciones sencillas y muy funcionales, utilizando interfaces gráficas en las que se insertan componentes o controles a los que se le programan eventos.
Existen distintos tipos de programas:

- Secuenciales, no hay interacción del usuario final
- Interactivos: el usuario provee datos necesarios para la ejecución del programa
- Basados en eventos: el programa se presenta a través de una interface gráfica, y se queda esperado a que el usuario o el sistema, produzca un evento