Unidad 2 Poo:

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

- **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
- **Dinámicamente tipado:** si una misma variable puede tomar valores de distinto tipo en distintos momentos.
- 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 almacena en el el montículo (heap)

Tipos de datos Inmutables y Mutables:

- **Datos inmutables:** son los más sencillos de utilizar en programación (String, Integer, Boolean, etc.) pues hacen exactamente lo que se espera que hagan en cada momento, y para funcionar así son los que más castigan a la memoria (no están optimizados para ocupar menos memoria al copiarse, y degradan más la vida útil de la memoria al escribirse más veces).
- **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 son los que menos perjudican a la memoria (se escriben una sola vez y se reutilizan siempre).

En Python todas las variables son objetos instancias de una clase

Todo programa en Python tendrá al menos una sentencia, en el ejemplo, la instrucción print, que muestra por pantalla la texto que se le pasa como parámetro.
Todo programa Python comienza a ejecutarse en la primer sentencia fuera de una función. Es una buena práctica para los programadores Python identificar un punto de inicio, para ello se provee la siguiente sentencia en un programa:

```
if __name__==‘__main__’:
        sentencia
```

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.

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 variables incluidas en una clase se denominan ATRIBUTOS.

•Existen múltiples formas de crear atributos en una clase. La más simple:

```
class     NuevaClase:
atributo1: int
```

Las clases pueden contener funciones. A éstas se les denomina MÉTODOS. La forma de
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 "."

ej: **unPunto.inicializar(3,4)**
unPunto.mostrarDatos()

Encapsulamiento de una clase – Visibilidad:

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. Este tipo de visibilidad, garantiza el encapsulamiento de la clase.

Tipo Abstracto de Datos

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

El **proceso de ocultar** la estructura interna de datos de un objeto y permitir el acceso sólo a través de la interfaz pública definida se llama **Encapsulamiento.**

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

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 miembros 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 inicializan los atributos.

- 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)

Listas en 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.

```
if __name__ == '__main__':
    listaPersonas = []
```

Luego con el método **append**, se pueden agregar objetos a la lista.

```
if __name__ == '__main__':
    listaPersonas = []
    unaPersona = Persona()
    listaPersonas.append(unaPersona)
```

Organización de los programas 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.

Módulos:

Para poder utilizar un módulo son necesarias dos cosas:
1) Debe estar instalado.
2) Incorporarlo al programa, para ello se usa la palabra reservada import.

Módulo	Descripción
abc	*Abstract base classes de acuerdo a: pep '3119'*
csv	Lectura y escritura de archivos delimitados por un caracter
datetime	Tipos básicos para fechas y horas
gc	Interface con funciones del Garbage Collector
io	Funciones importantes para el manejo de streams
json	Codificación y decodificación de archivos en formato JSON
os	Miscelánea de interfaces con el Sistema Operativo
random	Generador de números pseudo aletorios con varias distribuciones
sqlite3	Una implementación de la API DB 2.0 usando SQLite 3.x
sys	Acceso a funciones y parámetros específicos del sistema
tkinter	Interface a Tcl/Tk para interfaces gráficas
unittest	Framework para el testeo de unidad en Python

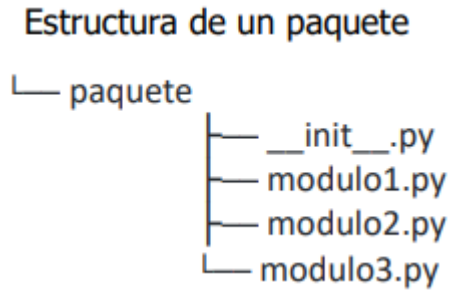
Para crear un módulo, se necesita:

1. Proveer un nombre al módulo
2. Guardar un archivo con es nombre y extensión .py
3. La importación del módulo se hace con la palabra reservada import

Otras formas de importación de Módulos:haciendo uso de la sentencia from

Paquetes 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. Para que una carpeta pueda ser considerada un paquete, debe contener un archivo de inicio llamado __init__.py. Este archivo, no necesita contener ninguna instrucción. De hecho, puede estar completamente vacío.
Los módulos no necesariamente tienen que formar parte de un paquete



Importación de módulos:

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, los módulos de terceros y finalmente, los módulos propios de la aplicación.
• 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.

- 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 items todos del mismo tipo, que puede accederse usando un subíndice desde la posición 0 hasta la dimensión menos uno**

```
import numpy as np
from clasePunto import Punto
class arregloNumPy:
    __cantidad: int
    __dimension: int
    __incremento = 5
    __puntos: n.ndarray
    def __init__(self, dimension, incremento=5):
        self.__puntos = np.empty(dimension, dtype=Punto)
        self.__cantidad = 0
        self.__dimension = dimension
    def agregarPunto(self, unPunto):
        if self.__cantidad==self.__dimension:
            self.__dimension+=self.__incremento
            self.__puntos.resize(self.__dimension)
        self.__puntos[self.__cantidad]=unPunto
        self.   cantidad += 1
```

Procesamiento de archivos en formato libre:

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

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()
```

Los métodos **__str__**, devuelve una representación string del estado de los objetos
El método **test()**, lee los datos desde el archivo CSV, separado por «;»

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**.

Datos miembro estáticos y Funciones miembro estáticas

- 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.
- 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 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 l**as variables de clase**
- **En Python, para declarar una función como método de clase, se utiliza el decorador @classmethod.**

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 la clases definidas por el programador, de modo que tengan el mismo significado que lo tienen para los tipos predefinidos del lenguaje.

- 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.

Operadores Sobrecargables (nómina reducida)		
Operador	**Método**	**Expresión**
+ Suma	__add__(self, otro)	a1+a2
- Resta	__sub__(self, otro)	a1-a2
* Multiplicación	__mul__(self, otro)	a1*a2
/ División	__div__(self, otro)	a1/a2
% Módulo	__mod__(self, otro)	a1%a2
> Mayor que	__gt__(self, otro)	a1>a2
>= Mayor o igual que	__ge__(self, otro)	a1>=a2
< Menor que	__lt__(self, otro)	a1<a2
<= Menor o igual que	__le__(self, otro)	a1<=a2
== Igual que	__eq__(self, otro)	a1==a2
!= Distinto	__ne__(self, otro)	a1!=a2

Destrucción de Objetos de una Clase

El lenguaje de programación Python, utiliza el mecanismo de recolección de basura (Garbage Collector), 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

¿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.