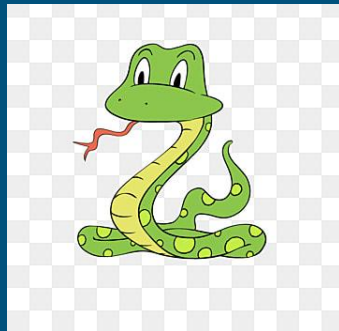


Clases y objetos



Objetivos:

- Conocer los conceptos de la POO

- Implementación de la POO en Python

Módulo

Archivo de texto plano con extensión .py

Se puede importar de la siguiente forma:

```
import nombre_modulo
```

Paquete

Directorio donde se encuentran varios módulos(archivos con extensión .py)

Clasificación de los lenguajes de programación

Por su **paradigma** → Clasificación más común

Paradigma: es un estilo fundamental de programación definido por la forma de dar soluciones a problemas.

Un paradigma proporciona y determina la visión que el programador tiene acerca de la ejecución de un programa.

En conclusión, los distintos paradigmas proporcionan diversas maneras de expresar el concepto de computación.

Programación Orientada a Objetos P00

Abstracción

Encapsulación

Herencia

Polimorfismo

Ventajas de P00

- Mantenimiento de un sistema más fácil (Agregar, modificar funcionalidad)
- Reutilización de código (Herencia)
- Seguridad

Elementos

Clase: Molde o plantilla de la cual se pueden generar objetos

Objeto: Instancia o ejemplar de una clase

<https://docs.python.org/es/3/tutorial/classes.html>

Ejemplo:



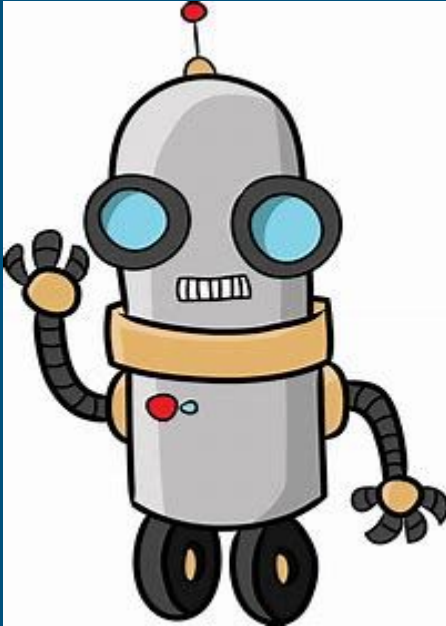
Coche

Atributos: color, marca, modelo, placas

Métodos:(Behaviour):

avanzar, detenerse, chocar, acelerar

Robot



Atributos: tamaño, nombre, edad, num_armas, peso, altura

Métodos: (Comportamiento del objeto) leer, escribir, jugar, hablar, caminar

Ejemplar, objeto :

bunny

= grande, Beto, 100, 0, 100kg, 2m (Estado de un objeto)

walle_e=

Recuerda que ...

Un objeto tiene un nombre que lo diferencia de otro objeto.

A los atributos se les conoce como **variables de instancia**

El estado de un objeto está dado por las variables de instancia

Un objeto tiene un comportamiento definido a través de sus métodos

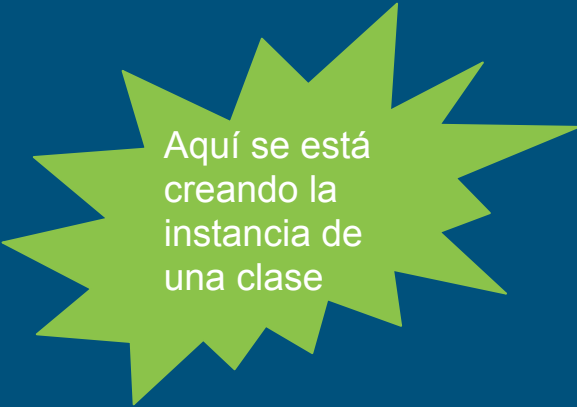
Ejemplo más básico de una clase

```
class NombreClase:
```

```
    pass
```

```
print('Termina la clase')
```

```
identificadorObjeto = NombreClase()
```



Aquí se está
creando la
instancia de
una clase

Métodos

Constructores: inicializan el estado del objeto

```
def __init__(self):
```

Métodos acceso

Acceso al valor de una atributo

```
def get_nombre_atributo(self):  
    return self.__nombre_atributo
```

Métodos modificadores

Pueden cambiar el valor de un atributo

```
def set_nombre(self, nuevo):
```

```
    self.__nombre = nuevo
```

Métodos str

Método para imprimir el estado del objeto

Ejercicio

Se necesita un programa para ayudar a los alumnos de nivel superior a comprender conceptos básicos de geometría analítica. Los conceptos con los que se va a trabajar son: **punto** **línea** y **triángulo**.

Un **punto** se define como una pareja ordena de números.
Un punto puede crearse, desplazarse,
sumarse con otro punto, restarse, calcular su distancia
con respecto a otro punto, determinar si están alineados
con respecto a otros dos puntos, determinar si es igual a
otro punto, e imprimirse en una pareja ordenada de puntos
separados por comas y entre paréntesis.

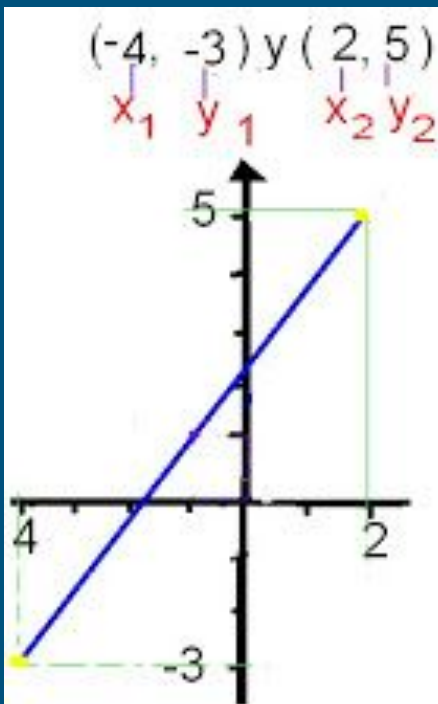
Para la **línea** recta se requieren dos puntos cualesquiera y lo que se desea hacer con ella es crearla, encontrar su ecuación, calcular su pendiente, su ordenada al origen, dadas dos rectas determinar si son paralelas, si son las mismas, si son perpendiculares, su punto de intersección y dada una recta y un punto determinar si éste está en la recta.

Para el **triángulo**, definido como tres puntos no alineados, se requiere determinar su perímetro, área y tipo (equilátero, escaleno o isósceles)

Punto

x: int

y: int



Distancia

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$d = \sqrt{(2 - (-4))^2 + (5 - (-3))^2}$$

$$d = \sqrt{(2 + 4) + (5 + 3)}$$

$$d = \sqrt{(6) + (8)}$$

$$d = \sqrt{36 + 64}$$

$$d = \sqrt{100} = 10$$

Recordando la metodología de diseño ...



Encontrar los objetos principales

Determinar el comportamiento deseado

Definir los escenarios

- Pedir nombre al usuario
- Dar la bienvenida al usuario
- Indicar al usuario que se va mostrar un programa que trabajará con puntos
- Mostrarle al usuario las diferentes operaciones que se pueden efectuar con la clase Punto

Define otro escenario

Atributos privados y encapsulamiento

Un atributo privado deberá de empezar con dos guiones bajos

Solo será accedido dentro de la clase

Asegura que un elemento externo no pueda acceder a ese atributo

Métodos constructores

- Todos los métodos empiezan con la palabra `def`
- Inicializa el estado de un objeto
- Su nombre siempre será `__init__`
- Por lo menos recibe un parámetro (`self`)

Self

Self es un nombre que se ha adaptado por convención

Representa la instancia de la clase

Ejercicio: Programar la clase Racional
