Introducción a la programación orientada a objetos con Python

Orientación a Objetos y clases

{desafío} latam_

¿Qué aprenderemos en este módulo?

En este módulo serás capaz de desarrollar y resolver problemas bajo el paradigma de la programación orientada a objetos, control de excepciones y manejo de archivos, con la finalidad de comprender el proceso de desarrollo de código, lectura e implementación de algoritmos usando al máximo las potencialidades del lenguaje Python.

{desafío} latam_

Describir los conceptos fundamentales del paradigma de orientación a objetos haciendo la distinción entre Clase y de Objeto.

- Unidad 1:
  Introducción a la programación orientada a objetos con Python
- Unidad 2:
  Tipos de métodos e interacciones entre objetos
- Unidad 3:
  Herencia y polimorfismo
- Unidad 4:
  Manejo de errores y archivos

¿Qué aprenderás en esta sesión?

- *Distingue las características de la programación orientada a objetos respecto a la programación estructurada.*
- *Describe el concepto de clase haciendo la distinción con el concepto de objeto.*
- *Explica el concepto de atributo de una clase haciendo la distinción con el concepto de estado de un objeto.*

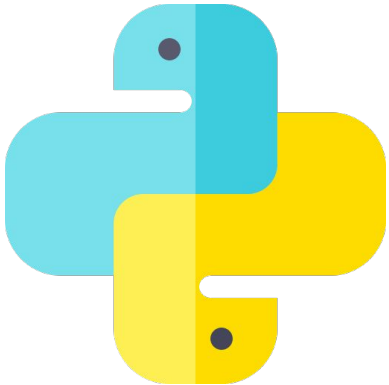
{desafío} latam_

¿Qué entendemos por paradigma?

/* Programación orientada a objetos */

¿Qué es?

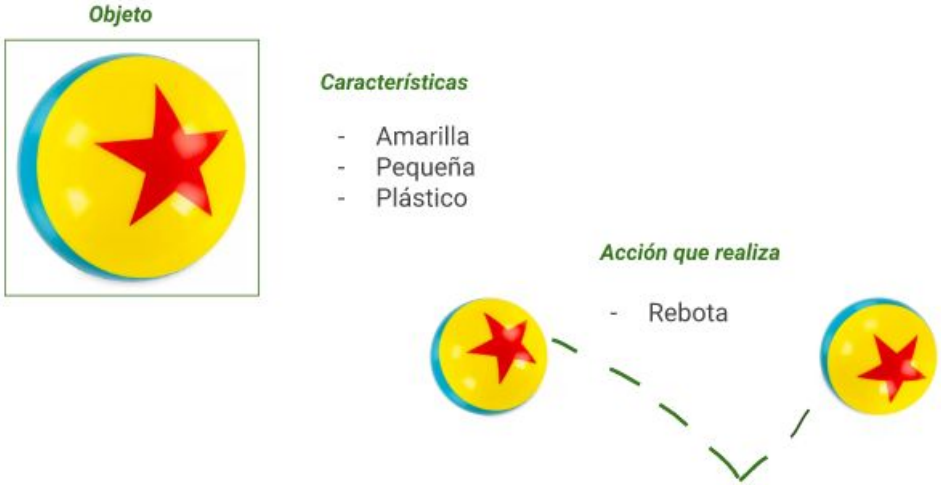
- Paradigma de programación.
- Forma de estructurar un programa, que se basa en empaquetar comportamientos y propiedades similares en objetos individuales.



Orientación a objetos aplicada en la vida cotidiana

Se busca hacer la relación con lo que entendemos por "objeto" en la vida cotidiana.

Ejemplo: Se muestra el objeto "Pelota".



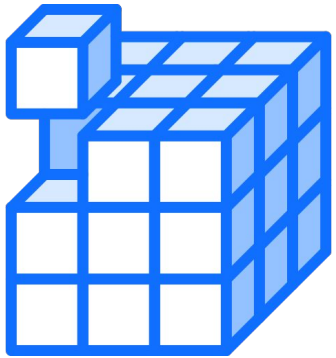
- El molde que permite crear una pelota es lo que se conoce como **Clase**.
- **Atributo**: características que pertenecen al objeto.
- **Método**: acciones que puede realizar el objeto.

Importancia de la orientación a objetos en programación

- Permite resolver problemas.
- Hace un nexo entre una problemática real y su resolución, mediante una codificación sencilla y fácil de entender.
- No solamente permite resolver un problema mediante un objeto aislado, sino que también establece relaciones entre distintos objetos.

Programación estructurada

Es otro paradigma de programación, que se centra principalmente en la construcción de un programa mediante **bloques de subrutinas**, las cuales son básicamente **funciones que realizan tareas específicas**.



¿En qué se diferencia la programación orientada a objetos (POO) y la programación estructurada?

{desafío}
latam_

/* Clases, atributos estáticos y objetos */

Clases

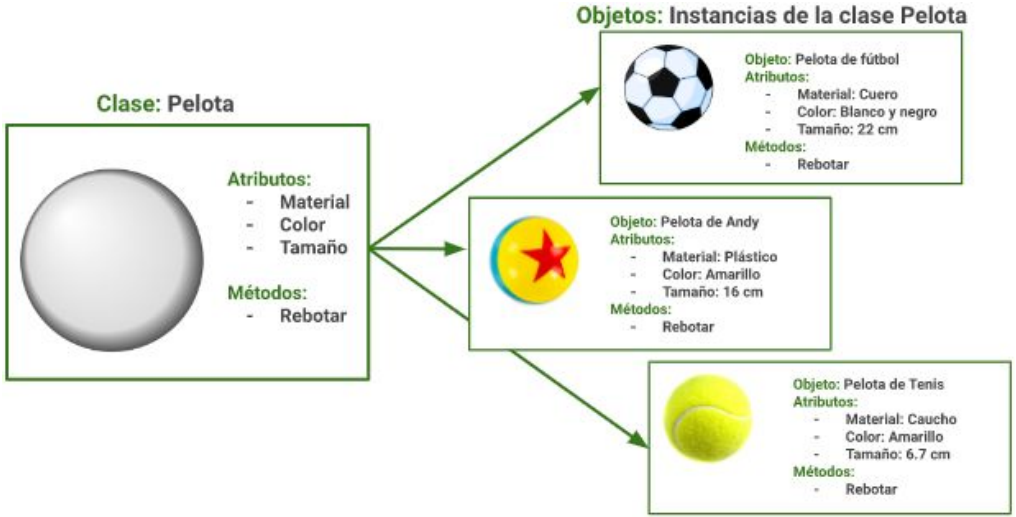
- Código que define qué contiene y qué hace un objeto de un tipo de dato, el cual corresponde al nombre de la clase.

- Por esta razón, muchas veces se hace la analogía diciendo que la clase corresponde a un “plano” o “molde” que permite fabricar objetos de un tipo específico.

- Si quieres profundizar más sobre qué son las clases y cómo trabajar con ellas en Python, puedes revisar la documentación oficial en este enlace.

Clases

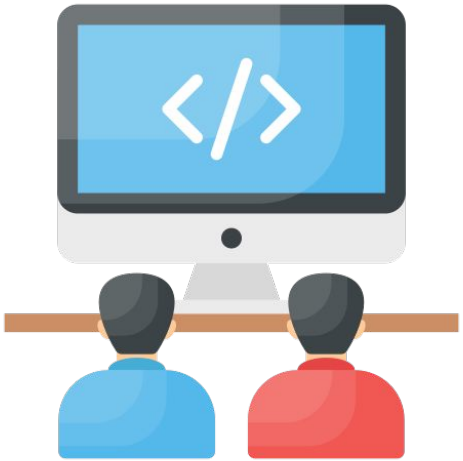
En la siguiente imagen, puedes ver que la **clase Pelota** constituye el molde para crear tres objetos diferentes. Cada **objeto**, o instancia de la clase pelota, tiene sus propios valores para los **atributos**, donde todos tienen la capacidad de rebotar, dado por el método “rebotar” definido en la **clase**.



Tipos de clases

Públicas

Cualquiera puede acceder a ellas.



No abstractas

Implementan los métodos que definen.

Objetos

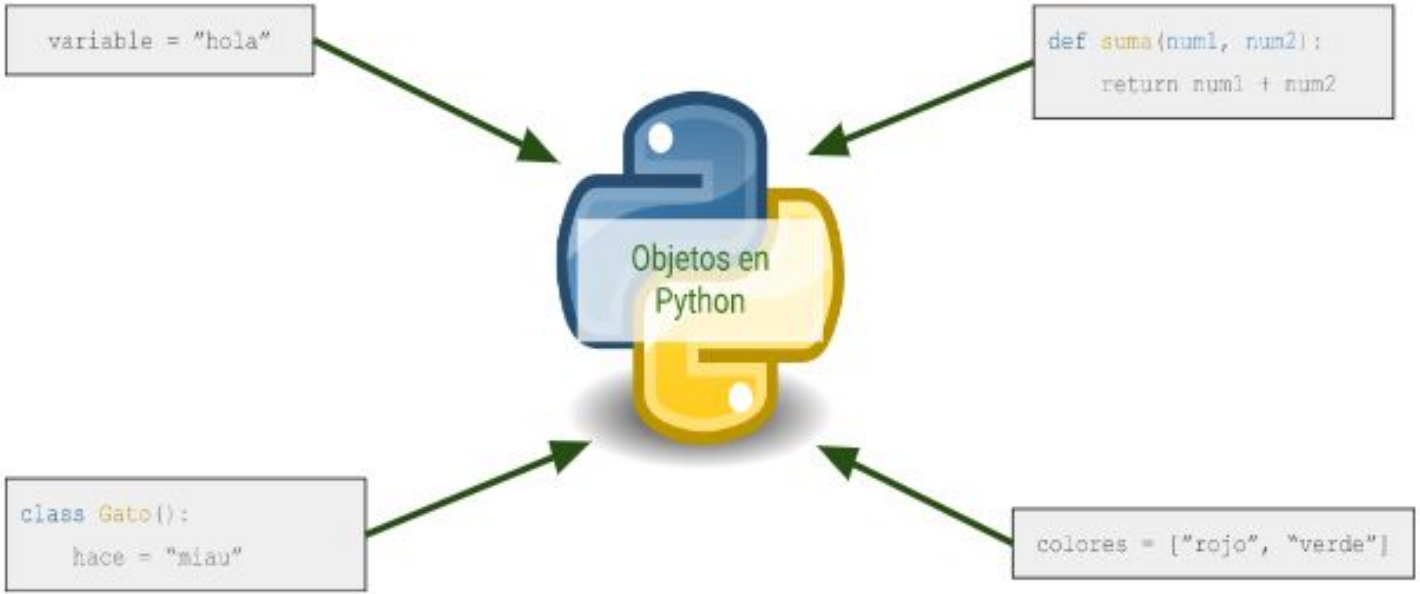
- Conjunto de **características** y **acciones** que afectan a estas, que están dadas por una estructura definida. Esta definición está dada por una clase específica.

- En Python, **todo es un objeto** y todo objeto tiene un tipo asociado, dado por la clase a la cual pertenece, por lo cual, decimos que un objeto es la instancia de una clase.

- Algunas de estas clases son nativas de Python y corresponden principalmente a los:
  - tipos de datos básicos *(bool, float, str, int)*
  - estructuras de datos *(list, dict, tuple)*
  - entre otros *(function, class)*

Objetos

Ejemplo



Si quieres profundizar más sobre qué son y cómo trabajar con objetos en Python, puedes revisar la documentación oficial en este enlace.

Atributos

Contenedor de un valor o de un conjunto de valores, definido dentro de una clase que adquiere un tipo de dato según el valor que se le asigne.

Es decir, **un atributo es análogo a una variable de Python**, pero que en este caso se define dentro de una clase, a diferencia de una variable cualquiera de Python que no necesita definirse dentro de una clase. Por lo tanto, para acceder al atributo o modificar su valor, **se debe hacer por medio de la clase**.

Tipos de atributos

Estáticos o de clase

Pertenecen a la definición de la clase como tal, por lo que es posible acceder a ellos o asignarles un valor, **directamente desde la clase**, sin necesidad de generar un objeto o instancia.

No estáticos o de instancia

No pertenecen a la definición de la clase como tal, por lo que para acceder a ellos o asignarles un valor **se requiere primero crear un objeto** (o instancia) de la clase.

Características	Atributos de clase	Atributos de instancia
Puede contener un dato o conjunto de datos	□	□
Su tipo de dato está dado por el valor asignado	□	□
Se define dentro de una clase	□	□
Se puede leer su valor sin crear una instancia u objeto de la clase	□	✗

Diferencia

clase - objeto - instancia

- **Clase:** corresponde al conjunto de atributos y métodos que permiten definir un objeto.

- **Objeto:** corresponde al conjunto de datos y métodos definidos por la clase a la cual pertenece, en una instancia específica de ella. Por ello, normalmente se tratan los términos objeto e **instancia** como sinónimos.

¿Cómo definir una clase con atributos estáticos?

En Python

Ejemplo:

```
class Pelota():
```

```
class Pelota():
    pass
```

```
class Pelota():
    forma = "redondeada"
```

- En Python, la definición de la clase está dada por la palabra reservada class.
- Luego, se escribe el nombre de la clase, el cual por convención se escribe en formato PascalCase (todo el texto junto, separando palabras por mayúscula en la primera letra).
- A continuación del nombre se debe añadir paréntesis redondos de apertura y cierre ‘()’, y finalmente, se debe escribir el símbolo de dos puntos ‘:’

¿Cómo definir una clase con atributos estáticos?

En Python

Ejemplo:

```
class Pelota():
```

```
class Pelota():
      pass
```

```
class Pelota():
      forma = "redondeada"
```

- El código que pertenece a la clase se escribe en la línea siguiente, por lo general con una indentación de 4 espacios.
- Si se quiere definir una clase sin atributos ni métodos, simplemente se debe escribir la palabra reservada “pass”.

¿Cómo definir una clase con atributos estáticos?

En Python

Ejemplo:

```
class Pelota():
```

```
class Pelota():
    pass
```

```
class Pelota():
    forma = "redondeada"
```

- Si se quiere definir atributos estáticos para la clase, se debe escribir cada atributo en el bloque indentado, siguiendo la misma convención que para los nombres de variables y funciones en Python (snake_case).
- Como buena práctica, se les asigna a estos atributos un valor que luego no será modificado, ya que debe ser un atributo común a todas las instancias.

Ejercicio guiado

"Definir la clase medicamento"

Definir la clase medicamento

Trabajas como programador para una cadena de farmacias que desea desarrollar un software para manejar su stock de medicamentos.

Como primera entrega, debes definir la clase que permite crear objetos de tipo **Medicamento**, los que tienen un descuento de 5% y un IVA de 18%.

Definir la clase medicamento

Solución

Paso 1

Definir la clase Medicamento

```
class Medicamento():
```

Paso 2

Agregar atributo “descuento” con valor 0.05

```
class Medicamento():
    descuento = 0.05
```

Paso 3

Agregar atributo “IVA" con valor 0.18

```
class Medicamento():
    descuento = 0.05
    IVA = 0.18
```

{desafío} latam_

¿Cómo instanciar objetos a partir de una clase?

Ejemplo:

Para instanciar o crear un objeto en Python a partir de una clase, se debe escribir el nombre de esta, seguido de paréntesis redondos de apertura y cierre '()' y hacer la asignación en una variable, como se observa a continuación:

```
pelota_de_andy = Pelota()

# Salida: "" (valor del atributo color)
print(pelota_de_andy.forma)

# archivo pelota.py
class Pelota():
    forma = "redondeada"

# archivo objetos.py
from pelota import Pelota


pelota_de_andy = Pelota()
```

¿Cómo instanciar objetos a partir de una clase?

Ejemplo:

La instancia se crea al momento de escribir el nombre de la clase seguido de los paréntesis (**`Pelota()`**), pero si no se hace la asignación en una variable, no se podrá hacer uso de esa instancia. El objeto o instancia de la clase, tiene la capacidad de hacer referencia a atributos de la clase, utilizando la sintaxis de "punto" (.): **`instancia.atributo`**.

```
pelota_de_andy = Pelota()

# Salida: "" (valor del atributo color)
print(pelota_de_andy.forma)

# archivo pelota.py
class Pelota():
    forma = "redondeada"

# archivo objetos.py
from pelota import Pelota


pelota_de_andy = Pelota()
```

¿Cómo instanciar objetos a partir de una clase?

Ejemplo:

Si la clase que se desea utilizar está definida en un módulo, también es posible hacer uso de ella, para lo cual primero se debe importar la clase desde este módulo (en este caso, en el archivo **pelota.py**) al espacio principal de trabajo (archivo **objetos.py**)

```
pelota_de_andy = Pelota()

# Salida: "" (valor del atributo color)
print(pelota_de_andy.forma)
```

```
# archivo pelota.py
class Pelota():
    forma = "redondeada"

# archivo objetos.py
from pelota import Pelota


pelota_de_andy = Pelota()
```

Ejercicio guiado

"Crear una instancia de Medicamento"

{desafío} latam_

Crear una instancia de Medicamento

Continuemos con el ejercicio guiado anterior

Desde la cadena farmacéutica, ahora solicitan que cada vez que se inicie el programa se debe generar una nueva instancia de un **Medicamento**.

Considera que el programa principal se ejecuta desde un archivo diferente a donde se ha definido la clase.

Crear una instancia de Medicamento

Solución

Paso 1

Importar la clase Medicamento desde el módulo medicamento (archivo medicamento.py), en el archivo actual (programa.py).

```
# archivo programa.py
from medicamento import
Medicamento
```

Paso 2

Instanciar clase Medicamento y almacenar la instancia en una variable.

```
primer_medicamento = Medicamento()
```

{desafío} latam_

¿Cuál es la diferencia entre clase, atributo e instancia?

{desafío} latam_

Próxima sesión...

- *Explica el concepto de método de una clase haciendo la distinción con el concepto de comportamiento de un objeto.*
- *Identifica los principios de abstracción y encapsulamiento de acuerdo al paradigma de orientación a objetos.*

{desafío} latam_

{desafío}
latam_

Academia de
talentos digitales