Módulo 5 – Programación Orientada a Objetos

Lección 1: Programación Orientada a Objetos

1. Video 1: Introducción a la POO

Orientación a Objetos.

* Conceptos básicos

Clase – Es un conjunto de Objetos que comparten una estructura, comportamiento y semántica común – Representación abstracta de un concepto

Objeto – Es una instancia a una clase

* Principios

Abstracción – apegado a la encapsulación, se refiere al proceso mediante el cual somo capaces de escoger la implementación de un objeto que contenga o modele las propiedades y acciones indetificada en el problema para la clase de objetos en cuestión

Encapsulamiento – consiste sen ocultar la representación y el estado de las propiedades de un objeto de manera que estos puedan ser consultados y modificados únicamente a través de las operaciones definidas para la clase – Se separa en su interfaz parte publica y su implementación generalmente privada.

Herencia – Cuando se aplica el proceso de abstracción. En ocasiones no se trabaja con objetos aislados, cada objeto puede estar relacionado con uno o mas objetos. En ocasiones podemos describir nuevos objetos a partir de como estos difieren en características y comportamiento de objetos ya existentes

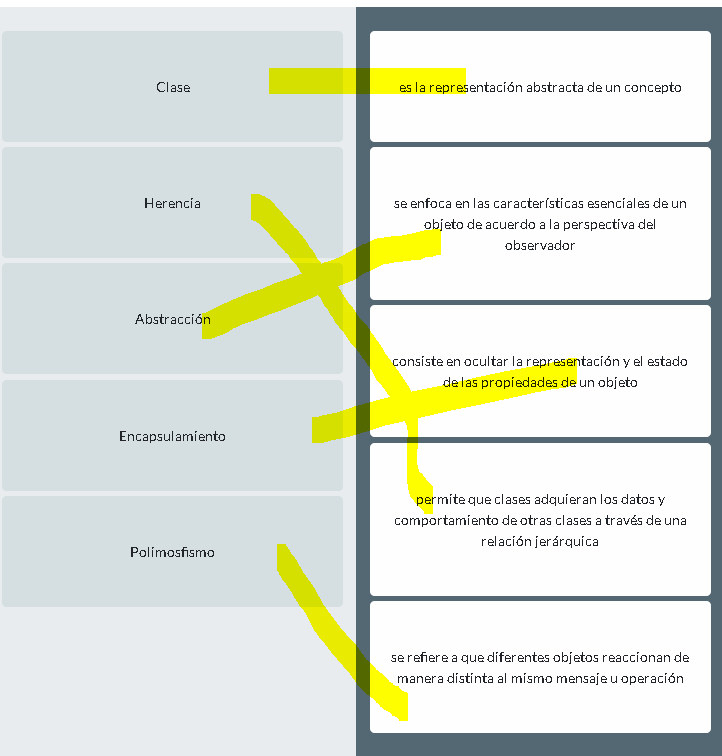
Clase hija hereda de la clase padre (clase base y clase derivada)

Polimorfismo – se refiere a la cualidad de tener más de una forma, se refiere al hecho que una misma op puede tener distintos comportamientos. Diferentes objetos reaccionan de una manera distinta al mismo mensaje u operación

1. Actividad 1

ANALIZA Y  
RELACIONA

¡Hola! Excelente ritmo de formación estás teniendo, aquí aplicarás lo visto en la introducción a la Programación Orientada a Objetos. Así podrás aplicar estos conceptos en las soluciones que plantees a problemas de la vida real. En este caso te solicitamos que relaciones cada concepto que aparece del lado izquierdo con su respectiva definición que te ofrecemos del lado derecho. Para ello, arrastra las opciones de izquierda a derecha según corresponda.



1. Video 2: Definición de clases y Notación UML

Modelar clases y sus relaciones

Rational Software Corporation – UML: Unified Modeling language – Propuesto como estandar en OMG

UML 2.0 🡪 es un language

* Diseño de Software
* Comunicación de procesos de software o negocio
* Captura de detalles sobre un sistema para sus requerimientos o análisis
* Documentación de un sistema existente, proceso u organización

Catalógo de diagramas dividos en:

* Estructurales:
  + Diagrama de clases
  + Diagrama de objetos
  + Diagrama de compuestos
  + Diagrama de componentes
  + Diagrama de despliegue
* De comportamiento
  + Diagrama de actividades
  + Diagrama de caso de uso
  + Diagrama de estado
  + Diagrama de secuencia
  + Diagrama de colaboración

Diagramas de Clase:

<Nombre de la clase>

<atributos>

<Operaciones o métodos> 🡪 Comportamiento de los objetos

Primitivos: int, float, bool

Estructurados: arreglos y registros, o otras clases

Ejemplo:

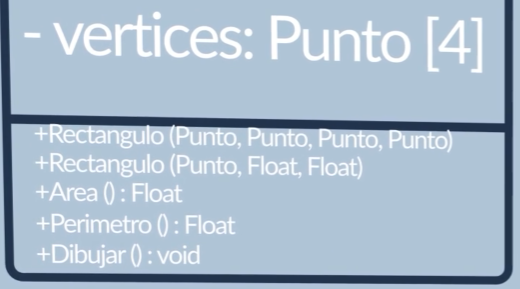


Atributo público puede ser utilizado desde fuera de una clase

Atributo privado, solamente en operaciones dentro de la clase

Atributo protegido, es similar a uno privado con la salvedad que es visible a otras clases relacionadas por herencia.

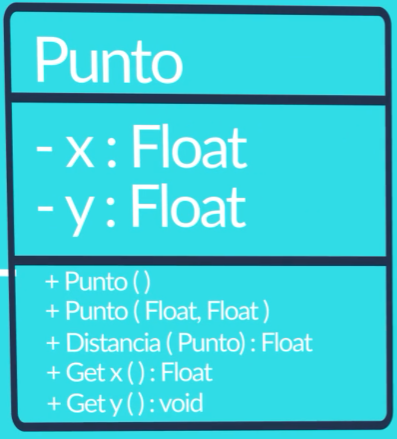
Ej:



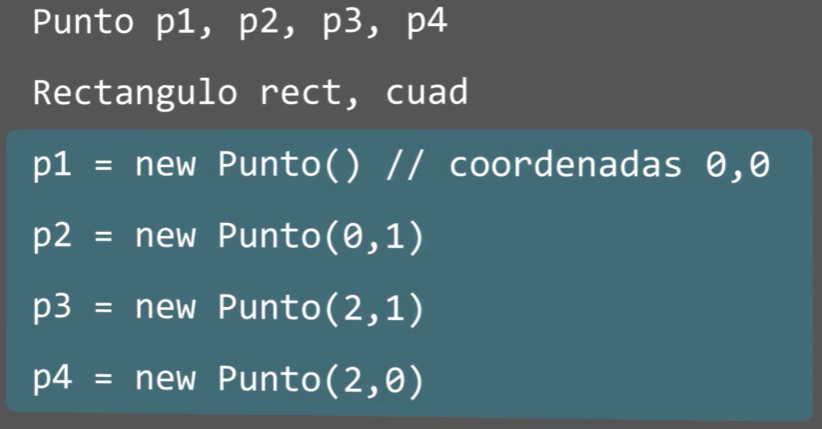


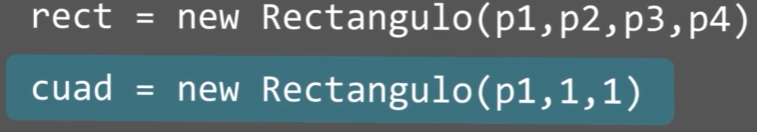
Lenguaje UML

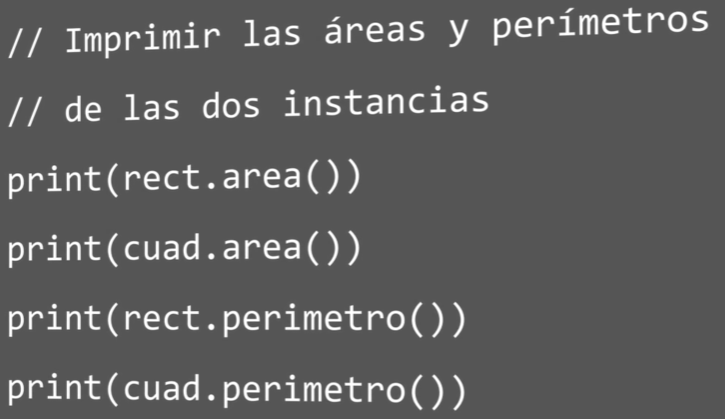


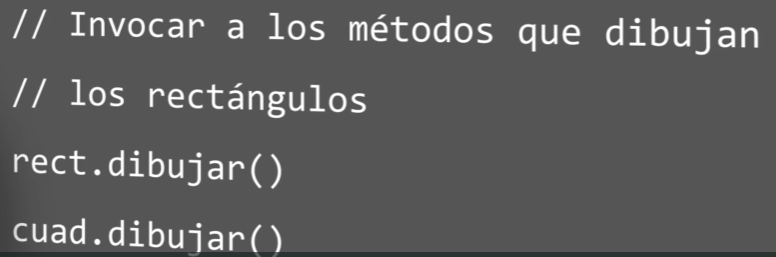
 

Pseudocódigo

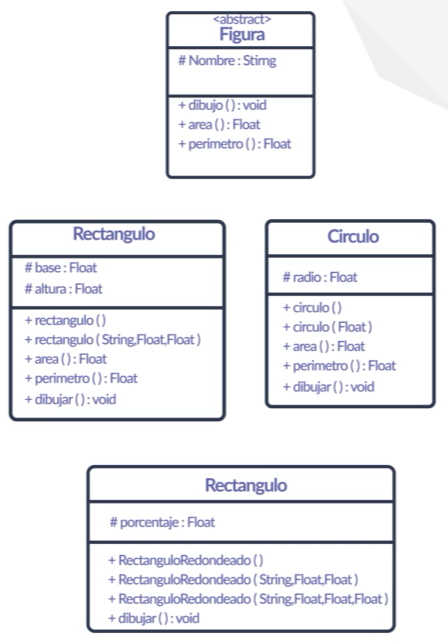








Ejemplo de herencia



1. Actividad 2

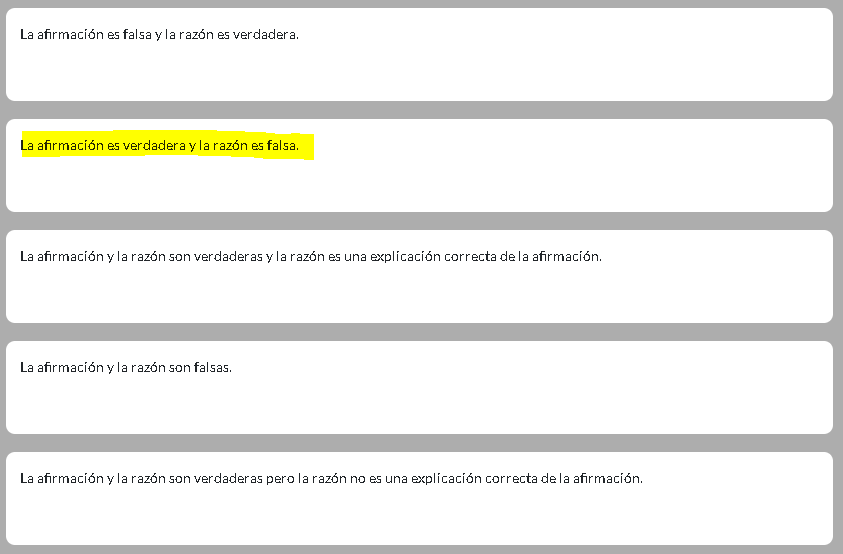
PRUEBA TUS  
CONOCIMIENTOS

¡Qué bien que estás aquí! En esta oportunidad vamos a consolidar lo visto en relación a la definición de clases en el pseudolenguaje y la Notación UML. A continuación, presentamos un texto que se divide en una afirmación y una razón que la sustenta. Te invitamos a leer el texto y escoger la opción correcta según lo estudiado hasta ahora.

Afirmación: los diagramas de clase en UML que son aquellos que capturan los detalles sobre las entidades que existen en un sistema y las relaciones entre éstas

...porque...

Razón: en este diagrama se modela la interacción entre objetos en un sistema.



Lección 2: Clases y Objetos en Python

1. Clases, atributos y métodos

CÁPSULA DE  
CONOCIMIENTO

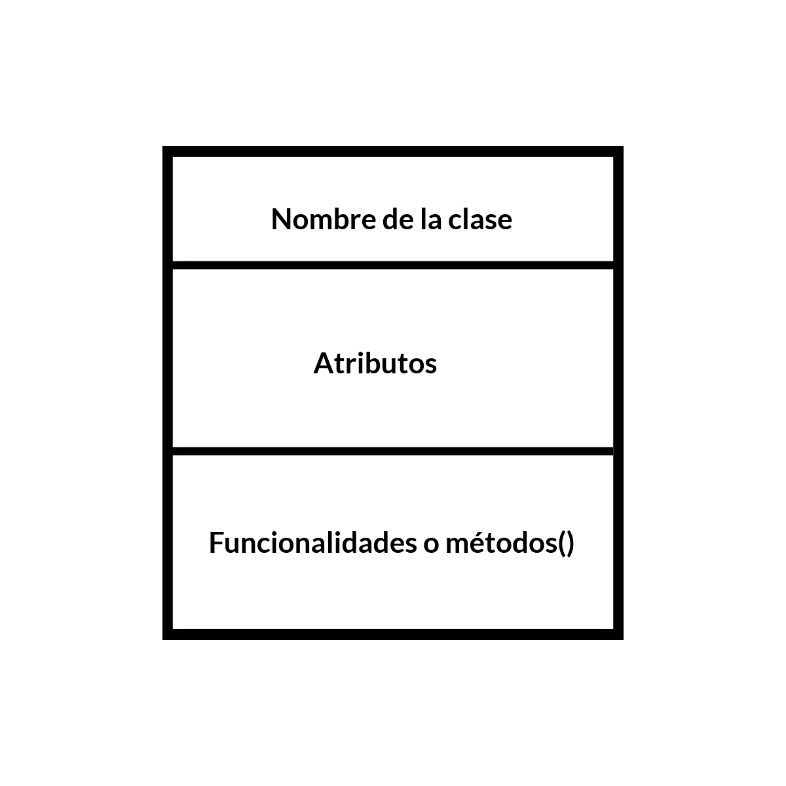
Clases en Python (class)

¡Enhorabuena! Excelente que estemos de nuevo juntos para seguir conociendo las ventajas de Python como lenguaje de programación.

Python, al igual que la gran mayoría de los lenguajes de programación, provee una serie suficientemente variada de tipos de datos integrados, tales como: dict, list, set, entre otros. Ahora bien, en ocasiones nos vemos en la necesidad de crear nuestros propios tipos de datos, adaptados al contexto de las soluciones que estamos planteando en nuestros códigos; como por ejemplo, para nuestro portafolio los tipos de datos Criptomonedas, Billeteras, Jugadores. Todo esto es posible hacerlo en Python a través de la definición de clases y objetos, junto con algunos mecanismos tales como abstracción, encapsulamiento y herencia. Así, a continuación vamos a conocer los conceptos que ofrece este lenguaje en el contexto de la programación orientada a objetos (POO), la cual centra su atención en la creación de objetos que contienen los datos o atributos y funcionalidades o métodos juntos, en una misma entidad denominada clase.

Una clase es una representación abstracta de un concepto; por ejemplo: animales, vehículos o criptomonedas. Las clases se componen de atributos y métodos; por ejemplo para los atributos podemos tener nombre, modelo, nomenclatura; para los métodos indicarEspecie, mostrarColor, darNomenclatura. Luego, un objeto es cada una de las instancias de una clase; por ejemplo miMascota, carro, bitcoin.

Así, una clase es una plantilla genérica de un objeto. La clase proporciona variables iniciales de estado (donde se guardan los atributos) e implementaciones de comportamiento (métodos). En la siguiente figura podemos observar la estructura general de una clase:



Las clases nos dan la posibilidad de crear estructuras de datos más complejas. Por ejemplo crearemos una clase Criptomoneda, para la cual definiremos tres atributos: nombre, saldo y cotización; así como los métodos que nos permitan actualizar y conocer el nombre de una criptomoneda, su saldo y su cotización.

Para crear nuestra clase comenzamos con la palabra class seguida del nombre que le damos a la clase, en este caso, Criptomoneda; "object" es una variable especial en python que utiliza la herencia para crear una nueva clase en python.

class Criptomoneda(object):

def \_\_init\_\_(self, nombre, saldo, cotizacion):

self.nombre = nombre

self.saldo = saldo

self.cotizacion = cotizacion

def indicarNombre(self, nombre):

self.nombre=nombre

def indicarCotizacion(self, cotizacion):

self.cotizacion=cotizacion

def indicarSaldo(self, saldo):

self.saldo=saldo

def mostrarNombre(self):

return self.nombre

def imostrarCotizacion(self):

return self.cotizacion

def mostrarSaldo(self):

return self.saldo

def calcularSaldo(self, moneda):

if moneda== “USD”:

return self.saldo\*self.cotizacion

else:

return self.saldo

Cuando creamos una nueva criptomoneda, necesitamos inicializarla con un nombre y un saldo. El método \_\_init\_\_ (método especial para las funciones que forman parte de una clase) es una función especial en Python que inicializa la clase con los atributos que nosotros le pasemos. Por ejemplo: si corremos el código bitCoin = Criptomoneda("BTC", 0.0) , el método \_\_init\_\_ es llamado con los valores bitCoin, "BTC" y 0.0 en las variables self, nombre y saldo, respectivamente.

La variable self es una instancia de la clase y es bueno aclarar que no es una palabra reservada de python, cualquier etiqueta utilizada como primer parámetro tendría el mismo valor; esto quiere decir que cada vez que declaremos un método en Python, tenemos que agregarle la variable self para que cuando sea invocado el método, Python pase el objeto instanciado y opere con los valores actuales de esa instancia.

También en estas líneas almacenamos en los atributos nombre, saldo y cotización los valores de los parámetro nombre (self.nombre = nombre), saldo (self.saldo = saldo) y cotizacion (self.cotizacion = cotizacion).

Podemos definir métodos que necesitemos para mostrar e interactuar con los contenidos de las instancias. El método indicarNombre toma una instancia de la clase Criptomoneda y devuelve el nombre de la misma. Lo mismo pasa con el método calcularSaldo. Nuevamente, necesitamos el parámetro self para que la función sepa con que instancia de Criptomoneda trabajar, y un parámetro que nos indicará en qué moneda calcularemos el saldo (criptomoneda o USD).

A continuación vamos de declarar dos objetos de la clase Criptomoneda, a saber: Bitcoin y Ethereum:

Bitcoin = Criptomoneda(“Bitcoin”,0.234,6435.321)

Ethereum = Criptomoneda(“Ethereum”,0.6734,467.563)

Una vez que viste cómo declarar clases y objetos en Python, te invitamos a seguir adelante a fin de aplicar estos conocimientos en actividades prácticas ¡Suerte y éxito!

1. Ejemplo: Clases y herencia en Python
2. Taller: Interactuando con clases y objetos

APRENDE  
HACIENDO

Interactuando con clases y objetos Python

¡Enhorabuena! Excelente momento para aplicar los conceptos vistos acerca de las clases y objetos en Python, en esta oportunidad vamos a manipular estos elementos instanciando clases e interactuando con objetos. Vamos a considerar la clase definida en el lenguaje Python y que denominados Criptomonedas, como tenemos a continuación:

class Criptomoneda(object):

def \_\_init\_\_(self, nombre, saldo, cotizacion):

self.nombre = nombre

self.saldo = saldo

self.cotizacion = cotizacion

def indicarNombre(self, nombre):

self.nombre=nombre

def indicarCotizacion(self, cotizacion):

self.cotizacion=cotizacion

def indicarSaldo(self, saldo):

self.saldo=saldo

def mostrarNombre(self):

return self.nombre

def imostrarCotizacion(self):

return self.cotizacion

def mostrarSaldo(self):

return self.saldo

def calcularSaldo(self, moneda):

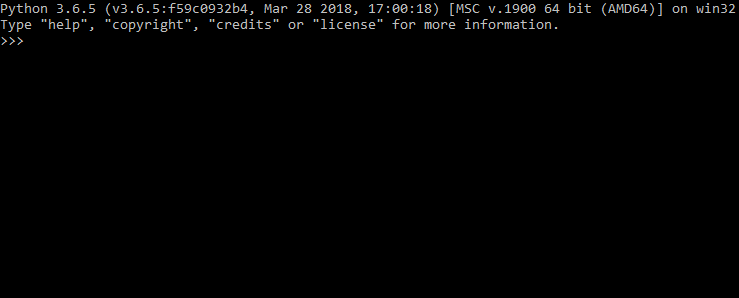
if moneda== “USD”:

return self.saldo\*self.cotizacion

else:

return self.saldo

En este punto vamos a interactuar en consola con la clase Criptomoneda. Para hacer esto debemos abrir la consola y colocarnos en el directorio donde tenemos guardado nuestro archivo Criptomonedas.py. Una vez que estamos en el directorio indicado, escribimos "python" para abrir el Shell interactivo de Python.



Lo primero que hacemos es importar la clase Criptomoneda del archivo Criptomonedas.py como se indica a continuación:

>>>from Criptomonedas import Criptomoneda

Luego, creamos una instancia de la clase Criptomoneda y la llamamos ethereum de la siguiente forma

>>> ethereum = Criptomoneda ("ETH ", 0.0, 454.543)

Posteriormente, escribimos por pantalla el nombre de la criptomoneda y el saldo inicial con la siguiente instrucción:

>>>print(ethereum.mostrarNombre(),ethereum.calcularSaldo("USD"))

En lo sucesivo, sigamos creando otras instancias de la clase Criptomoneda para afianzar nuestros conocimientos. A fin de seguir probando, nos mantenemos en la consola de Python y utilizamos el archivo denominado Criptomonedas.py donde tenemos definida la clase Criptomoneda descrita anteriormente. Ahora hagamos los siguientes ejercicios:

Ejercicio 1:

>>>from Criptomonedas import Criptomoneda

>>>bitcoin = Criptomoneda ("BTC", 0.34, 5000.00)

>>> bitcoin.mostrarNombre()

"BTC"

>>>bitcoin.calcularSaldo("USD")

1700.0000000000002

>>>bitcoin.indicarSaldo(0.5)

>>>bitcoin.calcularSaldo("USD")

2500.0

Ejercicio 2:

>>>from Criptomonedas import Criptomoneda

>>>litecoin = Criptomoneda ("LTC", 1.67, 124.00)

>>> litecoin.mostrarNombre ()

"LTC"

>>>litecoin.calcularSaldo("USD")

207.07999999999998

>>>litecoin.indicarCotizacion(154.45)

>>>litecoin.calcularSaldo("USD")

257.93149999999997

Ejercicio 3:

>>>from Criptomonedas import Criptomoneda

>>>ripple = Criptomoneda ("XPR", 34.93, 0.4785)

>>>ripple.mostrarNombre ()

"XPR"

>>>ripple.indicarNombre("XRP")

>>>ripple.mostrarNombre ()

"XRP"

>>>ripple.calcularSaldo("USD")

16.714005

¡Felicitaciones! Por dar tus primeros pasos en la Programación Orientada a Objetos con Python, ya pusiste en práctica tus conocimientos acerca del manejo de clases y objetos. ¡Sigue adelante!

1. Actividad

PRUEBA TUS  
CONOCIMIENTOS

¡Saludos y bienvenida! Según lo visto acerca de las herramientas de programación orientada a objetos en Python, te animamos a leer las siguientes afirmaciones y determinar cuáles son verdaderas:

