**IINFORME**

[Sidebar Title]

[Sidebars are great for calling out important points from your text or adding additional info for quick reference, such as a schedule.

They are typically placed on the left, right, top or bottom of the page. But you can easily drag them to any position you prefer.

When you’re ready to add your content, just click here and start typing.]

SOBRE

**POGRAMACIÓN**

**ORIENTADA**

A

**OBJETOS**

EN

**PYTHON**

4-06-2022

**UNIVERSIDAD TÉCNICA**

**PARTICULAR DE LOJA**

**Introducción**

A diferencia de otros lenguajes como Java o C++, Python es un lenguaje de programación de nivel alto que se emplea para diseñar soluciones de todo tipo. Se trata, frente a otros lenguajes, de un código interpretable, por lo que no es necesario recopilarlo para ejecutar correctamente los programas creados en Python, pues éstos son procesados automáticamente por el procesador utilizando un lenguaje llamado intérprete, sin necesidad de "traducirlo" al lenguaje de la máquina.

Python es un lenguaje fácil de leer y trazar debido a su gran semejanza con los lenguajes humanos. Conjuntamente, es un lenguaje de código abierto, multiplataforma y por lo tanto gratuito, que permite desarrollar software sin restricciones. Python ha ganado popularidad con el tiempo debido a su claridad y amplia gama de posibilidades, especialmente en los últimos años, ya que es más fácil de formar con inteligencia artificial, big data, aplicación mecánico y ciencia de datos, y muchos otros trabajos florecientes en el campo de desarrollo.

Con el fin de conseguir profundizar en el concepto de Python, en este documento hemos recopilado una importante cantidad de datos sobre los aspectos y atributos como: sus clases, constructores y objetos; conociendo su función de un lenguaje de programación que se puede utilizar con facilidad, así el usuario no posea experiencia alguna en el campo de la programación.

**Desarrollo:**

Como sabemos, la programación orientada a objetos es un paradigma. De esta forma, te enseña a probar y estudiar métodos para solucionar las necesidades de los sistemas informáticos basados ​​en interacciones basadas en objetos. a base a esto, Taltal menciona lo siguiente “Hasta el momento hemos trabajado con el paradigma de programación estructurada o procedural. Este resultado de mucha utilidad en ámbitos muy diversos, aunque en algunas cuestiones puede ser complejo de aplicar o, incluso, complicarnos la tarea que queremos realizar”. (Guagliano, 2019)

La programación orientada a objetos (también llamada OOP, del inglés Object‐Oriented Program‐ ming) es un método para organizar programas que reúne muchas de las ideas vistas hasta ahora. Al igual que las funciones en la abstracción de datos, los objetos imponen barreras de abstracción entre el uso y la implementación de los datos. Al igual que los diccionarios y funciones de despacho, los objetos responden a peticiones que otros objetos le hacen en forma de mensajes para que se comporte de determinada manera. Los objetos tienen un estado interno local al que no se puede acceder directamente desde el entorno global, sino que debe hacerse por medio de las operaciones que proporciona el objeto.

A efectos prácticos, por tanto, los objetos son datos abstractos. El sistema de objetos de Python proporciona una sintaxis cómoda para promover el uso de estas técnicas de organización de programas. Gran parte de esta sintaxis se comparte entre otros lenguajes de programación orientados a objetos. (Pérez, 2022)

Los elementos se pueden considerar como los materiales necesarios para diseñar y programar un sistema, mientras que las propiedades se pueden considerar como las herramientas necesarias para construir un sistema a partir de estos materiales. Entre los principales elementos podemos encontrar:

1. **Clases.**

Una clase es un modelo en el que se construye un objeto. Como se mencionó anteriormente, las clases y los objetos se utilizan para crear sus propios tipos de datos. Una clase es un tipo de datos definido por el usuario y la instanciación de una clase implica la creación de un objeto de ese tipo. Las clases y los objetos se consideran los componentes principales de Python, un lenguaje de programación orientado a objetos.

Una clase se dirige mediante una declaración class seguida de un nombre genérico para el objeto.

class ClassName:  
    statements

La definición de clase comienza con la palabra clave class y className es el nombre de la clase. Tenga en cuenta que los nombres de las clases siguen las mismas reglas que los nombres de las variables de Python. Es decir, los nombres de clase solo pueden comenzar con letras o guiones bajos y solo pueden contener letras, números o guiones bajos. Además, de acuerdo con PEP 8 (Guía de estilo de programación de Python), se recomienda que los nombres de las clases estén en mayúsculas.

A continuación, definimos la clase Person (persona). Esta clase actualmente no contiene nada más que una declaración de ruta. Según la documentación de Python:

La instrucción pass no hace nada. Se puede usar cuando la instrucción se requiere sintácticamente pero el programa no requiere ninguna acción.

class Person:  
    pass

Se crea una instancia, objeto simplemente haciendo lo siguiente:

María =  Person()

Esto significa que ha creado un nuevo Objeto de tipo Persona, María. Tenga en cuenta que, para crear un objeto, solo necesita escribir el nombre de la clase seguido de algunos paréntesis.

Puede determinar cuánto espacio María es una clase y ocupa en la memoria escribiendo: print María. En ese caso, obtendrás algo como esto:

< \_\_main\_\_.Person instance at 0x109a1cb48 >

1. **Constructores.**

Python tiene los llamados métodos "mágicos". Es decir, llamar a estos métodos directamente no es normal en el ámbito de tu programa, por lo que es un método que tiene un nombre especial y solo se llama en determinadas circunstancias.

El método \_\_init\_\_ actúa como constructor. Es decir, inicializa un conjunto de atributos y ejecuta el código que definió cuando creó el objeto de clase. Esto se llama "\_\_init\_\_" y tiene dos guiones bajos. Al principio y al final de la palabra init, ya tenemos la sintaxis adecuada para que Python pueda tomarlo como un método "mágico" y reconocer que debe ejecutarse al crear una instancia de la clase.

En lugar de hacer lo siguiente para inicializar el objeto, el constructor hace lo siguiente:

f =  FooBar()

F.init()

Al colocar este código, el objeto inicializó sus atributos solo usando:

f =  FooBar()

Como puede ver, esto ahorra mucho dinero al codificar y es útil si olvida inicializar el objeto.

Crear un constructor en Python es muy fácil. Todo lo que tiene que hacer es definir el método y agregar lo que necesita hacer al crear el objeto. Veamos un ejemplo.

Class FooBar:

def \_init\_(self):

self.somever = 42

Aquí se puede ver el método definido. Puede ver que el método que debe ejecutarse tiene un atributo somevar definido. Veamos un ejemplo de cómo se ejecuta un objeto de esta clase:

f = FooBar()

f.somevar

42

Aquí, podemos ver que definir un objeto f de tipo FooBar inicializa el objeto y acceder al atributo somevar devuelve el valor 42 definido en el constructor.

Ahora que pasa si queremos que los atributos se inicialicen dinámicamente, reescribamos el método de la siguiente forma:

class FooBar:

Def \_init\_(self, value=42):

Self.somevar = value

Es posible que deba heredar de la superclase, pero es posible que deba anular el constructor para eso. Veamos el siguiente ejemplo para entender de qué se trata.

Class Bird:

def \_init\_(self):

self.hungry = 1

def eat(self):

if self.hungry:

print ‘Aaaah…’

self.hungry = 0

else:

print ‘No, gracias’

Self.somevar = value

1. **Objetos**

La clase en sí es solo un modelo que te ayuda a crear un objeto en particular. Una clase es un razonamiento abstracto de un objeto, y un objeto es su encarnación. La acción de crear un objeto se llama instanciar una clase, y esa instancia consiste en asignar una clase como valor a una variable. Hay diferentes tipos de objetos:

*Objetos clase*

Los objetos de clase admiten dos tipos de operaciones: exploración de atributos e instanciación.

class MyClass:

"""A simple example class"""

i = 12345

def f(self):

return 'hello world'

Por lo tanto, MyClass.i y MyClass.f son referencias de atributo válidas y devuelven un entero y un objeto de función, respectivamente. También puede asignar atributos de clase para que pueda cambiar el valor de MyClass. Atributos válidos. Devuelve el documento "SimpleExampleClass" asociado a la clase.

La creación de instancias de clases utiliza la notación de funciones. Supongamos que un objeto de clase es una función sin parámetros que devuelve una nueva instancia de la clase. Ejemplo (para la clase anterior):

X = MyClass()

*Objetos instancia*

La única operación comprendida por un objeto de instancia es una referencia de atributo. Hay dos tipos de nombres de atributos válidos: atributos de datos y métodos.

Los atributos de datos corresponden a las "variables de instancia" de Smalltalk y las "variables miembros" de C ++. No es necesario declarar atributos de datos. Algo se asigna a una variable local de la misma manera que se crea inicialmente una variable local. Por ejemplo, si x es una instancia de MyClass que se creó anteriormente, el siguiente código generará el valor 16 y no dejará rastro.

x.counter = 1

while x.counter < 10:

x.counter = x.counter \* 2

print(x.counter)

del x.counter

El otro tipo de atributo de instancia es un método. Un método es una función que "pertenece" a un objeto. En Python, el término método no se limita a instancias de clase. También puede incluir métodos en otros tipos de objetos. Por ejemplo, los objetos de lista tienen métodos para agregar, insertar, eliminar, clasificar, etc. Sin embargo, la siguiente descripción utiliza el término método para referirse únicamente a métodos de objetos de instancia de clase, a menos que se indique lo contrario.

Objetos método

El método es llamado luego de ser vinculado:

x.f()

En el ejemplo de MyClass, esto devuelve la cadena 'helloworld'. Sin embargo, no es necesario llamar al método de inmediato. x.f es un objeto de método que se puede guardar y recuperar más tarde. por ejemplo:

xf = x.f

while True:

print(xf())

Esto seguirá imprimiendo hola mundo infinitamente.

**Conclusión**

Como hemos visto, Python es un lenguaje de programación multiplataforma que hace que sea sorprendentemente fácil desarrollar aplicaciones en cualquier sistema operativo. Debido a su simplicidad y alto poder de procesamiento de datos, muchas tecnologías se adaptan muy bien a Python. Esto, sin duda, resurge el lenguaje en el lugar de trabajo donde cada vez más empresas buscan expertos en Python.

# Referencias

Guagliano, C. (2019). Programación en Python. Creative Andina. <https://books.google.es/books?hl=es&lr=&id=y1yzDwAAQBAJ&oi=fnd&pg=PA5&dq=programacion+orientada+a+objetos,+python&ots=G0CSzJYD3i&sig=DGwH-Wf1CY7FinatAiizE_kbq2U#v=onepage&q&f=false>

Pérez, R. (2022). *Programación Orientada a Objetos.* IES Doñana. <https://pro.iesdonana.org/apuntes/programacion-orientada-a-objetos-apuntes.pdf>

uniwebsidad. (2011). *5.2. Programación Orientada a Objetos (Python para principiantes)*. <https://uniwebsidad.com/libros/python/capitulo-5/programacion-orientada-a-objetos>

Clases — documentación de Python - 3.10.4. (s. f.). python. <https://docs.python.org/es/3/tutorial/classes.html>

Castellano, E. S. P. L. (s. f.). CÃ3mo funcionan las clases y objetos en Python. Programación en Castellano. <https://programacion.net/articulo/como_funcionan_las_clases_y_objetos_en_python_1505>