Conceptos de la Programación Orientada a Objetos
Clases, objetos y métodos – Sintaxis y nomenclaturas

No es la finalidad de este curso ahondar en la Programación orientada a objetos, pero es necesario conocer algunos términos y principios para comprender la sintaxis al invocar métodos en Python.

Clase

Una clase es un tipo de dato definido por el usuario. Podemos definirlo en primera instancia como una plantilla o modelo para crear ciertos objetos. Esta plantilla contiene la información, características y capacidades que tendrá el objeto creado a partir de ella.

Las clases se crean con la palabra reservada ***class,*** y se denominan con algún nombre que comience con una letra mayúscula.

En ella, definimos sus

- ***atributos*** (variables que definirán sus características), y
- ***métodos*** (operaciones que es capaz de ejecutar). Los métodos se definen con la palabra reservada ***def***

Ejemplo: Creando una clase

```
class Cuenta():
    num = ''
    sucursal = ''
    tipo = ''
    saldo = 0
    cbu = ''

    def modifica_saldo(self, monto):
        self.saldo += monto

    def mostrar_saldo(self):
        print('\nEstado de CUENTA:')
        print(self.tipo,'N°',self.num,'-',self.sucursal)
        print('CBU:',self.cbu)
        print('Saldo: $', self.saldo,'\n')
```

Todos los objetos creados a partir de una clase estarán agrupados en esa misma clase. Crear un objeto a partir de una clase se llama **instanciar.**

Objetos

Cuando pensamos en objetos (también llamados **instancias**), nos referimos a entidades que tienen un comportamiento, un estado, almacenan información y pueden realizar tareas.

No necesariamente todos los objetos derivan de una clase en particular. Existen objetos sin que los hayamos creado. De hecho en Python, casi todo es un objeto.

Al crear un objeto, éste hereda la “forma” y las funcionalidades que hemos definido en la clase.

Ejemplo: creando un objeto

Vamos a crear un objeto de la clase Cuenta

```
17    c1 = Cuenta()
```

Estamos creando una variable, y en ella almacenamos un objeto de clase `Cuenta` vacía, sin datos en ella.

Ejemplo: modificando los atributos de un objeto

En este ejemplo, estamos asignando datos a nuestra cuenta: estamos modificando sus atributos iniciales.

Para ello, invocamos al objeto en cuestión (`c1`), seguido de un punto y el nombre del atributo que queremos modificar.

```
18    c1.num = '1245'
19    c1.sucursal = '021'
20    c1.tipo = 'Caja de Ahorros'
21    c1.saldo = 10000
22    c1.cbu = '4588745211548965000025'
```

Ejemplo: invocando los métodos de un objeto

A continuación, vemos cómo ejecutar los métodos de un objeto.

Al igual que hicimos con los atributos, para invocar a un método de un objeto, debemos especificar el objeto (c1), seguido de un punto, y el nombre del método que queremos ejecutar (si es necesario con los parámetros que requiera)

```
24    c1.mostrar_saldo()
25    c1.modifica_saldo(-4800)
26    c1.mostrar_saldo()
```

Aquí vemos cómo mostrar la información de la cuenta en la línea 24, a continuación modificamos el saldo en la línea 25 (suponemos que se realizó un pago por $4800 y lo enviamos como dato), y volvemos a mostrar la información de la cuenta para ver los cambios.

Podrás encontrar el código de este capítulo en el archivo clases.py.

```
1   class Cuenta():
2       num = ''
3       sucursal = ''
4       tipo = ''
5       saldo = 0
6       cbu = ''
7
8       def modifica_saldo(self, monto):
9           self.saldo += monto
10
11      def mostrar_saldo(self):
12          print('\nEstado de CUENTA:')
13          print(self.tipo,'Nº',self.num,'-',self.sucursal)
14          print('CBU:',self.cbu)
15          print('Saldo: $', self.saldo,'\n')
16
```

```
16
17    c1 = Cuenta()
18    c1.num = '1245'
19    c1.sucursal = '021'
20    c1.tipo = 'Caja de Ahorros'
21    c1.saldo = 10000
22    c1.cbu = '458874521154896500025'
23
24    c1.mostrar_saldo()
25    c1.modifica_saldo(-4800)
26    c1.mostrar_saldo()
```