IIC1103 - Sección 2

Clase 19: Programación orientada a objetos

Prof. Francisca Cattan

Universidad Católica de Chile

25 de octubre, 2022

Clases y Atributos

Vamos a aprender nuevos conceptos que nos ayudarán a crear nuestros propios objetos.

- Clases
- Atributos
- Constructores

Tipos de datos ... ¿o clases?

Hemos visto distintos tipos de datos en Python:

$$int, float, bool, string, list$$

Podemos saber la clase de un objeto con el comando type()

Tipos de datos ... ¿o clases?

Hemos visto distintos tipos de datos en Python:

int, *float*, *bool*, *string*, *list*

Podemos saber la clase de un objeto con el comando type()

```
1  a = 4.5
2  b = "4.5"
3  c = [10,125,325]
4  d = 4
5  e = 7/8 + 19 / 20 >= 2
6
7  print(type(a)) # <class 'float'>
8  print(type(b)) # <class 'str'>
9  print(type(c)) # <class 'list'>
10 print(type(d)) # <class 'int'>
11 print(type(e)) # <class 'bool'>
```

Programación orientada a objetos

- Vamos a aprender un nuevo tipo de dato llamado **Clase** en Python.

Programación orientada a objetos

- Vamos a aprender un nuevo tipo de dato llamado **Clase** en Python.

Sintaxis definición de Clase

```
1 class NombreClase:          # nombre de la clase
2     codigo_clase            # codigo de la clase
3     codigo_clase
4     ...
5 codigo_fuera_de_clase       # termina la definicion de la clase
```

Sintaxis creación de Clase

```
1 objeto = NombreClase()
2 print(type(objeto))
```

```
__main__.NombreClase
```

Programación orientada a objetos

Por ejemplo, podemos crear una clase **Estudiante** y un objeto perteneciente a esa clase.

```
1 class Estudiante:
2     pass # por ahora vamos a pasar
3
4 estudiante = Estudiante()
5 print(type(estudiante))
```

```
__main__.Estudiante
```

Programación orientada a objetos

Por ejemplo, podemos crear una clase **Estudiante** y un objeto perteneciente a esa clase.

```
1 class Estudiante:
2     pass # por ahora vamos a pasar
3
4 estudiante = Estudiante()
5 print(type(estudiante))
```

```
__main__.Estudiante
```

Aunque por supuesto, esta clase parece que no sirve para nada... por ahora.

Programación orientada a objetos

A cada **objeto** le podemos agregar **atributos** (variables) y ver o modificar sus valores.

Sintaxis creacion de atributos

```
1 variable = NombreClase()
2 variable.atributo = valor
3 print(variable.atributo)
```

Programación orientada a objetos

A cada **objeto** le podemos agregar **atributos** (variables) y ver o modificar sus valores.

Sintaxis creacion de atributos

```
1 variable = NombreClase()
2 variable.atributo = valor
3 print(variable.atributo)
```

Los atributos son los componentes que tiene ese objeto de esa clase.

Programación orientada a objetos

- **Pensemos juntos:** ¿Que atributos podria tener una clase **Estudiante**?

```
1 class Estudiante:
2     pass
```

Programación orientada a objetos

- **Pensemos juntos:** ¿Que atributos podria tener una clase **Estudiante**?

```
1 class Estudiante:
2     pass
```

- Nombre
- Apellidos
- Ramos actuales
- Puntaje
- Nota
- ... ¿profe favorita?

Programación orientada a objetos

Creemos varios estudiantes de la clase y agreguemos atributos.

```
estud1 = Estudiante()
estud1.nombre = "Sasha"
estud1.nota = 5.5
estud1.edad = 24

estud2 = Estudiante()
estud2.nombre = "Armin"
estud2.nota = 6.2
estud2.edad = 20

estud3 = Estudiante()
estud3.nombre = "Jean"
estud3.nota = 5.1
estud3.edad = 21
```

Programación orientada a objetos

Creemos varios estudiantes de la clase y agreguemos atributos.

```
1  estud1 = Estudiante()
2  estud1.nombre = "Sasha"
3  estud1.nota = 5.5
4  estud1.edad = 24
5
6  estud2 = Estudiante()
7  estud2.nombre = "Armin"
8  estud2.nota = 6.2
9  estud2.edad = 20
10
11 estud3 = Estudiante()
12 estud3.nombre = "Jean"
13 estud3.nota = 5.1
14 estud3.edad = 21
```

Parece que esto es mucho codigo copiado/pegado/modificado, ¿o no?

Programación orientada a objetos

Constructor es un método que se llama cuando se crea un nuevo objeto de la clase.

Sintaxis constructor

```
1 class NombreClase:        # nombre de la clase
2     # metodo constructor
3     def __init__(self, par1, par2, ...)
4         self.atributo1 = par1
5         self.atributo2 = par2
6         .
7         .
```

El constructor __init__ se ejecuta cuando se crea un objeto, entregando los valores iniciales.

```
1 # esto ejecuta automaticamente __init__(par1, par2, ...)
2 var = NombreClase(val1, val2, ...)
```

Programación orientada a objetos

self: variable especial que representa al objeto recien creado, que es una instancia de una clase. Nos permite referirnos al objeto único para agregar o modificar sus atributos.

```
1 # defino el tipo de dato (clase) "estudiante"
2 class Estudiante:
3     # comienzo con el constructor
4     def __init__(self, nombre, nota):
5         self.su_nombre = nombre
6         self.su_nota = nota
7         self.su_graduacion = 2026
```

```
1 # creo estudiantes
2 estud1 = Estudiante("Sasha", 5.5)
3 estud2 = Estudiante("Armin", 6.2)
4 estud3 = Estudiante("Jean", 5.1)
```

Programación orientada a objetos

self: gracias a esto, ahora puedo consultar o modificar los atributos.

```
1  class Estudiante:
2      # comienzo con el constructor
3      def __init__(self, nombre, nota):
4          self.su_nombre = nombre
5          self.su_nota = nota
6          self.su_graduacion = 2026
7
8  # creo 2 estudiantes
9  estud1 = Estudiante("Sasha", 5.5)
10 estud2 = Estudiante("Armin", 6.2)
11
12 # imprimo y cambio sus atributos
13 print(estud2.nota)    # >> 6.2
14 estud1.nota = 7.0
15 print(estud1.nota)    # >> 7.0
```

Programación orientada a objetos

Ventaja de tener un constructor:

- Lógica para asignar atributos dentro de la definición de la clase.
- Obligamos asignación de atributos.
- Podemos ejecutar códigon cada vez que se crea el objeto.

Programación orientada a objetos

metodos: las clases pueden tener sus propios metodos. Vamos a crear una clase Libro, que tendra atributos relacionados a un libro y un metodo que le permita avanzar de pagina.

```
1  class Libro:
2      def __init__(self, nombre, autor, genero):
3          self.su_nombre = nombre
4          self.su_autor = autor
5          self.su_genero = genero
6          self.pagina_actual = 15
7
8      def leer(self, paginas_leidas):
9          self.pagina_actual += int(paginas_leidas)
10         print('Ahora voy en la pagina',self.pagina_actual)
11
12 # creo un libro y llamo a su metodo
13 mi_libro = Libro("Harry Potter y el caliz de fuego", "J.K.
   Rowling","Fantasia")
14 mi_libro.leer(121)
15 mi_libro.leer(56)
```

Ahora voy en la pagina 136

Ahora voy en la pagina 192

Programación orientada a objetos

- **Pensemos juntos** ¿que otros metodos podríamos crear para la clase **Libro**?
- ¿Qué otros atributos interesantes tienen los libros?
- ¿Que podríamos hacer si tenemos muchos objetos distintos de la clase **Libro**?