

Programación Orientada a Objetos

Clase #16

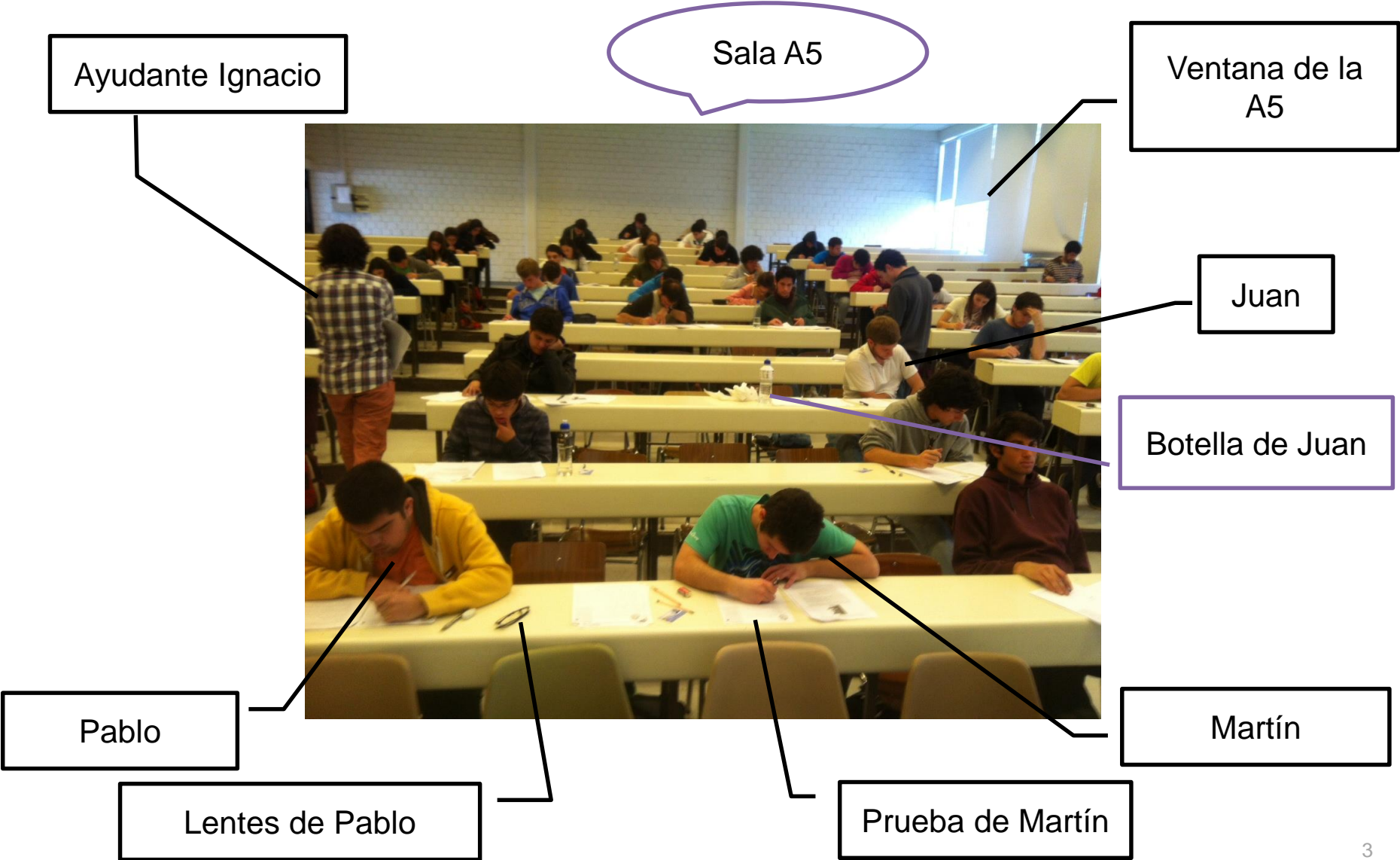
IIC1103 – Introducción a la Programación

Marcos Sepúlveda (marcos@ing.puc.cl)

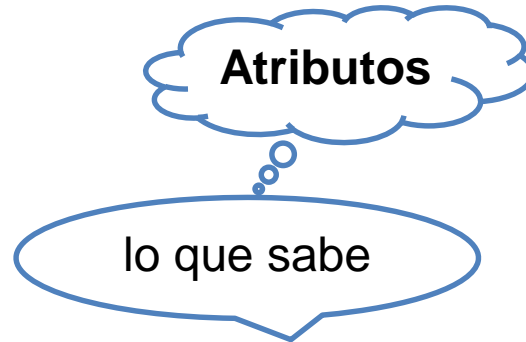
Veremos hoy ...

- ▶ Tipos de datos
- ▶ Objetos
- ▶ Clases
- ▶ Ejercicios

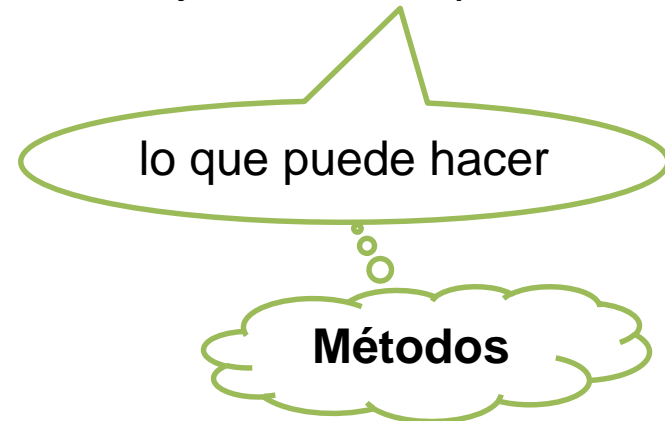
Introducción – todo es un objeto



¿Qué es un objeto?

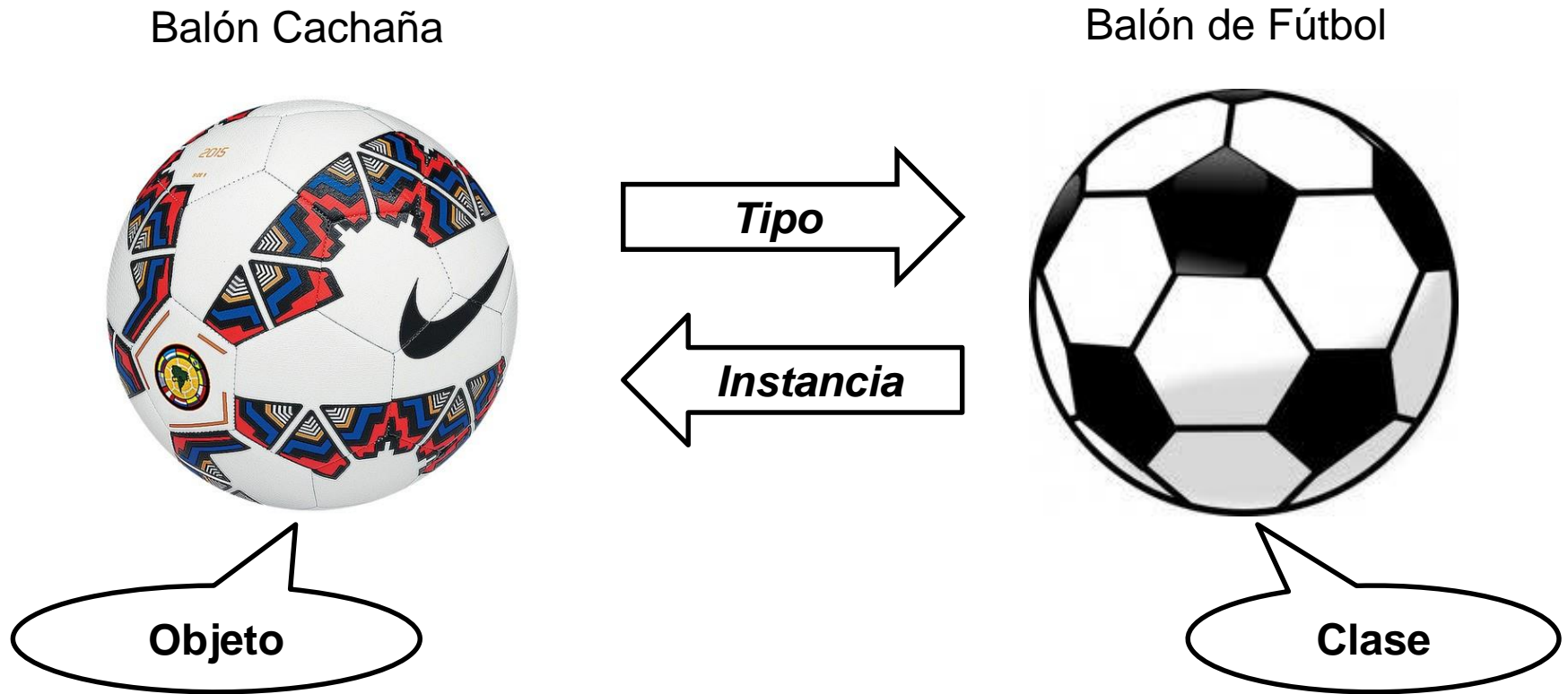


- Un **objeto** es un conjunto de datos y tiene un conjunto de comportamientos



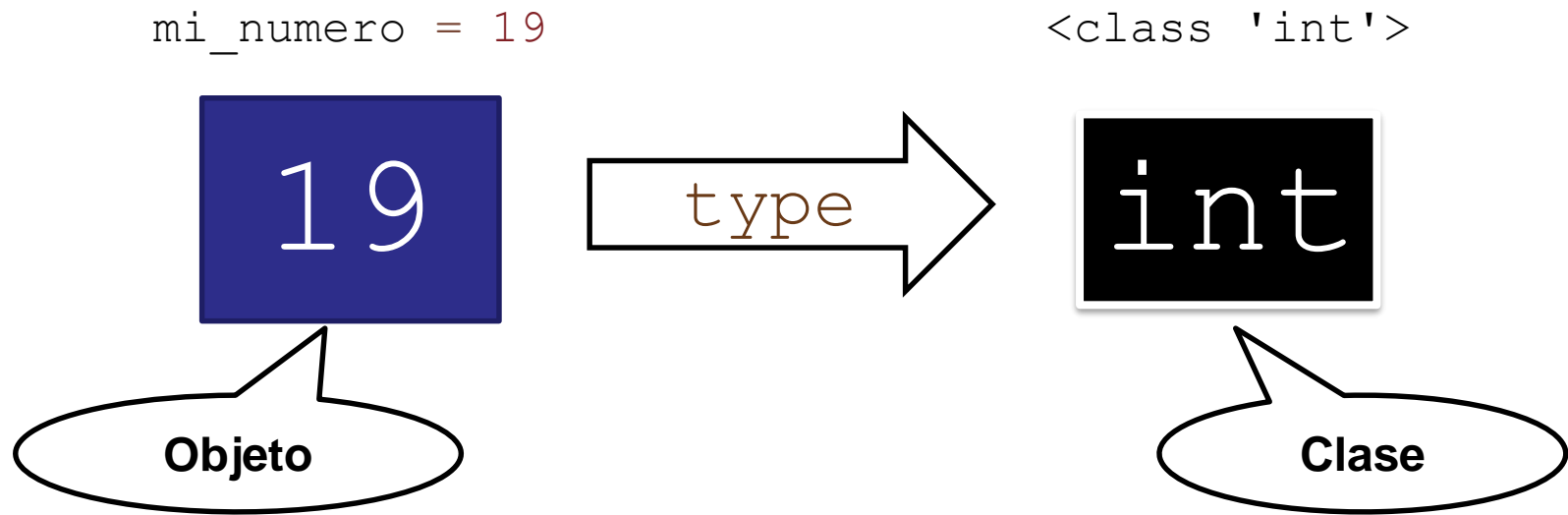
Clases – todo objeto es de un tipo

- Dependiendo de la **clase** a la que pertenezca, el objeto tendrá un **tipo de información** diferente y podrá realizar diferentes **acciones**.



Clases – función `type`

- ▶ La función `type` permite averiguar de qué clase es el objeto que le entregamos como parámetro.



Tipos de datos – básicos y compuestos

- ▶ En Python existen tipos de datos básicos y compuestos definidos por el lenguaje:

- ▶ **Tipos básicos:** enteros, reales, booleanos

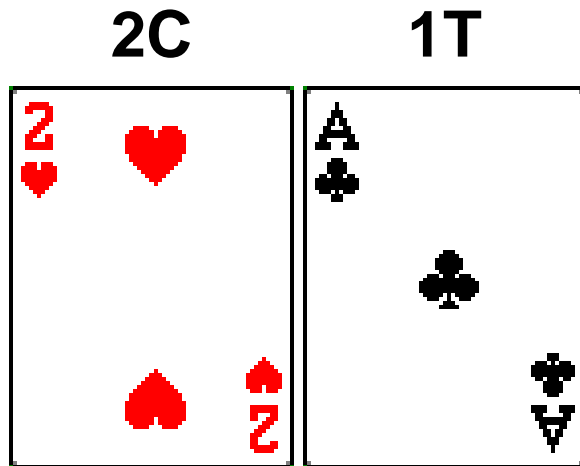
```
>>> mi_numero = 19
>>> print(type(mi_numero))
<class 'int'>
```

- ▶ **Tipos compuestos:** strings, listas, tuplas

```
>>> mi_lista = [1,2,3,4]
>>> print(type(mi_lista))
<class 'list'>
```

Clases – tipos de datos definidos por el usuario

- ▶ Además de los tipos de datos definidos por Python, podemos crear nuestros propios tipos de datos.
- ▶ **Ejemplo:** Una clase que representa las **cartas** de una baraja inglesa



carta

```
pinta  
numero  
  
Mostrar()  
Cambiar()  
Recoger()  
Tirar()
```


Clases – creando una clase

- ▶ Python utiliza la palabra `class` para crear clases de objetos.
- ▶ Como toda sentencia compuesta, está formada por:
 - un ***encabezado*** seguido por dos puntos y
 - un ***cuerpo*** o bloque de sentencias

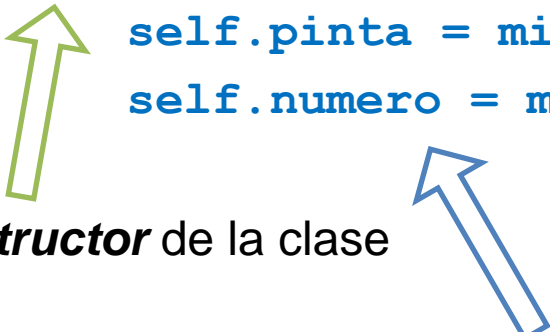
```
class Carta:  
    # bloque de definición de una clase
```

Clases – constructor

- ▶ Cada clase tiene definido un método especial llamado `__init__()`, que le permite controlar cómo se inicializan los atributos de los objetos de una clase.

```
class Carta:
```

```
    def __init__(self, mi_pinta, mi_numero):  
        self.pinta = mi_pinta  
        self.numero = mi_numero
```



- ▶ **Constructor** de la clase
- ▶ Los **atributos** de una clase son las variables que contienen las propiedades de un objeto (lo que sabe).
 - En el constructor, se da valor inicial a los atributos de la clase

Clases – creando objetos / instancias

- ▶ Para crear un objeto de una clase:

```
[variable] = [Nombre_de_Clase] (param1, param2, ...)
```

- ▶ Por ejemplo:

```
#programa principal
```

```
from carta import Carta
```

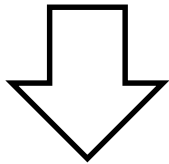
```
#instanciando 2 objetos de la clase carta
```

```
carta_1 = Carta("Trebol", 1)
```

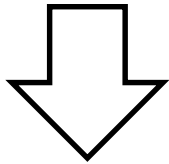
```
carta_2 = Carta("Corazon", 2)
```

Clases – ¿Internamente cómo se crean los objetos?

```
carta_1 = Carta("Trebol", 1)
```



```
Carta.__init__(carta_1, "Trebol", 1)
```



```
def __init__(self, mi_pinta, mi_numero):  
    self.pinta = mi_pinta  
    self.numero = mi_numero
```

Objetos – ¿Cómo se accede a los atributos de un objeto?

- ▶ A través del nombre del objeto, seguido de ".", y el nombre del atributo.

```
#programa principal
from carta import Carta

#instanciando 2 objetos de la clase carta
carta_1 = Carta("Trebol", 1)
carta_2 = Carta("Corazon", 2)

#mostrando sus atributos en pantalla
print(carta_1.pinta, carta_1.numero)
print(carta_2.pinta, carta_2.numero)
```

```
>>>
Trebol 1
Corazon 2
```

Clases – Métodos

- Para describir el comportamiento posible de un objeto de una clase, creamos un **método** (función dentro de la clase), que tiene como primer parámetro a **self**

```
class Carta:
    def __init__(self, mi_pinta, mi_numero):
        self.pinta = mi_pinta
        self.numero = mi_numero

    def CambiarPinta(self, nueva_pinta):
        self.pinta = nueva_pinta

    def CambiarNumero(self, nuevo_numero):
        self.numero = nuevo_numero
```

- **Nota:** las funciones dentro de una clase son llamadas **métodos**

Clases – ¿Cómo se invocan los métodos de una clase?


- ▶ Para invocar a un método de la clase, primero tenemos que crear un objeto y luego llamamos al método.

- ▶ Por ejemplo:

```
#programa principal
from carta import Carta
```

```
#instanciando 2 objetos de la clase carta
carta_1 = Carta("Trebol", 1)
carta_2 = Carta("Corazon", 2)
```

```
#utilizando sus métodos
```



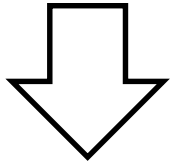
```
carta_1.CambiarPinta("Corazon")
carta_1.CambiarNumero(8)
```

```
#mostrando cómo cambian sus valores
print(carta_1.pinta, carta_1.numero)
```

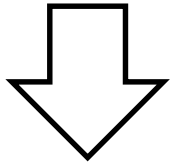
```
>>>
Corazon 8
```

Clases – ¿Cómo funciona la invocación de métodos?

```
carta_1 = Carta("Trebol", 1)  
carta_1.CambiarPinta("Corazon")
```



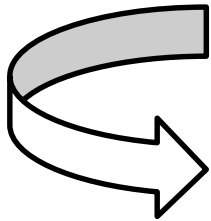
```
Carta.CambiarPinta(carta_1, "Corazon")
```



```
def CambiarPinta(self, nueva_pinta):  
    self.pinta = nueva_pinta
```

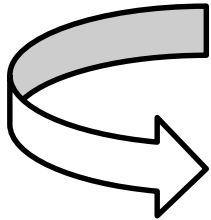

Clases – entendiendo cómo se instancian los objetos

- ▶ Cuando se invoca un método de una instancia, Python se encarga de que el **primer argumento** sea la instancia del objeto que invoca.
- ▶ Este hecho explica por qué **self** es tan importante y por eso mismo debe ser el primer argumento de cada método que se escribe.



```
carta_1 = Carta("Trebol", 1)
```

```
Carta.__init__(carta_1, "Trebol", 1)
```



```
carta_1.CambiarPinta("Corazon")
```

```
Carta.CambiarPinta(carta_1, "Corazon")
```

Objetos – ¿Qué es un objeto?

- Un objeto es una instancia de una clase, y por lo tanto “vive” en la memoria del computador.

```
carta_1t = Carta("Trebol", 1)
print(carta_1t)

<carta.Carta object at 0x100691050>
```

0x100691048	pi
0x100691049	3,1416
0x100691050	
	"Trebol"
0x100691060	1
0x100691065	...
0x100691070	12

Objetos – Diferentes objetos (**incorrecto**)

- En el siguiente fragmento de código, ¿por qué cambia el valor de `carta_1t.numero` ?

```
carta_1t = Carta("Trebol", 1)
carta_2c = carta_1t
carta_2c.CambiarPinta("Corazon")
carta_2c.CambiarNumero(2)
print(carta_1t, carta_1t.pinta, carta_1t.numero)
print(carta_2c, carta_2c.pinta, carta_2c.numero)
```

```
<carta.Carta object at 0x0205EE30> Corazon 2
<carta.Carta object at 0x0205EE30> Corazon 2
```

- Ambas variables apuntan al mismo objeto de la posición `0x0205EE30`

Objetos – Diferentes objetos (correcto)

- ▶ En el siguiente fragmento de código, ¿por qué **NO** cambia el valor de `carta_1t.numero` ?

```
carta_1t = Carta("Trebol", 1)
carta_2c = Carta("Corazon", 2)
carta_2c.CambiarNumero(3)
print(carta_1t, carta_1t.pinta, carta_1t.numero)
print(carta_2c, carta_2c.pinta, carta_2c.numero)
```

```
<carta.Carta object at 0x020AEE50> Trebol 1
<carta.Carta object at 0x020E0A90> Corazon 3
```

- ▶ Las dos variables apuntan a distintos objetos en la posición `0x020AEE50` y en la posición `0x020E0A90`

Ejercicio – creación de clases

- ▶ Una aerolínea considera útil mostrar a sus clientes la hora en los distintos países donde opera. Para ello, te pide que escribas un programa que muestre las horas de distintos países en formato **HH : mm.**

Ejercicio – creación de clases

prueba_paises.py

```
from paises import HoraPais

horaCL = HoraPais("Chile", 14, 43)
horaAR = HoraPais("Argentina", 14, 43)
horaBR = HoraPais("Brasil", 15, 13)
horaMX = HoraPais("Mexico", 11, 43)

print(horaCL.Formatear())
print(horaAR.Formatear())
print(horaBR.Formatear())
print(horaMX.Formatear())
```

paises.py

```
class HoraPais:
    def __init__(self, pais, hh, mm):
        self.pais=pais
        self.hh = hh
        self.mm = mm

    def Formatear(self):
        return self.pais+" > "+str(self.hh)+":"+str(self.mm)
```

```
>>>
Chile > 14:43
Argentina > 14:43
Brasil > 15:13
Mexico > 11:43
```

Ejercicio – alumno

alumno.py

```
class Alumno:
    def __init__(self, nom, ed, n1, n2, n3):
        self.nombre = nom
        self.edad = ed
        self.nota1 = n1
        self.nota2 = n2
        self.nota3 = n3

    def Promedio(self):
        return (self.nota1+self.nota2+self.nota3)/3

    def Mostrar(self):
        print(self.nombre,self.edad)

alumno_1 = Alumno(input("nombre: "),int(input("edad: ")),
                  float(input("nota 1: ")),float(input("nota 2: ")),
                  float(input("nota 3: ")))

alumno_1.Mostrar()
print("Promedio:", alumno_1.Promedio())
```

```
>>>
nombre: Juan
edad: 22
nota 1: 4.5
nota 2: 5.1
nota 3: 6.3
Juan 22
Promedio: 5.3
```