

Programando en Python II

Enero 2023



red.es

Centro de Referencia Nacional en Comercio Electrónico y Marketing

CRN Digital



Barrabés

The Valley

"El FSE invierte en tu futuro"
Fondo Social Europeo

Fecha de la presentación

1. Funciones

- 1.1 Llamadas
- 1.2 Parámetros
- 1.3 Variables
- 1.4 None

2. Estructuras de datos

- 2.1 Listas
- 2.2 Tuplas
- 2.3 Diccionarios
- 2.4 Numpy arrays
- 2.5 Clases



Programando en Python

Funciones



Funciones

¿Qué son?

Son fragmentos de código definidos de manera que se pueden reutilizar y realizan un determinado algoritmo.

Reducen el número total de líneas de tu proyecto y lo hacen más legible!

Las funciones pueden recibir uno o más argumentos y a su vez devolver algún resultado o nada.

Funciones

Llamar a una función

Acción de usar una función.

```
def sumar():  
    print 5 + 10  
  
sumar()
```

Funciones

Predefinidas en librerías

```
>>> abs(-3)↵
3
>>> abs(round(2.45, 1))↵
2.5
>>> from math import sin↵
>>> sin(1)↵
0.8414709848078965
```

Funciones

Definir una función

```
1 def cuadrado(x):  
2     return x ** 2
```

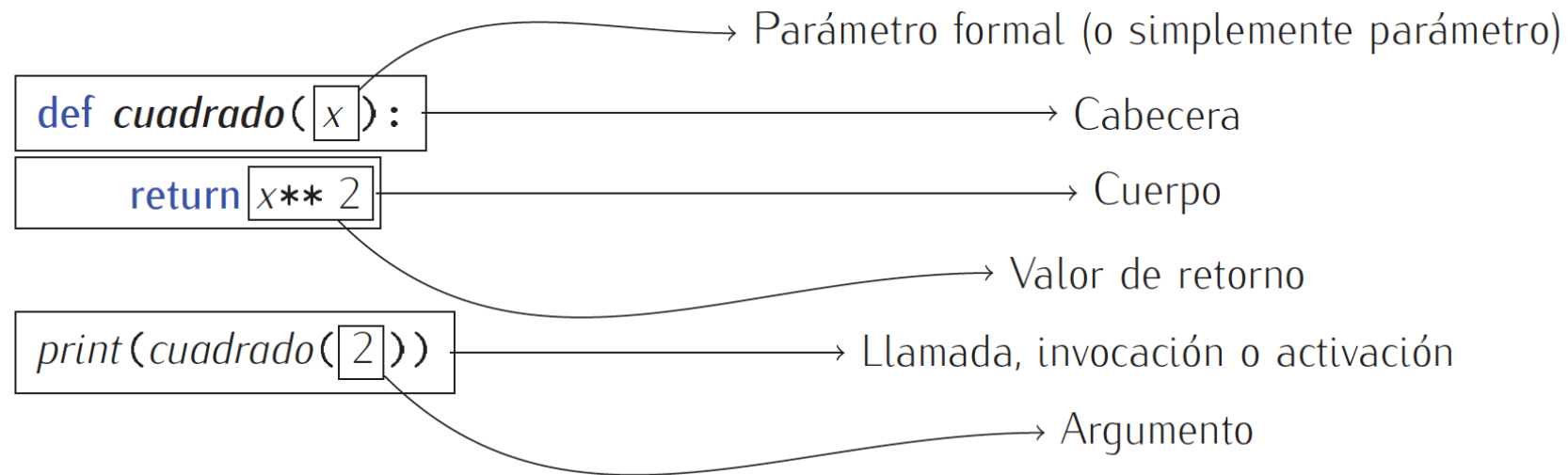
Funciones

Usar una función

```
1 def cuadrado(x):  
2     return x ** 2  
3  
4 print(cuadrado(2))  
5 a = 1 + cuadrado(3)  
6 print(cuadrado(a * 3))
```


Funciones

Definiciones



Funciones

Nombres

Las reglas para dar nombre a las funciones y a sus parámetros son las mismas que las de las variables. Pero cuidado, no se puede usar el mismo nombre para una variable y una función al mismo tiempo.

Funciones

Funciones que utilizan funciones

No hay problema si necesitas utilizar otra función dentro de una función.

```
1 from math import sin
2
3 def xsin(x):
4     return x * sin(x)
```

Funciones

No importa el nombre del identificador

```
1 def cubo(z):  
2     return z ** 3
```

```
1 def cubo(x):  
2     return x ** 3
```

```
1 y = 1  
2 print(cubo(y))
```

Funciones

También puedes utilizar sentencias de control

```
1 def es_mayor_de_edad(edad):  
2     if edad < 18:  
3         resultado = False  
4     else:  
5         resultado = True  
6     return resultado
```

```
1 def es_mayor_de_edad(edad):  
2     if edad < 18:  
3         return False  
4     else:  
5         return True
```

```
1 def es_mayor_de_edad(edad):  
2     if edad < 18:  
3         return False  
4     return True
```

Funciones

Más de un parámetro

```
1 def área_rectángulo(altura, anchura):  
2     return altura * anchura  
3  
4 print(área_rectángulo(3, 4))
```

Funciones

Sin parámetros

```
1 def lee_entero():  
2     return int(input())  
3  
4 a = lee_entero()
```

Funciones

Cómo ordenar un programa para que sea legible

```
1 from math import sqrt
2
3 def cuadrado(x):
4     return x**2
5
6 def suma_cuadrados(vector):
7     suma = 0
8     for elemento in vector:
9         suma += cuadrado(elemento)
10    return suma
11
12 # Programa principal
13 mivector = []
14 for i in range(3):
15     mivector.append(float(input('Dame un número: ')))
16 s = suma_cuadrados(mivector)
17 print('Distancia al origen:', sqrt(s))
```


Funciones

Variables locales y globales

```
1 from math import sqrt
2
3 def área_triángulo(a, b, c):
4     s = (a + b + c) / 2
5     return sqrt(s * (s-a) * (s-b) * (s-c))
6
7 print(área_triángulo(1, 3, 2.5))
8 print(s)
```

None

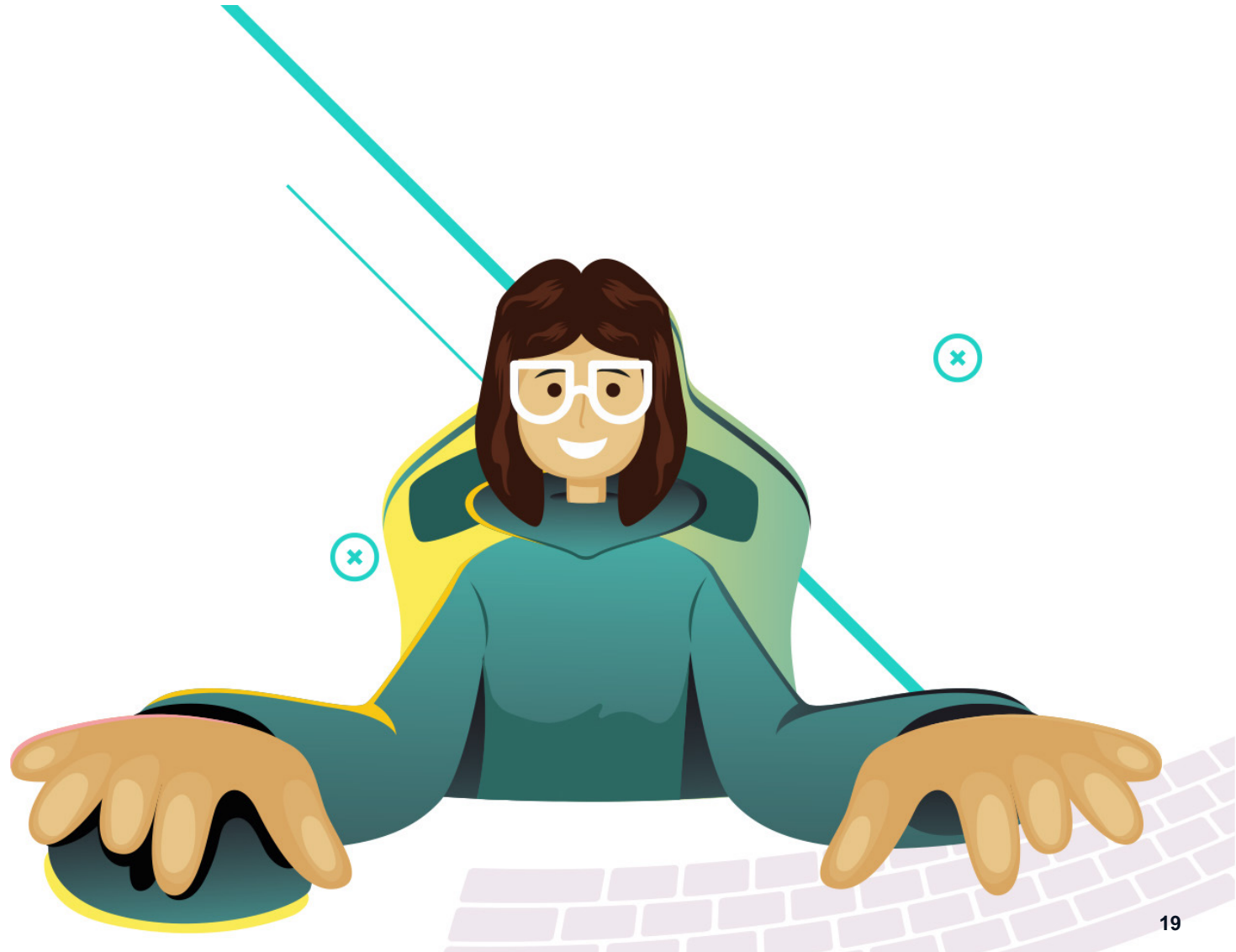
Tipo de datos especial

Significa ausencia de valor.

Ejercicios



Notebook 2



TÍTULO DE LA PRESENTACIÓN

Estructuras de datos



Estructuras de datos

Listas

Las listas son un tipo de datos delimitado por corchetes que pueden contener cualquier tipo de datos. Puede haber listas de enteros, de flotantes, de cadenas,...

Las operaciones con ellos son parecidas a las cadenas. Se pueden concatenar listas, añadir elementos, eliminar,...

```
miLista1 = [3, 5, 1, 0, 3, 5]
```

```
miLista2 = ['pera', 'melon', 'cereza', 'plátano']
```



Estructuras de datos

Listas

```
69 lista1 = ["a", "b", "c"]
70 lista2 = [1, 2, 3]
71
72 lista3 = lista1 + lista2
73 print(lista3)
74
```

```
['fresa', 'uva', 'cereza']
['fresa', 'uva', 'cereza']
```

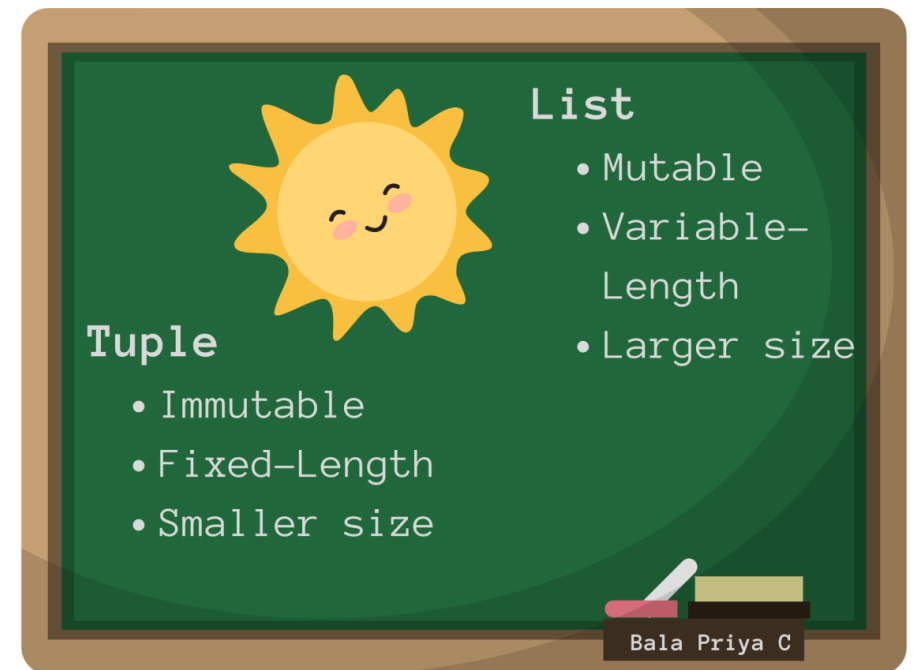
```
myList = ["pan", "leche", "azúcar", 2, 4.6, True, ["Javi", "María"], 99]
print(myList)
myList.remove(2)
print(myList)
```

```
['pan', 'leche', 'azúcar', 2, 4.6, True, ['Javi', 'María'], 99]
['pan', 'leche', 'azúcar', 4.6, True, ['Javi', 'María'], 99]
```

Estructuras de datos

Tuplas

**Son parecidas a las listas pero no se pueden modificar sus valores.
Se definen mediante paréntesis.**



Estructuras de datos

Tuplas

objetos = (7, 'Hola', True, 3.5)

7	'Hola'	True	3.5
0	1	2	3

```
hello.py x
1 name_of_tuple = ('apple', 10, 'bannana', 24)
2 print(name_of_tuple[0])
3 print(name_of_tuple[1])
4 print(name_of_tuple[2])
5 print(name_of_tuple[3])

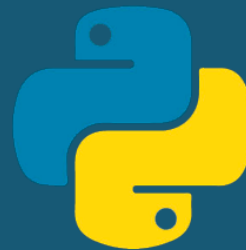
Run: hello x
/home/ahmed/PycharmProjects/function/venv/bin/pyt
apple
10
bannana
24
```


Estructuras de datos

Diccionarios

Correspondencia entre claves y valores.

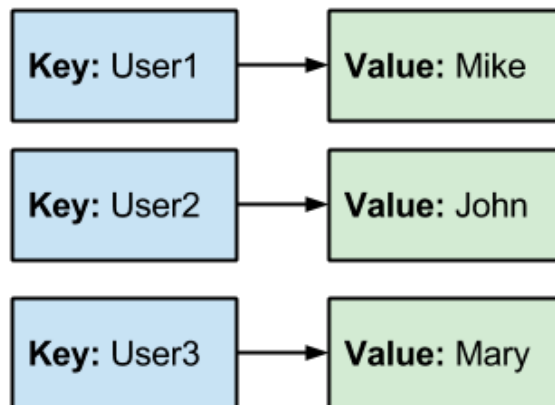
Diccionarios en



```
diccionario = {'nombre': 'Carlos', 'edad': 22, 'cursos': ['Python', 'Django']}
```

Estructuras de datos

Diccionarios



```
32 diccionario = {  
33     "Marca" : "Ford",  
34     "Modelo" : "Mustang",  
35     "Año" : 1964  
36 }  
37 for x in diccionario.values():  
38     print(x)  
39  
40 for x, y in diccionario.items():  
41     print(x, y)  
42
```

```
Mustang  
Ford  
1964  
Modelo Mustang  
Marca Ford  
Año 1964
```

Estructuras de datos

CHULETA	Tupla	Lista	Diccionarios
Definición	<code>mi_tupla = ('texto', 20, 1275.48)</code>	<code>mi_lista = ['texto', 20, 1275.48]</code>	<code>mi_dict = {'clave uno':'texto', 'clave dos':20, 'clave tres':1275.48}</code>
Obtener uno de los valores	<code>print mi_tupla[0]</code> # imprime texto <code>print mi_tupla[1]</code> # imprime 20 <code>print mi_tupla[2]</code> # imprime 1275.48	<code>print mi_lista[0]</code> # imprime texto <code>print mi_lista[1]</code> # imprime 20 <code>print mi_lista[2]</code> # imprime 1275.48	<code>print mi_dict['clave uno']</code> # imprime texto <code>print mi_dict['clave dos']</code> # imprime 20 <code>print mi_dict['clave tres']</code> # imprime 1275.48
Modificar uno de sus valores	NO SE PUEDE	<code>mi_lista[0] = 'cambió'</code>	<code>mi_dict['clave dos'] = 34</code>

Ejercicios



Notebook 3



Estructuras de datos

Arrays, matrices y tensores

Sin embargo, no permiten realizar operaciones. Para ello, hace falta una librería, por ejemplo, la *numpy*.

La librería *numpy* permite definir arrays y matrices de 2 o más dimensiones que permiten realizar operaciones matemáticas.

Estructuras de datos

Arrays, matrices y tensores

1D Array

3	2
---	---

2D Array

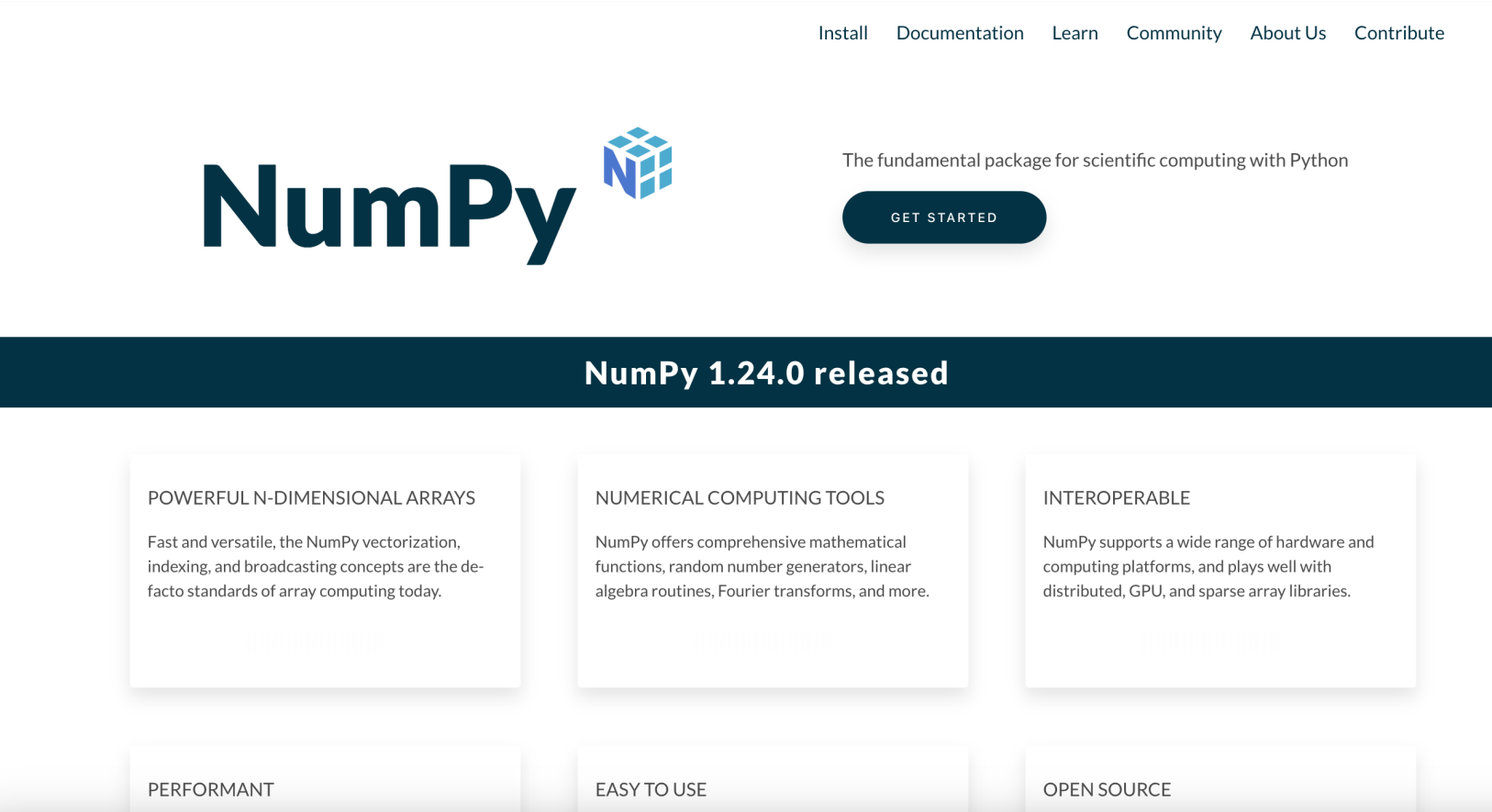
1	0	1
3	4	1

3D Array

1	7	9
5	9	3
7	9	9

Estructuras de datos

NumPy

The image shows the NumPy website homepage. At the top right, there are navigation links: Install, Documentation, Learn, Community, About Us, and Contribute. The main header features the NumPy logo, which consists of the word "NumPy" in a dark blue font and a 3D cube icon with blue and white faces. To the right of the logo, it says "The fundamental package for scientific computing with Python" and a dark blue button with the text "GET STARTED". Below this is a dark blue banner with the text "NumPy 1.24.0 released". Underneath the banner are six white boxes arranged in two rows of three. The first row contains boxes for "POWERFUL N-DIMENSIONAL ARRAYS", "NUMERICAL COMPUTING TOOLS", and "INTEROPERABLE". The second row contains boxes for "PERFORMANT", "EASY TO USE", and "OPEN SOURCE". Each box has a title and a short description of the feature.

Install Documentation Learn Community About Us Contribute

NumPy

The fundamental package for scientific computing with Python

GET STARTED

NumPy 1.24.0 released

POWERFUL N-DIMENSIONAL ARRAYS
Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

NUMERICAL COMPUTING TOOLS
NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

INTEROPERABLE
NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.

PERFORMANT

EASY TO USE

OPEN SOURCE

NumPy.org

Ejercicio: Jugar con el shell interactivo de la Plataforma.

Estructuras de datos

Clases

```
1 class Persona:  
2     def __init__(self, nombre, dni, edad):  
3         self.nombre = nombre  
4         self.dni = dni  
5         self.edad = edad
```

`self.atributo = parámetro`

Estructuras de datos

Clases: Objetos e inicialización

```
toni = Persona('Antonio_Pérez', '98761234Q', 20)
```

```
toni.nombre = 'Antonio_Pérez'  
toni.dni = '98761234Q'  
toni.edad = 20
```

Estructuras de datos

Clases: Recorrer lista de objetos

```
toni = Persona('Antonio_Pérez', '98761234Q', 20)
juan = Persona('Juan_Pérez', '12345678Z', 19)
pedro = Persona('Pedro_López', '23456789D', 18)
alumnos = [toni, juan, pedro]
```

```
1 for alumno in alumnos:
2     print(alumno.dni)
```

```
1 for i in range(len(alumnos)):
2     print(alumnos[i].dni)
```

Estructuras de datos

Clases: Métodos

```
1 class Persona:
2     def __init__(self, nombre, dni, edad):
3         self.nombre = nombre
4         self.dni = dni
5         self.edad = edad
6
7     def iniciales(self):
8         cadena = ''
9         for carácter in self.nombre:
10             if carácter >= 'A' and carácter <= 'Z':
11                 cadena = cadena + carácter + '._'
12         return cadena
```

Estructuras de datos

Clases: Método *string*

```
>>> print(juan)↵  
Nombre: Juan Pérez  
DNI: 12345678Z  
Edad: 19
```

```
1 class Persona:  
2     def __init__(self, nombre, dni, edad):  
3         self.nombre = nombre  
4         self.dni = dni  
5         self.edad = edad  
6  
7     def iniciales(self):  
8         cadena = ''  
9         for carácter in self.nombre:  
10             if carácter >= 'A' and carácter <= 'Z':  
11                 cadena = cadena + carácter + '.  
12         return cadena  
13  
14     def __str__(self):  
15         cadena = 'Nombre: {0}\n'.format(self.nombre)  
16         cadena = cadena + 'DNI: {0}\n'.format(self.dni)  
17         cadena = cadena + 'Edad: {0}\n'.format(self.edad)  
18         return cadena
```

Ejercicios



Notebook 4



Contacto

Correo: a.cobo.aguilera@gmail.com

LinkedIn: [Aurora Cobo Aguilera](#)

GitHub: [AuroraCoboAguilera](#)

Google Scholar: [Aurora Cobo Aguilera](#)





red.es

Centro de
Referencia Nacional
en Comercio Electrónico
y Marketing

CRN
Digital



UNIÓN EUROPEA

"El FSE invierte en tu futuro"

Fondo Social Europeo

