

# SISTEMAS DE GESTIÓN EMPRESARIAL

**Curso:** 2º Desarrollo de Aplicaciones Multiplataforma

**Centro:** Escuela Arte Granada

**Profesor:** Daniel López Lozano





# TEMA 4.

## Elementos avanzados de Python

# INDICE DE CONTENIDOS

1) Programación Funcional

2) Comprensión de listas y diccionarios.

3) Acceso a ficheros

4) Manejo de excepciones

5) Programación Orientada a Objetos

6) Instalación de módulos de terceros

7) Acceso a APIs usando JSON

# PROGRAMACIÓN FUNCIONAL



# INTRODUCCION

- ❑ Los diccionarios sirven para no abusar tanto de los condicionales:

```
DISFRAZ_DEFECTO="PECERA"
adversario=input("Quien tienes enfrente")

if adversario.lower()=="loki":
    misterio="Lady Loki"
elif adversario.lower()=="hulk":
    misterio="Thanos"
elif adversario.lower()=="thor":
    misterio="Odin"
elif adversario.lower()=="superman":
    misterio="Darkseid"
else:
    misterio=DISFRAZ_DEFECTO

print(f"Misterio se disfraza:{misterio}")
```

```
disfraces_misterio={
    "superman":"Darkseid",
    "thor":"Odin",
    "loki":"Lady Loki",
    "hulk":"Thanos"
}

adversario=input("Quien tienes enfrente")
misterio=disfraces_misterio.get(adversario.lower(),DISFRAZ_DEFECTO)
print(f"Misterio se disfraza:{misterio}")
```

# LAS FUNCIONES SON TIPOS DE DATOS

- ❑ Podemos ampliar las posibilidades de los anteriores usando funciones en un diccionario

```
from random import choice
numeros=[1,2,3,4,5]
operacion=input("Que operacion quieres")

if operacion == "sum":
    res=0
    for num in numeros:
        res+=num
elif operacion == "multi":
    res=1
    for num in numeros:
        res*=num
elif operacion == "ale":
    res=choice(numeros)
else:
    res="No existe"
```

# LAS FUNCIONES SON TIPOS DE DATOS

- ❑ Permitiendo hacer acciones todo lo complejas que queramos:

```
def sumador(lista):  
    res=0  
    for num in lista:  
        res+=num  
    return res  
  
def multiplicador(lista):  
    res=0  
    for num in lista:  
        res*=num  
    return res  
  
def aleatorio(lista):  
    return choice(lista)  
  
operaciones={}  
operaciones["sum"]=sumador  
operaciones["multi"]=multiplicador  
operaciones["ale"]=aleatorio
```

# LAS FUNCIONES SON TIPOS DE DATOS

- ❑ Para ejecutar lo anterior si error debemos usar el metodo get para mapas y una funcion lambda para el caso en que no se llame a un valor correcto:

```
nombre=input("Que operacion quieres realizar?")
operacion=operaciones.get(nombre,lambda lista: "No existe dicha operacion")
print(operacion(numeros))
```



# FUNCIONES DE ORDEN SUPERIOR

- ❑ En Python una función puede tener como parámetro otra función.
- ❑ A ese tipo de funciones se les llama de orden superior.

```
def aplica(funcion, argumento):  
    return funcion(argumento)  
  
def cuadrado(n):  
    return n*n  
  
def cubo(n):  
    return n**3  
  
aplica(cuadrado, 5)  
#Resultado 25  
aplica(cubo, 5)  
#Resultado 125
```

# FUNCION MAP

- ❑ Devuelve una objeto iterable con los resultados de aplicar la función f a los elementos de la colección.
- ❑ Si la función requiere n argumentos entonces deben pasarse n colecciones con los argumentos.

```
def cuadrado(n):  
    return n * n  
  
list(map(cuadrado, [1, 2, 3]))  
#Resultado [1, 4, 9]  
  
def rectangulo(a, b):  
    return a * b  
  
list(map(rectangulo, (1, 2, 3), (4, 5, 6)))  
#Resultado(4, 10, 18)
```

# FUNCION FILTER

- ❑ Devuelve una objeto iterable con los elementos de la colección que devuelven True al aplicarles la función

```
def par(n):  
    return n % 2 == 0  
  
list(filter(par, range(19)))  
#Resultado [0, 2, 4, 6, 8]
```

# FUNCION REDUCE

- ❑ Aplicar la función a los primeros elementos de la secuencia.
- ❑ Con el valor obtenido vuelve a aplicar la función a ese valor y el siguiente de la secuencia, y así hasta que no quedan más elementos en la lista.
- ❑ Devuelve el valor resultado de la última aplicación de la función.

# FUNCION REDUCE

```
from functools import reduce

def suma(acumulador=0, elemento=0):
    return acumulador * elemento

reduce(suma, range(1, 5))
#Resultado 10

def producto(acumulador=1, elemento=0):
    return acumulador * elemento

reduce(producto, range(1, 5))
#Resultado 24
```

# COMPRENSIÓN DE LISTAS Y DICIONARIOS



# INTRODUCCIÓN

- ❑ En muchas aplicaciones es habitual aplicar una función o una operación con los elementos de una colección y obtener una nueva colección de elementos transformados.
- ❑ Aunque esto se puede hacer recorriendo la secuencia con un bucle iterativo y con otras opciones en Python
- ❑ Python incorpora un mecanismo muy potente que permite esto mismo de manera más simple.

# INTRODUCCIÓN

**[EXPRESION for VARIABLE in  
lista if CONDICIÓN]**

- ❑ Esta instrucción genera la lista cuyos elementos son el resultado de evaluar la expresión para cada valor que toma la variable , donde variable toma todos los valores de la lista que cumplen la condición.



```
[x ** 2 for x in range(10)]  
#Resultado  
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]  
  
[x for x in range(10) if x % 2 == 0]  
#Resultado  
[0, 2, 4, 6, 8]  
  
[x ** 2 for x in range(10) if x % 2 == 0]  
#Resultado  
[0, 4, 16, 36, 64]
```

## EJEMPLOS DE USO

- ❑ Como dice la definición compacta líneas de código de bucle for

# PARA QUÉ SIRVE

- ❑ Esto sería un ejemplo normal para generar listas en comparación que usando compresión de listas

```
#SIN COMPRENSION
list_a = []
for i in range(1, 20):
    list_a.append(i**2)
#CON COMPRENSION
list_b = [i**2 for i in range(1, 20)]
```



```
notas = {'Carmen':5,  
        'Antonio':4,  
        'Juan':8,  
        'Mónica':9,  
        'María': 6,  
        'Pablo':3}  
  
#SIN COMPRENSION  
aprobados=[]  
for (nombre,nota) in notas.items():  
    if nota>=5:  
        aprobados.append(nombre)  
  
#CON COMPRENSION  
[nombre for (nombre, nota) in notas.items() if nota >= 5]
```

## PARA QUÉ SIRVE

- ❑ Podemos usarlo con datos que provengan de otras fuentes siempre y cuando procedan de una colección



# MÁS EJEMPLOS

- ❑ Aplicados a string
- ❑ Solo mostrar ficheros de códigos Python (acaba en .py)

```
palabras = ['casa', 'perro', 'puerta', 'pizza']  
cap = [palabra.title() for palabra in palabras]  
#Resultado  
['Casa', 'Perro', 'Puerta', 'Pizza']  
  
import os  
python=[fichero for fichero in os.listdir('.')  
        |         if fichero.endswith('.py')]
```

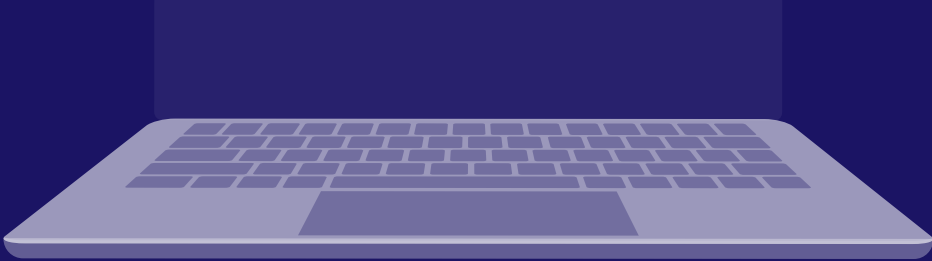


# COMPRESION DE DICCIONARIOS

```
{EXPRESSION-CLAVE:EXPRESSION-VALOR  
  for VARIABLES  
  in lista  
  if CONDICION}
```

# COMPRESION DE DICCIONARIOS

- ❑ Esta instrucción genera el diccionario formado por los pares cuyas claves son el resultado de evaluar la expresion-clave y cuyos valores son el resultado de evaluar la expresión expresion-valor, para cada valor que toma la variable, donde variable toma todos los valores de la lista que cumplen la condición.



```
{palabra:len(palabra)
for palabra in ['I', 'love', 'Python']}
#Resultado
{'I': 1, 'love': 4, 'Python': 6}
```

## EJEMPLOS DE USO

- ❑ Partiendo siempre de una lista



# MÁS EJEMPLOS

- ❑ Si convertimos un diccionario en lista

```
notas = {'Carmen':5,  
        'Antonio':4,  
        'Juan':8,  
        'Mónica':9,  
        'María': 6,  
        'Pablo':3}  
  
{nombre: nota +1  
for (nombre, nota) in notas.items()  
if nota >= 5}  
#Resultado  
{'Carmen': 6, 'Juan': 9, 'Mónica': 10, 'María': 7}
```





```
import random
clientes = ["Alex", "Bob", "Carol", "Dave", "Flow", "Katie", "Nate"]
descuentos = {cliente: random.randint(1, 100)
               for cliente in clientes}
#Resultado
{'Alex': 16, 'Bob': 26, 'Carol': 83, 'Dave': 21,
 'Flow': 38, 'Katie': 47, 'Nate': 89}
```

## MÁS EJEMPLOS

- ❑ Crear un diccionario para añadir datos a una lista

# CUANDO USAR COMPRENSION

- ❑ Puede utilizar la comprensión de listas en muchos casos en los que necesite generar una lista a partir de un iterable.
- ❑ Sin embargo, el mejor momento para usar este método es cuando necesita agregar o extraer elementos a una lista de manera consistente de acuerdo con un patrón establecido.
- ❑ Los desarrolladores de Python los utilizan principalmente para extraer datos de una gran colección de elementos.

# LISTA DE DICCIONARIOS

- ❑ Suponemos que tenemos declarada la siguiente estructura

```
#El objetivo es filtrar tanto campos
#como filtrar datos
students = [
    {
        "name" : "Jacob Martin",
        "father name" : "Ros Martin",
        "Address" : "123 Hill Street",
    }, {
        "name" : "Angela Stevens",
        "father name" : "Robert Stevens",
        "Address" : "3 Upper Street London",
    }, {
        "name" : "Ricky Smart",
        "father name" : "William Smart",
        "Address" : "Unknown",
    }
]
```

# EJEMPLOS LISTAS DE DICCIONARIOS

```
#SOLO LOS NOMBRES
```

```
nombre = [student['name'] for student in students]
```

```
#DICCIONARIO NOMBRE-DIRECCION
```

```
nombre_dir={student['name']:student["Address"]  
            for student in students}
```

```
#LISTA SOLO NOMBRE Y DIRECCION
```

```
agenda= [{"nombre":student['name'],  
          "direccion":student['Address']}  
         for student in students]
```



# ACCESO A FICHEROS



# INTRODUCCION

- ❑ Hasta ahora hemos visto como interactuar con un programa a través del teclado (entrada de datos) y la terminal (salida),pero en la mayor parte de las aplicaciones reales tendremos que leer y escribir datos en ficheros.
- ❑ Al utilizar ficheros para guardar los datos estos perdurarán tras la ejecución del programa, pudiendo ser consultados o utilizados más tarde.

# INTRODUCCION

Las operaciones más habituales con ficheros son:

- ❑ Crear un fichero.
- ❑ Escribir datos en un fichero.
- ❑ Leer datos de un fichero.
- ❑ Borrar un fichero.



# CREAR UN FICHERO

- ❑ Para crear un fichero nuevo se utiliza la instrucción

**`open(ruta, 'w')`** : Crea el fichero con la ruta `ruta`, lo abre en modo escritura (el argumento `'w'` significa write) y devuelve un objeto que lo referencia.

- ❑ Si el fichero indicado por la ruta ya existe en el sistema, se reemplazará por el nuevo.
- ❑ Una vez creado el fichero, para escribir datos en él se utiliza el método

**`fichero.write(c)`**: Escribe la cadena `"c"` en el fichero referenciado por `fichero`.

# CERRAR UN FICHERO

- ❑ Para cerrar un fichero se utiliza el método:

**`fichero.close()`**: Cierra el fichero referenciado por el objeto fichero.

- ❑ Cuando se termina de trabajar con un fichero conviene cerrarlo, sobre todo si se abre en modo escritura, ya que mientras está abierto en este modo no se puede abrir por otra aplicación.
- ❑ Si no se cierra explícitamente un fichero, Python intentará cerrarlo cuando estime que ya no se va a usar más.

# AÑADIR DATOS A UN FICHERO

- ❑ Si en lugar de crear un fichero nuevo queremos añadir datos a un fichero existente se debe utilizar la instrucción `open(ruta, 'a')` : Abre el fichero con la ruta ruta en modo añadir (el argumento 'a' significa append) y devuelve un objeto que lo referencia.
- ❑ Una vez abierto el fichero, se utiliza el método de escritura anterior y los datos se añaden al final del fichero.

# EJEMPLOS

```
f = open('bienvenida.txt', 'w')  
f.write('¡Bienvenido a Python!')  
f.close()
```

```
f = open('bienvenida.txt', 'a')  
f.write('\n¡Hasta pronto!')  
f.close()
```

# LECTURA DE DATOS

- ❑ Para abrir un fichero en modo lectura se utiliza la instrucción:

**`open(ruta, 'r')`** : Abre el fichero con la ruta ruta en modo lectura (el argumento 'r' significa read) y devuelve un objeto que lo referencia.

- ❑ Una vez abierto el fichero, se puede leer todo el contenido del fichero o se puede leer línea a línea.

# LECTURA DE DATOS

- ❑ **`fichero.read()`** :Devuelve todos los datos contenidos en fichero como una cadena de caracteres.
- ❑ **`fichero.readlines()`** : Devuelve una lista de cadenas de caracteres donde cada cadena es una línea del fichero referenciado.

Si se necesita leer contenido con acentos y caracteres especiales

- ❑ **`open(fichero, 'r', encoding='utf-8')`** :  
Devuelve una lista de cadenas de caracteres donde cada cadena es una línea del fichero referenciado.

# LECTURA DE DATOS

```
f = open('bienvenida.txt', 'r')
print(f.read())
f.close()
#¡Bienvenido a Python!
#¡Hasta pronto!
```

```
f = open('bienvenida.txt', 'r')
lineas = print(f.readlines())
print(lineas)
f.close()
['¡Bienvenido a Python!\n', '¡Hasta pronto!']
```

# OPERACIONES CON FICHEROS

- ❑ Para renombrar o borrar un fichero se utilizan funciones del módulo `os`:

`os.rename(ruta1, ruta2)` : Renombra y mueve el fichero de la ruta1 a la ruta2.

`os.remove(ruta)` : Borra el fichero de la ruta.

```
import os
f = 'bienvenida.txt'
if os.path.isfile(f):
    os.rename(f, 'saludo.txt') # renombrado
else:
    print('¡El fichero', f, 'no existe!')
```



# OPERACIONES CON FICHEROS

- ❑ Antes de borrar o renombra un directorio conviene comprobar que existe para que no se produzca un error. Para ello se utiliza la función.
- ❑ **`os.path.isfile(ruta)`**: Devuelve True si existe un fichero en la ruta y False en caso contrario.

```
f = 'saludo.txt'
if os.path.isfile(f):
    os.remove(f) # borrado
else:
    print('¡El fichero', f, 'no existe!')
```

# MÁS OPERACIONES

- ❑ `os.listdir(ruta)` : Devuelve una lista con los ficheros y directorios contenidos en la ruta.
- ❑ `os.mkdir(ruta)` : Crea un nuevo directorio en la ruta.
- ❑ `os.chdir(ruta)` : Cambia el directorio actual al indicado por la ruta.
- ❑ `os.getcwd()` : Devuelve una cadena con la ruta del directorio actual.
- ❑ `os.rmdir(ruta)` : Borra el directorio de la ruta, siempre y cuando esté vacío.

# LEER FICHERO DE INTERNET

- ❑ Para leer un fichero de internet hay que utilizar la función `urlopen` del módulo `urllib.request`.

**`urlopen(url)`:** Abre el fichero con la url especificada y devuelve un objeto del tipo fichero al que se puede acceder con los métodos de lectura de ficheros anteriores.

```
from urllib import request
f=request.urlopen("https://firebasestorage.googleapis.com/
print(f.read())
```

# LEER FICHERO DE INTERNET

- ❑ Al obtener un fichero que ha pasado a través de distintas redes con distintas codificaciones puedes obtener datos en formato con caracteres que producen errores.

- ❑ Original

Lorem Ipsum es simplemente el texto de relleno de las imprentas y  
 Lorem Ipsum ha sido el texto de relleno estándar de las industrias  
 cuando un impresor (N. del T. persona que se dedica a la imprenta)  
 galería de textos y los mezcló de tal manera que logró hacer un li  
 No sólo sobrevivió 500 años, sino que tambien ingresó como texto d  
 quedando esencialmente igual al original. Fue popularizado en los  
 las cuales contenian pasajes de Lorem Ipsum, y más recientemente d  
 el cual incluye versiones de Lorem Ipsum.

- ❑ Desde URL de internet

```
b'Lorem Ipsum es simplemente el texto de relleno de las imprentas y archivos de texto. \r\nLorem Ipsum ha sido el texto de relleno est\xc3\xa1ndar de las industrias desde
```

# LEER FICHERO DE INTERNET

- ❑ El propio manejador de ficheros incluye una opcion para obtener un fichero en la codificacion deseada al leer datos

```
f=request.urlopen("https://fire  
print(f.read().decode('utf-8'))
```

- ❑ Sin decode

```
b'MATRICULA;MODELO;PRECIO\r\n1234GPY;GT400;39500.65\r\n9845FEZ;IBIZA;10000.0\r\n4367HAB;MODEL S;105970.0\r\n4891UNO;F40;251000.0'
```

- ❑ Con decode

```
MATRICULA;MODELO;PRECIO  
1234GPY;GT400;39500.65  
9845FEZ;IBIZA;10000.0  
4367HAB;MODEL S;105970.0  
4891UNO;F40;251000.0
```

# MANIPULAR DATOS CSV

- ❑ Python dispone de una librería estandar para el manejo de datos en formato CSV de manera rápida. Si estáis interesados (csv es un formato de intercambio bastante aceptado para hojas de cálculo de excel o similares) aquí teneis el enlace de documentación:

<https://docs.python.org/es/3/library/csv.html>

# MANEJO DE EXCEPCIONES



# INTRODUCCION

- ❑ Python utiliza un objeto especial llamado excepción para controlar cualquier error que pueda ocurrir durante la ejecución de un programa.
- ❑ Cuando ocurre un error durante la ejecución de un programa, Python crea una excepción.
- ❑ Si no se controla esta excepción la ejecución del programa se detiene y se muestra el error (traceback).

```
1 >>> print(1 / 0) # Error al intentar dividir por 0.  
2 Traceback (most recent call last):  
3   File "<stdin>", line 1, in <module>  
4 ZeroDivisionError: division by zero
```



# TIPOS DE EXCEPCIONES

Los principales excepciones definidas en Python son:

- ❑ `TypeError` :Ocurre cuando se aplica una operación o función a un dato del tipo inapropiado.
- ❑ `ZeroDivisionError` : Ocurre cuando se intenta dividir por cero.
- ❑ `OverflowError` :Ocurre cuando un cálculo excede el límite para un tipo de dato numérico.
- ❑ `IndexError` : Ocurre cuando se intenta acceder a una secuencia con un índice que no existe.

# TIPOS DE EXCEPCIONES

- ❑ `KeyError` : Ocorre cuando se intenta acceder a un diccionario con una clave que no existe.
- ❑ `FileNotFoundError`: Ocorre cuando se intenta acceder a un fichero que no existe en la ruta indicada.
- ❑ `ImportError` : Ocorre cuando falla la importación de un módulo.

# CONTROL DE EXCEPCIONES

```
try:
```

```
    bloque código1
```

```
except excepción:
```

```
    bloque código2
```

```
else:
```

```
    bloque código3
```

# CONTROL DE EXCEPCIONES

- ❑ Esta instrucción ejecuta el primer bloque de código y si se produce un error que genera una excepción del tipo especificado entonces ejecuta el segundo bloque de código.
- ❑ Mientras que si no se produce ningún error, se ejecuta el tercer bloque de código.

# EJEMPLOS

```
def division(a, b):  
    try:  
        result = a / b  
    except ZeroDivisionError:  
        print('¡No se puede dividir por cero!')  
    else:  
        print(result)
```

# EJEMPLOS

```
try:
    edad=int(input("Dime tu edad con números"))
except TypeError:
    print("No has escrito un número")
else:
    print("Gracias")
```

# EJEMPLOS

```
try:
    f = open('fichero.txt') # El fichero no existe
except FileNotFoundError:
    print('¡El fichero no existe!')
else:
    print(f.read())
# ¡El fichero no existe
```

# PROGRAMACIÓN ORIENTADA A OBJETOS





# INTRODUCCIÓN

- ❑ Los objetos suelen representar conceptos del dominio del programa, como un estudiante, un coche, un teléfono, etc.
- ❑ Los datos que describen las características del objeto se llaman atributos y son la parte estática del objeto, mientras que las operaciones que puede realizar el objeto se llaman métodos y son la parte dinámica del objeto.

# INTRODUCCIÓN

- ❑ Ejemplo. Una tarjeta de crédito puede representarse como un objeto:
  - ✓ Atributos: Número de la tarjeta, titular, balance, fecha de caducidad, pin, entidad emisora, estado (activa o no), etc.
  - ✓ Métodos: Activar, pagar, renovar, anular.

# INTRODUCCIÓN

- ❑ Los objetos con los mismos atributos y métodos se agrupan clases. Las clases definen los atributos y los métodos, y por tanto, la semántica o comportamiento que tienen los objetos que pertenecen a esa clase.
- ❑ Se puede pensar en una clase como en un molde a partir del cuál se pueden crear objetos.
- ❑ Para declarar una clase se utiliza la palabra clave `class` seguida del nombre de la clase y dos puntos, de acuerdo a la siguiente sintaxis:

# INTRODUCCIÓN

```
class <nombre-clase>:  
    <atributos>  
    <métodos>
```

- ❑ Los atributos se definen igual que las variables mientras que los métodos se definen igual que las funciones. Tanto unos como otros tienen que estar indentados por 4 espacios en el cuerpo de la clase.

# INTRODUCCIÓN

- ❑ Para crear un objeto de una determinada clase se utiliza el nombre de la clase seguida de los parámetros necesarios.

## `clase(parámetros) :`

- ❑ Crea un objeto de la clase `clase` inicializado con los parámetros dados.
- ❑ Cuando se crea un objeto de una clase se dice que el objeto es una instancia de la clase.

# INTRODUCCIÓN

- ❑ Los métodos de una clase son las funciones que definen el comportamiento de los objetos de esa clase.
- ❑ Se definen como las funciones con la palabra reservada `def`.
- ❑ La única diferencia es que su primer parámetro es especial y se denomina `self`.

# INTRODUCCIÓN

- ❑ Este parámetro hace siempre referencia al objeto desde donde se llama el método, de manera que para acceder a los atributos o métodos de una clase en su propia definición se puede utilizar la sintaxis `self.atributo` o `self.método`.

# INTRODUCCIÓN

- ❑ La razón por la que existe el parámetro `self` es porque Python traduce la llamada a un método de un objeto `objeto.método(parámetros)` en la llamada `clase.método(objeto, parámetros)`, es decir, se llama al método definido en la clase del objeto, pasando como primer argumento el propio objeto, que se asocia al parámetro `self`.



# MÉTODO `__init__`

- ❑ En la definición de una clase suele haber un método llamado `__init__` que se conoce como inicializador.
- ❑ Este método es un método especial que se llama cada vez que se instancia una clase y sirve para inicializar el objeto que se crea.
- ❑ Este método crea los atributos que deben tener todos los objetos de la clase y por tanto contiene los parámetros necesarios para su creación, pero no devuelve nada.

# MÉTODO `__init__`

- ❑ Se invoca cada vez que se instancia un objeto de esa clase.

```
class Tarjeta:
    def __init__(self, id, cantidad = 0):
        self._id = id
        self._saldo = cantidad
    def mostrar_saldo(self):
        print('El saldo es', self._saldo, '€')

# Creación de un objeto con argumentos
t = Tarjeta('1111111111', 1000)
t.muestra_saldo()
#El saldo es 1000 €
```

# ATRIBUTOS DE INSTANCIA vs ATRIBUTOS DE CLASE

- ❑ Los atributos que se crean dentro del método `__init__` se conocen como atributos del objeto, mientras que los que se crean fuera de él se conocen como atributos de la clase.
- ❑ Mientras que los primeros son propios de cada objeto y por tanto pueden tomar valores distintos, los valores de los atributos de la clase son los mismos para cualquier objeto de la clase.
- ❑ En general, no deben usarse atributos de clase, excepto para almacenar valores constantes o para tener valores que compartan todas las instancias de una misma clase.

# ATRIBUTOS DE INSTANCIA vs ATRIBUTOS DE CLASE

- ❑ No existen modificadores de visibilidad (private o public), por lo que existen un acuerdo no escrito que si una variable empieza por `_` significa que es una variable que no se pueden modificar desde fuera de la clase

```
class Circulo:
    # Atributo de clase
    pi = 3.14159
    def __init__(self, radio):
        # Atributo de instancia
        self._radio = radio
    def area(self):
        return Circulo.pi * self._radio ** 2

c1 = Circulo(2)
c2 = Circulo(3)
print(c1.area())
12.56636
print(c2.area())
28.27431
print(c1._pi)
3.14159
print(c2._pi)
3.14159
```

# MÉTODO `__str__`

- ❑ Otro método especial es el método llamado `__str__` que se invoca cada vez que se llama a las funciones `print` o `str`.
- ❑ Devuelve siempre una cadena que se suele utilizar para dar una descripción informal del objeto.
- ❑ Si no se define en la clase, cada vez que se llama a estas funciones con un objeto de la clase, se muestra por defecto la posición de memoria del objeto.

# MÉTODO `__str__`

```
class Tarjeta:
    def __init__(self, numero, cantidad = 0):
        self._numero = numero
        self._saldo = cantidad
    def __str__(self):
        return f"Tarjeta número{self._numero} con saldo {self._saldo}€"

t = Tarjeta('0123456789', 1000)
print(t)
#Tarjeta número 0123456789 con saldo 1000.00€
```

# MÉTODO `__eq__`

- ❑ Otro método especial es el método llamado `__eq__` que es el equivalente al método `equals` de Java para poder comparar si dos objetos de la misma clase son iguales. Por ejemplo con la clase `Tarjeta`:

```
def __eq__(self, otra_tarjeta):  
    return self.numero==otra_tarjeta.numero
```

# HERENCIA

- ❑ Una de las características más potentes de la programación orientada a objetos es la herencia, que permite definir una especialización de una clase añadiendo nuevos atributos o métodos.
- ❑ La nueva clase se conoce como clase hija y hereda los atributos y métodos de la clase original que se conoce como clase madre.
- ❑ Para crear una clase a partir de otra existente se utiliza la misma sintaxis que para definir una clase, pero poniendo detrás del nombre de la clase entre paréntesis los nombres de las clases madre de las que hereda.



# HERENCIA

- ❑ Ejemplo. A partir de la clase Tarjeta definida antes podemos crear mediante herencia otra clase Tarjeta\_Descuento para representar las tarjetas de crédito que aplican un descuento sobre las compras.

```
class Tarjeta:
    def __init__(self, id, cantidad = 0):
        self.id = id
        self.saldo = cantidad
    # Método de la clase Tarjeta que hereda la clase Tarjeta_descuento
    def mostrar_saldo(self):
        print('El saldo es', self.saldo, '€.')
```

# HERENCIA

```
class Tarjeta_descuento(Tarjeta):  
    def __init__(self, id, descuento, cantidad = 0):  
        self.id = id  
        self.descuento = descuento  
        self.saldo = cantidad  
    # Método exclusivo de la clase Tarjeta_descuento  
    def mostrar_descuento(self):  
        print('Descuento de', self.descuento, '% en los pagos.')
```

```
t = Tarjeta_descuento('0123456789', 2, 1000)  
t.mostrar_saldo()  
#El saldo es 1000 €.  
t.mostrar_descuento()  
#Descuento de 2 % en los pagos.
```

# HERENCIA

- ❑ La principal ventaja de la herencia es que evita la repetición de código y por tanto los programas son más fáciles de mantener.
- ❑ En el ejemplo de la tarjeta de crédito, el método `mostrar_saldo` solo se define en la clase madre.
- ❑ De esta manera, cualquier cambio que se haga en el cuerpo del método en la clase madre, automáticamente se propaga a las clases hijas.

# HERENCIA

- ❑ Sin la herencia, este método tendría que replicarse en cada una de las clases hijas y cada vez que se hiciese un cambio en él, habría que replicarlo también en las clases hijas.
- ❑ A partir de una clase derivada mediante herencia se pueden crear nuevas clases hijas aplicando de nuevo la herencia.
- ❑ Ello da lugar a una jerarquía de clases que puede representarse como un árbol donde cada clase hija se representa como una rama que sale de la clase madre.

# HERENCIA

- ❑ Debido a la herencia, cualquier objeto creado a partir de una clase es una instancia de la clase, pero también lo es de las clases que son ancestros de esa clase en la jerarquía de clases.
- ❑ El siguiente comando permite averiguar si un objeto es instancia de una clase:

**`isinstance(objeto, clase):`**

- ❑ Devuelve True si el objeto objeto es una instancia de la clase clase y False en caso contrario.

# HERENCIA

```
t1 = Tarjeta('1111111111', 0)
t2 = t = Tarjeta_descuento('2222222222', 2, 1000)
instance(t1, Tarjeta)
True
instance(t1, Tarjeta_descuento)
False
instance(t2, Tarjeta_descuento)
True
instance(t2, Tarjeta)
True
```

# SOBRECARGA Y POLIMORFISMO

- ❑ Los objetos de una clase hija heredan los atributos y métodos de la clase madre y, por tanto, a priori tienen el mismo comportamiento que los objetos de la clase madre.
- ❑ Pero la clase hija puede definir nuevos atributos o métodos o reescribir los métodos de la clase madre de manera que sus objetos presenten un comportamiento distinto. Esto último se conoce como sobrecarga.

# SOBRECARGA Y POLIMORFISMO

- ❑ De este modo, aunque un objeto de la clase hija y otro de la clase madre pueden tener un mismo método, al invocar ese método sobre el objeto de la clase hija, el comportamiento puede ser distinto a cuando se invoca ese mismo método sobre el objeto de la clase madre.
- ❑ Esto se conoce como polimorfismo y es otra de las características de la programación orientada a objetos.



# SOBRECARGA Y POLIMORFISMO

```
class Tarjeta:
    def __init__(self, id, cantidad = 0):
        self.id = id
        self.saldo = cantidad
    def mostrar_saldo(self):
        print('El saldo es {:.2f}€.'.format(self.saldo))
    def pagar(self, cantidad):
        self.saldo -= cantidad

class Tarjeta_Oro(Tarjeta):
    def __init__(self, id, descuento, cantidad = 0):
        self.id = id
        self.descuento = descuento
        self.saldo = cantidad
    def pagar(self, cantidad):
        self.saldo -= cantidad * (1 - self.descuento / 100)
```

# SOBRECARGA Y POLIMORFISMO

```
t1 = Tarjeta('1111111111', 1000)
t2 = Tarjeta_Oro('2222222222', 1, 1000)
t1.pagar(100)
t1.mostrar_saldo()
#El saldo es 900.00€.
t2.pagar(100)
t2.mostrar_saldo()
#El saldo es 901.00€.
```

# SUPER

- ❑ Si queremos usar en una clase Hija cualquier método de una clase tenemos que usar la llamada `super()` como Java:

```
class Tarjeta_Oro(Tarjeta):  
    def __init__(self, id, descuento, cantidad = 0):  
        super().__init__(id,cantidad)  
        self.descuento = descuento
```

# INSTALACIÓN DE MÓDULOS DE TERCEROS



# INSTALAR MODULOS DE TERCEROS DE PYTHON OFICIALES

- ❑ Para instalar un módulo de terceros que se encuentra en el repositorio base utiliza el comando pip o pip3
- ❑ En este caso necesitamos qrcode y Pillow que es una dependencia del anterior

```
Terminal: Local x + v
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

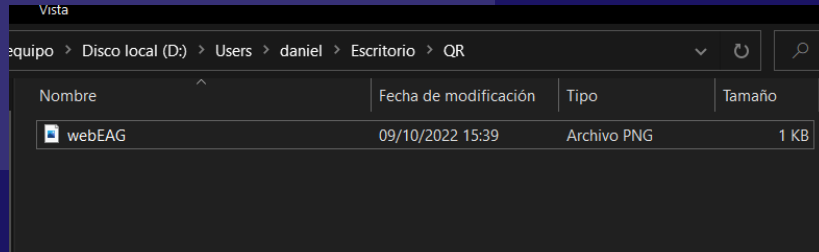
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\danie\PycharmProjects\pythonProject1> pip install qrcode
```

# CODIGO PARA GENERAR CODIGO QR

```
from qrcode import make
informacion="https://www.escuelaartegranada.com"

codigo_qr=make(informacion)
codigo_qr.save("D:/Users/daniel/Desktop/QR/webEAG.png")
```



# ACCESO A APIS USANDO JSON



# INTRODUCCIÓN

- ❑ JSON (JavaScript Object Notation) es un formato usado para representar y almacenar información.
- ❑ Se usa comúnmente para transferir información a través de la web y para almacenar ajustes y configuración.
- ❑ Los archivos JSON tienen una extensión .json.
- ❑ Puedes leer archivos JSON y crear diccionarios de Python a partir de sus pares clave-valor.



# INTRODUCCIÓN

- ❑ Para obtener información de una API JSON que es básicamente un servidor de datos que permite el intercambio de información en formato JSON para dar un servicio consumido por un programa que se apoya en él.
- ❑ En Python tenemos la librería **json** que permite transformar un string en formato json a un diccionario Python y viceversa.

# EJEMPLO API RTVE

- Hay API que son de dominio publico como por ejemplo: <https://www.rtve.es/api/noticias.json>

```
{
  "page": {
    "items": [
      {
        "uri": "http://www.rtve.es/api/noticias/2405510",
        "htmlUrl": "https://www.rtve.es/deportes/20221008/futbol-jornada-8-liga-santander-sevilla-athletic-club-cronica/2405510.shtml",
        "htmlShortUrl": "https://www.rtve.es/n/2405510/",
        "id": "2405510",
        "language": "es",
        "shortTitle": null,
        "mainCategoryRef": "http://www.rtve.es/api/tematicas/124631",
        "popularity": "0.0",
        "popHistoric": "8.0",
        "numVisits": "0",
        "publicationDate": "08-10-2022 20:27:00",
        "expirationDate": null,
        "modificationDate": "08-10-2022 20:27:32",
        "publicationDateTimestamp": 1665253620000,
        "breadCrumbRef": "http://www.rtve.es/api/noticias/2405510/breadCrumb",
        "image": "https://img2.rtve.es/imagenes/sevilla-athletic-club-jornada-8-laliga/1665253045584.jpg",
        "imageSEO": "https://img2.rtve.es/n/futbol-jornada-8-liga-santander-sevilla-athletic-club-cronica_2405510.png",
        "contentType": "noticia",
        "statistics": {
          "numComentarios": 0,
          "numCompartidas": 0
        },
        "contentInitDate": null,
        "contentEndDate": null,
        "anteTitle": "LaLiga | Sevilla - Athletic Club",
        "anteTitleUrl": null,
        "longTitle": "El Athletic amarga el estreno de Sampaoli en Sevilla",
        "frontTitle": null,
        "tabTitle": null,
        "ticker": {
          "tickerSports": false,
          "tickerNews": false
        },
        "essentialInfo": {
          "info": null,
          "photo": null
        },
        "summary": "
        <ul>
        <li>Un gol de Vesga da a los rojiblancos un punto ante los hispalenses, que se adelantaron por Oliver Torres
        </li>
        <li><a href='\"https://www.rtve.es/play/videos/resumenes-de-laliga/\">Todos los goles de la jornada y los res\u00fames de los partidos de Primera</a>, en RTVE Play
        </li>
        </ul>
        ",
        "frontSummary": "
        <ul>
        <li><a href='\"https://www.rtve.es/play/videos/resumenes-de-laliga/\">Todos los goles de la jornada y los res\u00fames de los partidos de Primera</a>, en RTVE Play
        </li>
        </ul>
        ",
        "category": "El Sevilla enfrenta al Athletic Club en el estadio de San Sebasti\u00e1n"
      }
    ]
  }
}
```

```
{
  "page": {
    "items": [
      {
        "uri": "http://www.rtve.es/api/noticias/2405510",
        "htmlUrl": "https://www.rtve.es/deportes/20221008/futbol-jornada-8-liga-santander-sevilla-athletic-club-cronica/2405510.shtml",
        "htmlShortUrl": "https://www.rtve.es/n/2405510/",
        "id": "2405510",
        "language": "es",
        "shortTitle": null,
        "mainCategoryRef": "http://www.rtve.es/api/tematicas/124631",
        "popularity": "0.0",
        "popHistoric": "8.0",
        "numVisits": "0",
        "publicationDate": "08-10-2022 20:27:00",
        "expirationDate": null,
        "modificationDate": "08-10-2022 20:27:32",
        "publicationDateTimestamp": 1665253620000,
        "breadCrumbRef": "http://www.rtve.es/api/noticias/2405510/breadCrumb",
        "image": "https://img2.rtve.es/imagenes/sevilla-athletic-club-jornada-8-laliga/1665253045584.jpg",
        "imageSEO": "https://img2.rtve.es/n/futbol-jornada-8-liga-santander-sevilla-athletic-club-cronica_2405510.png",
        "contentType": "noticia",
        "statistics": {
          "numComentarios": 0,
          "numCompartidas": 0
        },
        "contentInitDate": null,
        "contentEndDate": null,
        "anteTitle": "LaLiga | Sevilla - Athletic Club",
        "anteTitleUrl": null,
        "longTitle": "El Athletic amarga el estreno de Sampaoli en Sevilla",
        "frontTitle": null,
        "tabTitle": null,
        "ticker": {
          "tickerSports": false,
          "tickerNews": false
        },
        "essentialInfo": {
          "info": null,
          "photo": null
        },
        "summary": "
        <ul>
        <li>Un gol de Vesga da a los rojiblancos un punto ante los hispalenses, que se adelantaron por Oliver Torres
        </li>
        <li><a href='\"https://www.rtve.es/play/videos/resumenes-de-laliga/\">Todos los goles de la jornada y los res\u00fames de los partidos de Primera</a>, en RTVE Play
        </li>
        </ul>
        ",
        "frontSummary": "
        <ul>
        <li><a href='\"https://www.rtve.es/play/videos/resumenes-de-laliga/\">Todos los goles de la jornada y los res\u00fames de los partidos de Primera</a>, en RTVE Play
        </li>
        </ul>
        ",
        "category": "El Sevilla enfrenta al Athletic Club en el estadio de San Sebasti\u00e1n"
      }
    ]
  }
}
```

# EJEMPLO API RTVE

```
from urllib import request
from json import loads

f=request.urlopen("https://www.rtve.es/api/noticias.json")
datos_json=f.read().decode("utf-8")
datos_diccionario=loads(datos_json)
for noticia in datos_diccionario['page']['items']:
    print(f"Titulo:{noticia['longTitle']}")
    print(f"Enlace:{noticia['htmlUrl']}")
```

Titulo:El Athletic amarga el estreno de Sampaoli en Sevilla

Enlace:<https://www.rtve.es/deportes/20221008/futbol-jornada-8-liga-santander-sevilla-athletic-club-cronica/2405>

Titulo:Guerra Ucrania - Rusia, en directo | Putin aumenta la seguridad en el puente de Crimea

Enlace:<https://www.rtve.es/noticias/20221008/guerra-ucrania-directo-ultima-hora-noticias/2405492.shtml>

Titulo:¿Cuánto gana un 'influencer'? Las cifras de Laura Escanes, Paula Echevarría, María Pombo y Dulceida

## EJEMPLO API WEATHER MAP

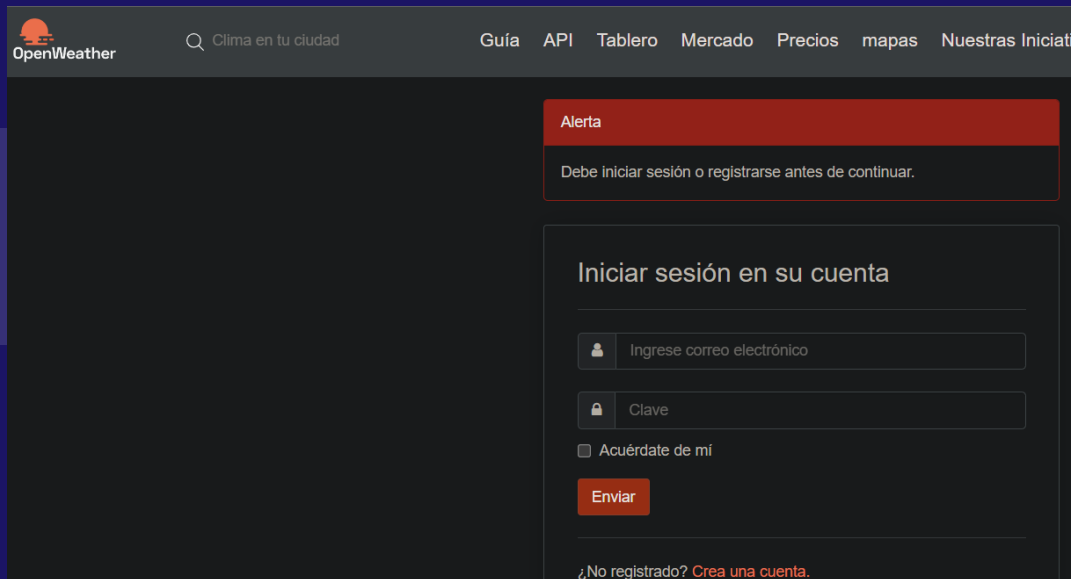
- ❑ Existen otras API que tienen restringido su contenido y hay que registrarse para obtener acceso mediante una API key
- ❑ Para realizar una petición hay que incorporar parametros a la URL
- ❑ En este caso tambien tenemos que incorporar la ciudad de la que queremos saber su meteorologia.

<http://api.openweathermap.org/data/2.5/weather?appid=1eb90fe7a37a9e683d9f8acff01f6871&q=Motril>

# EJEMPLO API WEATHER MAP

❑ Para obtener la API key registrarse en:

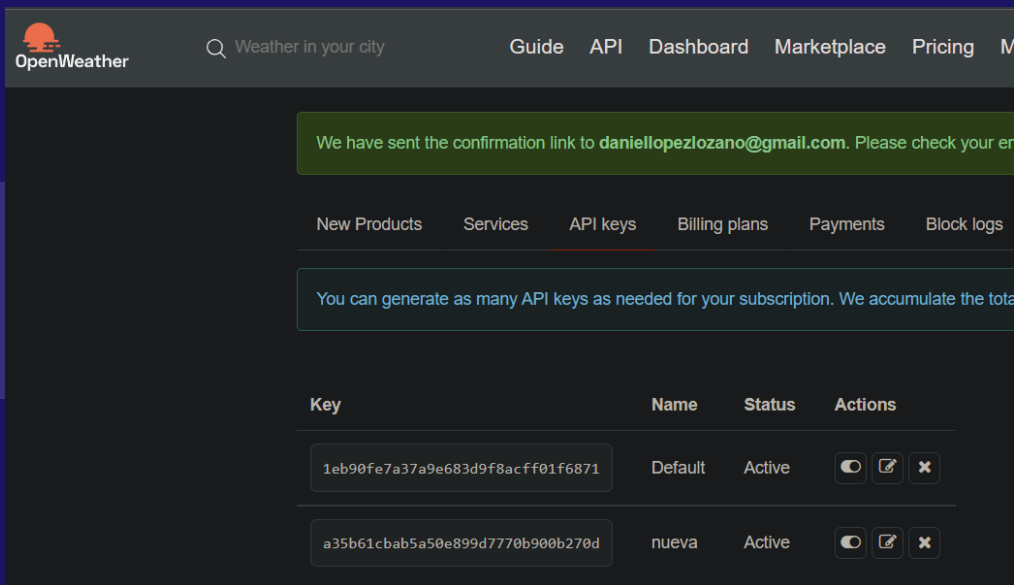
[https://home.openweathermap.org/users/sign\\_in](https://home.openweathermap.org/users/sign_in)







The screenshot shows the OpenWeatherMap login page. At the top, there is a navigation bar with the OpenWeather logo, a search bar with the text "Clima en tu ciudad", and links for "Guía", "API", "Tablero", "Mercado", "Precios", "mapas", and "Nuestras Iniciati". Below the navigation bar, there is a red alert box with the text "Alerta" and "Debe iniciar sesión o registrarse antes de continuar." Below the alert box, there is a login form titled "Iniciar sesión en su cuenta". The form contains two input fields: "Ingrese correo electrónico" and "Clave". Below the input fields, there is a checkbox labeled "Acuérdame de mí" and a red "Enviar" button. At the bottom of the form, there is a link that says "¿No registrado? Crea una cuenta."

# EJEMPLO API WEATHER MAP

- ❑ Generar una clave en la seccion API keys



The screenshot shows the OpenWeather API dashboard. At the top, there's a navigation bar with the OpenWeather logo, a search bar, and links to Guide, API, Dashboard, Marketplace, Pricing, and My account. A green notification banner states: "We have sent the confirmation link to [daniellopezlozano@gmail.com](mailto:daniellopezlozano@gmail.com). Please check your email." Below this, a horizontal menu includes New Products, Services, API keys (which is underlined), Billing plans, Payments, and Block logs. A light blue box below the menu says: "You can generate as many API keys as needed for your subscription. We accumulate the total usage." The main content area displays a table of API keys.

| Key                              | Name    | Status | Actions   |
|----------------------------------|---------|--------|---|
| 1eb90fe7a37a9e683d9f8acff01f6871 | Default | Active | <input checked="" type="checkbox"/>   |
| a35b61cbab5a50e899d7770b900b270d | nueva   | Active | <input checked="" type="checkbox"/>   |

# EJEMPLO API WEATHER MAP

- ❑ El código con la url construida y muestra la info

```
API_key='1eb90fe7a37a9e683d9f8acff01f6871'  
ciudad=input('Nombre de la ciudad:')  
base_url = "http://api.openweathermap.org/data/2.5/weather?appid="+API_key+"&q="+ciudad"  
print(base_url)  
f=request.urlopen(base_url)  
datos_json=f.read().decode("utf-8")  
datos_diccionario=loads(datos_json)  
print(datos_diccionario)
```

# EJEMPLO API WEATHER MAP

❑ El resultado obtenido:

```
{
  "coord":{
    "lon":-3.25,
    "lat":37.25
  },
  "weather":[
    {
      "id":800,
      "main":"Clear",
      "description":"clear sky",
      "icon":"01d"
    }
  ],
  "base":"stations",
  "main":{
    "temp":293.53,
    "feels_like":292.77,
    "temp_min":292.2,
```



# Bibliografía y enlaces

## APUNTES MANUALES Y TUTORIALES:

- Autor original: Guido van Rossum, Fred L. Drake, Jr "El tutorial de Python". Python Software Foundation 2018
- Documentación de usuario de Odoo  
<https://www.odoo.com/documentation/user/15.0/es/>
- Documentación del desarrollador de Odoo  
<https://www.odoo.com/documentation/15.0/es/>

## VIDEO-TUTORIALES:

- Python For Everybody  
<https://www.youtube.com/watch?v=o0XbHvKxw7Y>
- TechWorld with Nana  
<https://www.youtube.com/watch?v=t8pPdKYpowI>
- Freecodecamp  
<https://www.youtube.com/watch?v=DLikpfc64cA>
- Píldoras informáticas  
<https://www.youtube.com/watch?v=G2FCfQj-9ig&list=PLU8oAlHdN5BlvPxziopYZRd55pdqFwkeS>

# FIN DEL TEMA 4!

Cualquier duda enviar un correo a  
[daniel.lopez@escuelaartegranada.com](mailto:daniel.lopez@escuelaartegranada.com)

CREDITS: This presentation template was created by  
Slidesgo, including icons by Flaticon, and  
infographics & images by Freepik.