

**NTER TECH SERVICES**

**Python – Ejercicios: Día 2**

Contenido

[4. Funciones 3](#_Toc169071986)

[5. Programación Orientada a Objetos 4](#_Toc169071987)

[6. Ficheros 6](#_Toc169071988)

# Funciones

**Ejercicio 1:**

1. ¿Qué es un decorador y cómo se utiliza?
2. ¿Qué es una función lambda?
3. ¿Qué es un *docstring*?
4. Explica el concepto de funciones de primera clase.

**Ejercicio 2:**

En este ejercicio, crearás una función que busque anagramas de una palabra específica dentro de un contenedor de palabra. Un anagrama es una reordenación de las letras de una palabra que forma otra palabra válida.

Define la función **buscar\_anagramas:**

* Esta función debe aceptar una palabra y un contenedor que contiene palabras conocidas. Por ejemplo:

palabra = “listen”

contenedor = ["enlists", "foo", "bar", "inlets", "listen", "silent"]

* Debe devolver una lista de anagramas encontrados en el contenedor:

anagramas = ['inlets', 'listen', 'silent']

**Ejercicio 3: Funciones anónimas**

1. Define una función lambda que calcule el cuadrado de un número
2. Define una función lambda que devuelva True si el cuadrado de un número es mayor que 999 si no devuelve False.
3. Implementa una función en Python que **invierta el orden de las palabras en una cadena** dada, eliminando cualquier espacio adicional:

cadenas = ["el cielo es azul", " hola mundo ", "un buen ejemplo"]

Esta función debe aceptar una cadena como entrada y devolver una cadena con las palabras en orden inverso. **Elimina cualquier espacio adicional antes de invertir el orden** de las palabras. **Utiliza la función filter()** para ello.

Al ejecutar el código, deberías obtener el siguiente resultado:

Entrada: 'el cielo es azul' -> Salida: 'azul es cielo el'

Entrada: ' hola mundo ' -> Salida: 'mundo hola'

Entrada: 'un buen ejemplo' -> Salida: 'ejemplo buen un'

# Programación Orientada a Objetos

**Ejercicio 1:**

1. ¿Cuál es el propósito de la declaración if \_\_name\_\_ == "\_\_main\_\_":?
2. ¿Para qué sirve la función super() en Python?
3. Describe el uso del decorador @staticmethod

**Ejercicio 2:**

1. Cree una clase Vehicle que reciba el nombre, velocidad máxima y el kilometraje.
2. Cree una clase Bus que herede todos los métodos y variables de Vehicle.
3. Cree un método a Vehicle que, recibiendo la capacidad, imprima por pantalla “La capacidad de (nombre) es de (capacidad).
4. Cree el mismo método para Bus en el que por defecto la capacidad sea de 50.
5. Defina la variable owner que tenga el valor “NWorld” para todos los objetos de la clase.

**Ejercicio 3:**

En este ejercicio, deberás implementar una clase Dni en Python que represente un Documento Nacional de Identidad (DNI) español. La clase debe validar el número del DNI y calcular su letra correspondiente:

1. Define la Clase Dni:

* Implementa el método constructor \_\_init\_\_ para inicializar el número de DNI.
* Implementa un método privado \_\_calcular\_letra para calcular la letra del DNI.
* Utiliza propiedades para manejar el acceso y la validación del número de DNI.

1. Validación del número de DNI:

* El número de DNI debe tener exactamente 8 dígitos y ser completamente numérico.
* Si el número de DNI no es válido, asigna el valor '00000000' y una letra especial, por ejemplo, '#'.

1. Cálculo de la letra del DNI:

* La letra del DNI se calcula utilizando el resto de la división del número entre 23.
* Usa la cadena 'TRWAGMYFPDXBNJZSQVHLCKE' para obtener la letra correspondiente.

**Ejercicio 4:**

Define la clase Empleado utilizando el decorador @*dataclass*:

1. La clase debe tener los atributos nombre, edad, ciudad y empresa. La variable empresa debe tener un valor por defecto.
2. Añade un método estático ‘calcular\_salario’. Este método debe calcular el salario de un empleado basado en el salario base, el número de horas extras y el coste de la hora extra.
3. Añade un método de clase ‘obtener\_nombre\_empresa’:
4. Utiliza el método de clase para obtener y mostrar el nombre de la empresa.
5. Utiliza el método estático para calcular el salario de un empleado y muestra el resultado.
6. Crea una instancia de la clase Empleado y muestra su representación.

**Ejercicio 5 - Slots:**

Investiga sobre el uso de ‘\_\_slots\_\_’ en Python y comenta el siguiente código:

Texto

Descripción generada automáticamente

# Ficheros

**Ejercicio 1:**

1. Explica el propósito de la declaración *with*.
2. ¿Cómo puedes realizar operaciones de entrada/salida de archivos en modo binario?
3. ¿Cómo serializas y deserializas objetos en Python?

**Ejercicio 2 - TXT:**

line1 = “Es mi primera línea.”

line2 = “Es mi segunda línea.”

line3 = “Es mi tercera línea.”

line4 = “¡Es el final del fichero!”

1. Crea un fichero vacío llamado primerfichero.txt sin usar with.
2. Crea un segundo fichero vacío llamado segundofichero.txt usando with.
3. Escribe line1 en primerfichero.txt
4. Escribe line2, line3 y line4 en segundofichero.txt cada uno en una línea.
5. Lee primerfichero.txt y añade su contenido al final de segundofichero.txt.

**Ejercicio 3 – CSV:**

En este ejercicio, aprenderás a leer archivos CSV utilizando el módulo csv en Python. Explorarás dos métodos: usando csv.reader y csv.DictReader.

1. Lee el archivo “estudiantes.csv” con csv.reader

Texto, Carta

Descripción generada automáticamente

1. Extrae y muestra la cabecera.
2. Convierte cada fila en un diccionario usando la cabecera como claves y muestra el contenido.
3. Abre el mismo archivo CSV y utiliza csv.DictReader para leer el archivo.
4. Muestra la información de cada fila en un formato legible.

**Ejercicio 4 – JSON:**

1. Carga los datos de los productos existentes desde el archivo “productos.json”.

[

{

"nombre": "Monitor",

"cantidad": 20,

"precio": 249.99

},

{

"nombre": "Teclado",

"cantidad": 50,

"precio": 39.99

},

{

"nombre": "Disco Duro Externo",

"cantidad": 15,

"precio": 129.99

}

]

1. Implementa la función agregar\_producto que agregue un producto al inventario.
2. Implementa la función escribir\_fichero. Esta función toma el inventario como argumento y lo escribe en un archivo JSON llamado ‘inventario.json’.Formatéalo con una indentación de 4 espacios para que sea más legible.