Universidad Pública de Navarra
Curso 2024/2025

Informática - Práctica de laboratorio 08

Programación VIII: Ficheros de textos

1. Normas de Entrega

Todas las prácticas deberán entregarse siguiendo una convención de nombres y un formato específico. En el caso de las prácticas de Python, se solicitará un script de Python para cada ejercicio, generando así varios archivos de texto por práctica.

Todas las prácticas se desarrollarán utilizando el IDE `spyder`. `Spyder` es un entorno científico de código abierto y gratuito diseñado para científicos, ingenieros y analistas de datos. Ofrece una combinación única de funcionalidades avanzadas para edición, análisis, depuración y perfilado, con capacidades excelentes para exploración de datos, ejecución interactiva, inspección profunda y visualización de paquetes científicos. Puedes acceder a él aquí: `https://www.spyder-ide.org/`.

1.1. Nombre del Archivo

Cada ejercicio de programación debe estar codificado en un archivo independiente, es decir, cada ejercicio será un archivo separado. A menos que se indique lo contrario, la convención de nombres será *Ejercicio_YY.py*, donde *XX* será el número de la práctica y *YY* el número del ejercicio en el documento de la práctica. Por ejemplo, el primer ejercicio de la práctica cero llevará el nombre *Ejercicio_01.py*.

Para entregar los ejercicios, debes acceder a la sección del curso en mi aula virtual (*20XX_0_501103_91_G*). Allí, en la columna izquierda, aparecerá la sección *Tareas*.

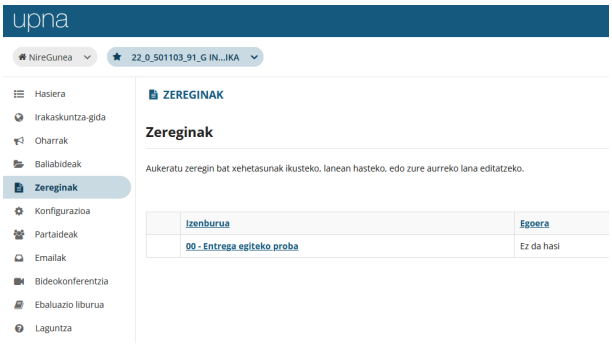

Figura 1: Primer paso para entregar las tareas.

En cada entrega aparecerá una tarea y podrás entregar los ejercicios. También tendrás la opción de escribir un mensaje junto con la entrega. Para cada práctica, se solicitarán diferentes ejercicios, y deberás subir un archivo para cada ejercicio (ver imagen 2).

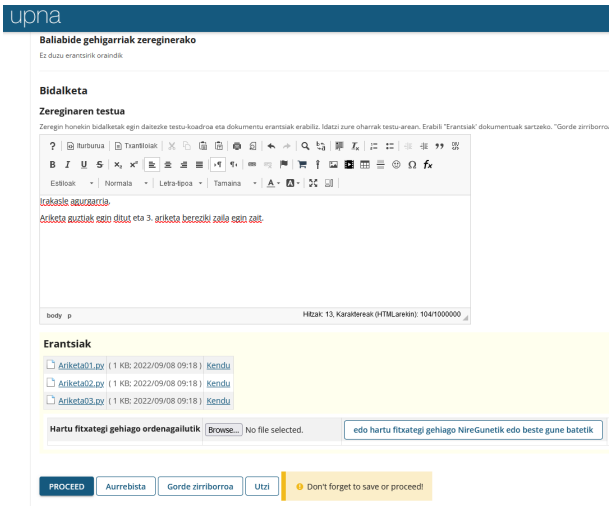

Figura 2: Primer paso para entregar las tareas.

2. Normas de Desarrollo

Con cada práctica tendrás una chuleta (LabXXPR_CheatSheet). En la chuleta encontrarás un resumen del material trabajado en clase. Por ejemplo, en Lab01PR_CheatSheet, se presentarán los elementos necesarios para realizar Lab01PR. Se recomienda imprimir la chuleta o, al menos, tenerla a la vista mientras desarrollas el programa. La CheatSheet contendrá tipos de datos, operadores, estructuras y funciones trabajadas en clase. Cualquier elemento de Python utilizado fuera de estos será evaluado con una nota de 0 en los evaluables.

Al desarrollar los programas, debes seguir las siguientes normas

1. Los nombres de las variables deben ser descriptivos
   - Todos en minúsculas
   - Los nombres compuestos por más de una palabra se separarán con '_'
2. Todos los programas deben tener una cabecera obligatoria
   - La primera línea de comentario debe ser `# python script`
   - En la segunda línea debe aparecer `# Autor: tu nombre`, donde `tu nombre` es el nombre del estudiante
   - En la tercera línea debe aparecer `# Descripción: descripción del programa`, donde `descripción del programa` es una frase o párrafo que describa el programa
3. No se deben usar funciones que no aparezcan en la CheatSheet.

3. Lista de ejercicios

Ejercicio 1. Crea un archivo y escribe en él el texto `Hola, este es un archivo de prueba`.

Ejercicio 2. Crea un programa que lea el contenido de un archivo llamado *'archivo.txt'* y lo muestre en la pantalla.

Ejercicio 3. Crea un programa que añada nueva información a un archivo existente. Por ejemplo, puede añadir la línea `¡Nueva información!`.

Ejercicio 4. Crea un programa que solicite la dirección de un archivo desde el teclado. Luego, el programa debe leer el archivo y mostrar toda la información que contiene en la pantalla.

Ejercicio 5. Crea un programa que solicite la dirección de un archivo y dos números enteros desde el teclado. Luego, el programa debe abrir y leer el archivo, y mostrar únicamente el rango de líneas entre los dos números enteros proporcionados por el usuario. Ten en cuenta que el usuario puede ingresar números de línea que no estén presentes en el archivo.

Ejercicio 6. Crea un programa que lea la dirección de un archivo. Luego, el programa debe abrir el archivo, leerlo y hacer una copia del mismo. En la primera línea de la copia, las dos primeras palabras deben estar en orden inverso; si hay solo una palabra o ninguna, la línea debe permanecer sin cambios. Es decir, si la primera línea del archivo original es `uno dos`, en la copia debe ser `dos uno`. El nombre de la copia debe ser el nombre original con el sufijo `-output`.

Nota: El sufijo `-output` no es una extensión del archivo. Por ejemplo, si el nombre del archivo original es `hola.txt`, el nombre del archivo resultante debe ser `hola-output.txt`.

⊡

Ejercicio 7. Crea un nuevo archivo y escribe una lista de números del 1 al 10, con cada número en una línea separada.

⊡

Ejercicio 8. Crea una función que lea un archivo con números y calcule la suma de todos los números. Cada línea del archivo contendrá un solo número entero. La función debe tomar la dirección del archivo como argumento y devolver la suma de todos los números.

⊡

Ejercicio 9. Escribe un programa que solicite al usuario la dirección de un archivo. El programa debe calcular y mostrar en pantalla el número de palabras, el número de signos de puntuación y el número de dígitos presentes en el texto del archivo. Los signos de puntuación a tener en cuenta son "`.,-¿?!¡`". El archivo puede contener más de una línea de texto, y puedes usar la función `join` para manejar este aspecto.

⊡

Ejercicio 10. Escribe un programa que solicite al usuario la dirección de un archivo a través del teclado. El programa debe leer el texto del archivo y mostrar todas las palabras que aparecen en él. Cada palabra debe aparecer solo una vez y debe mostrarse ordenada por el número de letras, de mayor a menor. Las palabras declinadas se considerarán palabras nuevas, pero ten en cuenta que las letras mayúsculas y minúsculas deben ser identificadas como iguales.

Nota: Las palabras con el mismo número de letras no necesitan estar ordenadas alfabéticamente.

□

Ejercicio 11. Escribe un programa que solicite al usuario la dirección de un archivo y un número entero (`zut`) a través del teclado. El programa debe leer el texto del archivo y mostrarlo en pantalla. El valor de `zut` definirá el número máximo de palabras que puede haber en cada línea cuando se muestra el texto. Es decir, el texto puede mostrarse en tantas líneas como se necesite, pero cada línea puede tener como máximo `zut` palabras.

Nota: Ten en cuenta que el texto puede contener menos palabras que el valor de `zut`, por lo que es posible que el texto se muestre en una sola línea.

Ejemplo: Si el texto en el archivo es `Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc pharetra felis nec lorem lobortis, eget tristique magna pharetra. Sed vitae neque suscipit, aliquet nulla a, sagittis augue.` y el valor de `zut` es `6`, el programa debe mostrar:

```
Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Nunc pharetra felis nec
lorem lobortis, eget tristique magna pharetra.
Sed vitae neque suscipit, aliquet nulla
a, sagittis augue.
```

□