Universidad Pública de Navarra
Curso 2024/2025

Informática - Práctica de laboratorio 05
Programación V: Estructuras iterativas

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*.

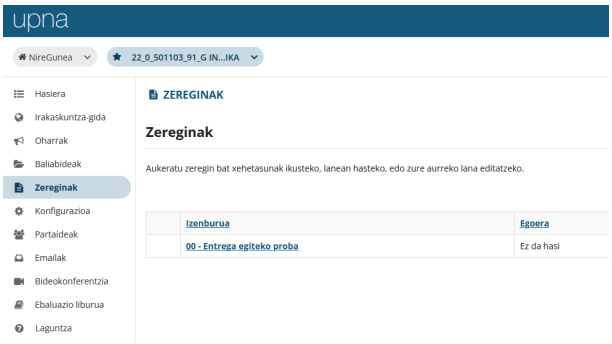

Figura 1: Primer paso para entregar las tareas.

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).

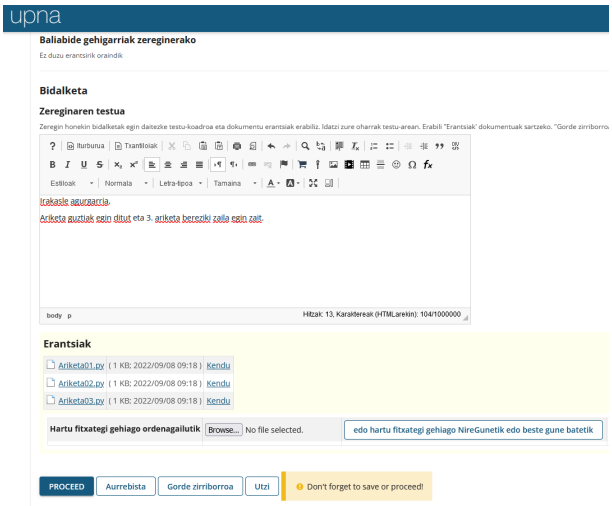

Figura 2: Primer paso para entregar las tareas.

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. Escribe un programa que solicite dos números enteros y positivos desde el teclado. El script debe mostrar la secuencia de números entre el menor y el mayor.

Ejemplo: Si los números introducidos por el usuario son `7` y `3`, el programa debería mostrar el mensaje `3 4 5 6 7`.

⊡

Ejercicio 2. Escribe un programa que solicite un número entero y positivo en el rango $[1, 10]$. Luego, debe mostrar los primeros diez elementos de la tabla de multiplicar del número introducido. Para esta tarea, debes usar la estructura `for` y la función `range`.

Ejemplo: Si el valor introducido desde el teclado es `4`, el programa debería mostrar en pantalla `4 8 12 16 20 24 28 32 36 40`.

⊡

Ejercicio 3. Escribe un programa que solicite dos números enteros y positivos desde el teclado. El script debe mostrar la secuencia de números entre el mayor y el menor.

Ejemplo: Si los números introducidos por el usuario son `7` y `3`, el programa debería mostrar el mensaje `7 6 5 4 3`.

⊡

Ejercicio 4. Escribe una función que solicite un número real y positivo. Si el usuario introduce un número positivo, la función debe devolver el número; de lo contrario, debe mostrar `Valor incorrecto` y volver a solicitar el número. El usuario tendrá 5 intentos para ingresar el número; si en el 5º intento el número es incorrecto, debe mostrar el mensaje `¡Vete al diablo!` y devolver `None`. Para realizar esta tarea no se debe usar la estructura `while` y solo se puede usar una `if`.

Ejemplo: Si el usuario introduce el número `1`, no repetirá la solicitud y mostrará `Programa terminado.`. Si el usuario introduce `-1`, `-61`, `-8`, `-3`, `-2`, el mensaje final del programa será `¡Vete al diablo!. Programa terminado.`.

Nota: Cualquier estructura iterativa puede ser detenida con la palabra reservada `break`. Al ejecutar esta palabra clave, Python ejecutará la lista de instrucciones que sigue (no las internas) a la estructura.

⊡

Ejercicio 5. Escribe una función que solicite un número real y positivo. Si el usuario introduce un número positivo, la función debe devolver el número; de lo contrario, debe mostrar `Valor incorrecto` y volver a solicitar el número. El usuario tendrá 5 intentos para ingresar el número; si en el 5º intento el número es incorrecto, debe mostrar el mensaje `¡Vete al diablo!` y devolver `None`. Para realizar esta tarea se debe usar `while` y solo se puede usar una `if`.

Ejemplo: Si el usuario introduce el número `1`, no repetirá la solicitud y mostrará `Programa terminado.`. Si el usuario introduce `-1`, `-61`, `-8`, `-3`, `-2`, el mensaje final del programa será `¡Vete al diablo!. Programa terminado.`.

□

Ejercicio 6. Escribe un programa que determine si un número entero introducido por el usuario es primo o no. Los números primos son aquellos que solo son divisibles por 1 y por sí mismos. Para este ejercicio, debes usar la estructura `for` y no `while`.

Ejemplo: Si el número es 89, el programa debería mostrar `'El número es primo'`.

□

Ejercicio 7. Escribe un *script* que lea una secuencia de números positivos y los almacene en un conjunto. Si el número introducido por el usuario es 0, la aplicación debe mostrar el conjunto y finalizar el programa. En este ejercicio no se puede usar el operador `in`.

Ejemplo: Si la secuencia introducida por el usuario es `1 3 5 3 23 3 4 5 6 0`, el conjunto mostrado tendrá los elementos `1 3 5 23 4 6` (nuevamente, el orden de los resultados no tiene que seguir este orden).

□

Ejercicio 8. Escribe un *script* que lea una secuencia de números enteros y los almacene en una lista. El programa debe finalizar cuando el número introducido sea mayor que el promedio de los números previamente introducidos. Al finalizar, la lista debe mostrarse en pantalla.

Nota: El primer número siempre debe ser aceptado (ya que no se puede calcular el promedio sin números previos).

Ejemplo: Si la secuencia de números introducida por el usuario es `65 4 8 7 42`, los valores en la lista mostrada serán `65 4 8 7`.

□

Ejercicio 9. Escribe un *script* que lea desde el teclado los nombres y números de identificación (ID) de 5 empleados. Utilizando los datos de cada empleado, crea un diccionario. El número de ID debe ser un número entero positivo. Cuando la lista esté completa, la lista de empleados debe mostrarse ordenada por el número de ID (del 1 al 5). Cada empleado debe aparecer en una línea diferente.

Ejemplo: Si los empleados son 'Juan' (ID 45), 'Pedro' (ID 3), 'Antonio' (ID 5), 'Martin' (ID 58), y 'Gonzalo' (ID 1), el mensaje mostrado en pantalla será `1- Gonzalo`, `3- Pedro`, `5- Antonio`, `45- Juan`, y `58- Martin`.

□

Ejercicio 10. Escribe un *script* que solicite al usuario un conjunto de números enteros positivos (el usuario ingresará números hasta que introduzca el valor -1). Luego, la aplicación debe multiplicar todos los elementos del conjunto por el primer número ingresado y mostrar el resultado.

Ejemplo: Si los números introducidos por el usuario son `4`, `1`, `2`, `4`, `5`, `4`, `7`, y `-1`, el resultado mostrado será `{16,4,8,20,28}` (en cualquier orden).

⊡

Ejercicio 11. Escribe un script que solicite un número entero positivo. Luego, debe mostrar en pantalla los números de Fibonacci hasta la cantidad especificada. Recuerda que cada número en la secuencia de Fibonacci es la suma de los dos números anteriores.

Ejemplo: Si el usuario ingresa `7`, la aplicación debe mostrar la secuencia `1 1 2 3 5 8 13`.

Nota: Asegúrate de que el número ingresado sea positivo, el usuario tendrá 5 intentos para ingresar un número positivo.

⊡

Ejercicio 12. Escribe un programa que calcule el máximo común divisor entre dos números. Recuerda que siempre habrá al menos un divisor común, que es `1`.

Ejemplo: Si los valores son `12` y `30`, el máximo común divisor es `6`.

⊡

Ejercicio 13. Genera un número aleatorio en el rango de 1 a 100 y desafía al usuario a adivinarlo. Proporciona pistas como `más alto` o `más bajo` hasta que adivinen el número correcto.

⊡

Ejercicio 14. Escribe un programa que indique si los elementos de un conjunto de números son positivos o negativos. Cuando se ejecute el programa, pedirá el número de valores que se van a ingresar. Luego, solicitará el número de valores especificados por el usuario y, antes de finalizar el programa, deberá mostrar la cantidad de números positivos y negativos ingresados.

Ejemplo: Si el usuario ingresa `4`, `2`, `-5`, `4`, `-7`, el mensaje mostrado en pantalla será `2 números negativos y 3 positivos`. Si el usuario ingresa `3`, `-2`, `4`, `-7`, el mensaje será `2 números negativos y 2 positivos`.

⊡

Ejercicio 15. Escribe un programa que determine si los números ingresados son pares o impares. Después de ingresar un número, el programa preguntará si deseas ingresar otro número.

Ejemplo:

```
Introduce un número: 4
El número ingresado es par.
¿Deseas ingresar otro número? (S/N): S
Introduce un número: 3
El número ingresado es impar.
¿Deseas ingresar otro número? (S/N): N
El programa ha terminado.
```

⊡

Ejercicio 16. Escribe un programa que calcule el mínimo común múltiplo entre dos números. Ten en cuenta que el múltiplo común entre dos números siempre será un múltiplo de ambos números.

Ejemplo: Si los valores son `12` y `30`, el mínimo común múltiplo es `60`.

⊡

Ejercicio 17. Escribe un *script* que solicite al usuario una frase desde el teclado y convierta todos los dígitos en palabras. Por ejemplo, si la frase ingresada es `Eguberritarako 6 opari erosi ditut.`, el script debe mostrar `Eguberritarako seis opari erosi ditut.`. Este ejercicio solo convertirá números de un solo dígito.

⊡

Ejercicio 18. Escribe un *script* que solicite al usuario una frase desde el teclado y convierta todos los dígitos en palabras. Por ejemplo, si la frase ingresada es `Eguberritarako 6 opari erosi ditut.`, el script debe mostrar `Eguberritarako seis opari erosi ditut.`. Al igual que el ejercicio anterior, pero en esta ocasión, si el script encuentra un número con más de un dígito, debe separar cada dígito con el símbolo `-`. Es decir, si aparecen números como `20` o `313`, el script debe mostrar `dos-cero` o `tres-uno-tres`, respectivamente.

⊡