

Informática - Práctica de laboratorio 03

Programación: Estructuras alternativas y Funciones

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](https://www.spyder-ide.org/). [Spyder](https://www.spyder-ide.org/) 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).

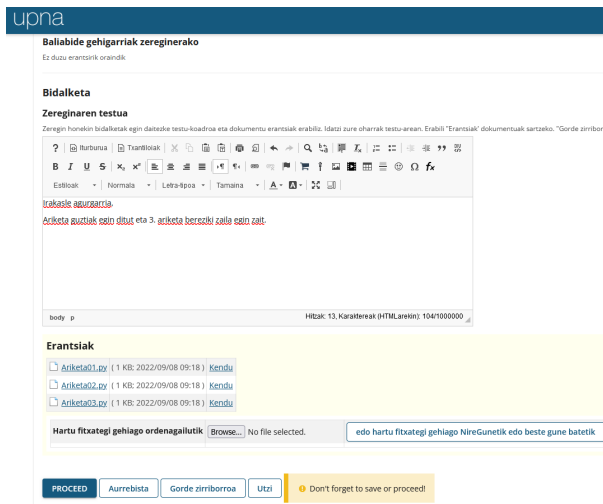


Figura 2: Primer paso para entregar las tareas.

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 una función que reciba el día de la semana como un número (1 para lunes, 2 para martes, etc.). Después, debe mostrar el nombre del día de la semana correspondiente. Si el valor no está en el rango aceptado (1-7), el mensaje a mostrar será **Valor desconocido**.

Código 1: Programa principal del ejercicio.

```
1 # definir funci n
2 ...
3
4 # ejecuci n de la funci n
5 dia_semana(1)
6 dia_semana(4)
7 dia_semana(-5)
```



Ejercicio 2. Escribe una función que reciba un número de mes (1 para enero, 2 para febrero, etc.) y el año. Luego, debe mostrar el número de días de ese mes. Ten en cuenta los años bisiestos para febrero (29 días), puedes utilizar la función **es_bisiesto** del ejercicio 02 para desarrollarlo. Si el valor no está en el rango aceptado (1-12), el mensaje a mostrar será **Valor desconocido**.

Código 2: Programa principal del ejercicio.

```
1 # definir funci n
2 ...
3
4 # ejecuci n de la funci n
5 mes(1,2020)
6 mes(4,2021)
7 mes(-5,-2023)
```



Ejercicio 3. Escribe una función que reciba como números las longitudes de los tres lados de un triángulo. Luego debe indicar si es equilátero (todos los lados iguales), isósceles (dos lados iguales) o escaleno (todos los lados diferentes).

Código 3: Programa principal del ejercicio.

```
1 # definir funci n
2 ...
3
4 # ejecuci n de la funci n
5 tipo_triangulo(2,4,1)
6 tipo_triangulo(-54,3,6)
```

□

Ejercicio 4. Crea una función que simule un cajero automático. Le dará la opción al usuario de retirar dinero, pero la acción solo será aceptada si hay saldo suficiente en la cuenta. Antes de finalizar, mostrará el dinero retirado y el saldo restante en la pantalla.

Nota: El saldo inicial deberá asignarse al comienzo.

□

Ejercicio 5. Crea una función que solicite al usuario su edad y devuelva la categoría correspondiente. Las categorías de edad son: `niño` [0,12), `adolescente` [12,18), `joven` [18,30) o `adulto` [30,Inf). Si el valor recibido es menor que 0, el mensaje a devolver será `Valor desconocido`. Solo puedes utilizar el operador `>` para realizar este ejercicio.

□

Ejercicio 6. Escribe una función que permita al usuario generar una contraseña. El usuario deberá introducir una contraseña y, si cumple con ciertos criterios de seguridad, la función devolverá la contraseña. En caso contrario, permitirá al usuario generar una nueva. Los criterios de seguridad serán que tenga al menos 8 caracteres, una letra y un número.

Nota: Los caracteres que no sean letras o números no serán aceptados para generar la contraseña.

□

Ejercicio 7. Escribe una función que solicite al usuario su edad y devuelva la generación correspondiente. Las categorías generacionales son: `Baby Boomer` [1946 – 1964], `Generación X` [1965 – 1980], `Millennials` [1981 – 1996], `Generación Z` [1997 – 2012], o `Generación Alfa` [2013 – actualidad]. Si el año de nacimiento es anterior a 1946, el mensaje a mostrar será `Valor desconocido`. Para este ejercicio, solo puedes utilizar el operador `<`. Puedes guardar el año actual como una variable constante.

□

Ejercicio 8. Define una función que aplique descuentos. Los argumentos serán el precio original y el porcentaje de descuento, y la función deberá devolver el precio a pagar.

□

Ejercicio 9. Crea un programa que calcule el área de diferentes polígonos. Al iniciar, el programa mostrará un menú de opciones, en el que se podrá elegir entre calcular el área de un triángulo,

un círculo, un cuadrado, un rectángulo o un triángulo equilátero. Después de elegir una opción, solicitará la información necesaria al usuario y mostrará el resultado en pantalla.

Puedes reutilizar las funciones que creaste en el informe anterior, pero deberás adaptarlas a la nueva aplicación. Ten en cuenta que la información necesaria para calcular el área de cada polígono es diferente y deberá ser solicitada al usuario. Las funciones para calcular el área de los diferentes polígonos son:

1. Define una función que calcule el área de un triángulo. El nombre de la función será `area_triangulo` y deberá tomar como argumentos la base y la altura del triángulo y devolver el área.
2. Define una función que calcule el área de un círculo. El nombre de la función será `area_circulo` y deberá tomar como argumento el radio del círculo y devolver el área.
3. Define una función que calcule el área de un cuadrado. El nombre de la función será `area_cuadrado`, tomará como argumento uno de los lados del cuadrado y devolverá el área.
4. Define una función que calcule el área de un rectángulo. El nombre de la función será `area_rectangulo`, tomará como argumentos la longitud y la anchura, y deberá devolver el área.
5. Define una función que calcule el área de un triángulo equilátero. El nombre de la función será `area_triangulo_equilatero`, y deberá tomar como argumento la longitud de un lado y devolver el área.

□

Ejercicio 10. Crea un juego de ordenador de piedra, papel y tijeras. El usuario jugará contra el ordenador, y el programa deberá indicar quién gana cada ronda. Para hacer este ejercicio, tendrás que estructurar los datos. Puedes utilizar la equivalencia 1-piedra, 2-papel, 3-tijeras. La opción que elija el ordenador se seleccionará de manera aleatoria con la librería `random`, aquí tienes el código para la selección del ordenador:

Código 4: El ordenador selecciona aleatoriamente una opción entre Piedra-Papel-Tijeras.

```
1 # importar librerías
2 import random
3 # definir función
4 ...
5
6 # asignar variable
7 opcion_ordenador = random.randint(1, 3)
```

□

Ejercicio 11. Desarrolla una función de conversión de moneda que permita al usuario introducir una cantidad en una moneda y convertirla a otra utilizando las tasas de cambio proporcionadas como argumento.

□

Ejercicio 12. Escribe una función que calcule los descuentos basados en la cantidad de productos comprados. Pedirá al usuario que ingrese la cantidad de productos que desea comprar y el precio unitario. Luego, aplicará diferentes descuentos según la cantidad comprada. Se aplicará un 10

□

Ejercicio 13. Crea un conversor de unidades de tiempo. Al iniciar, mostrará un menú con opciones para convertir entre unidades de tiempo, distancia, temperatura o peso. Luego, pedirá la unidad de origen y el valor. Antes de finalizar, pedirá la unidad de destino y mostrará la respuesta en pantalla, siguiendo el formato `5 horas ->300 minutos`.

Las unidades que aceptará el conversor son las siguientes:

	Origen → Destino
Conversor de tiempo	hora, minuto, segundo → hora, minuto, segundo
Conversor de distancia	metro, kilómetro, milla → metro, kilómetro, milla
Conversor de temperatura	Celsius, Fahrenheit, Kelvin → Celsius, Fahrenheit, Kelvin
Conversor de peso	gramo, libra, kilogramo → gramo, libra, kilogramo

□