**LABORATORIOS CISCO**

**LABORATORIO** 5.1.6.4 

**Tiempo Estimado**

15-25 minutos

**Nivel de dificultad**

Mediano

**Objetivos**

* Mejorar las habilidades del alumno para definir funciones.
* Utilizar excepciones para proporcionar un entorno de entrada seguro.

**Escenario**

Tu tarea es escribir una **función capaz de ingresar valores enteros y verificar si están dentro de un rango especificado**.

La función debe:

* Aceptar tres argumentos: una entrada, un límite inferior aceptable y un límite superior aceptable.
* Si el usuario ingresa una cadena que no es un valor entero, la función debe emitir el mensaje Error: entrada incorrecta, y solicitará al usuario que ingrese el valor nuevamente.
* Si el usuario ingresa un número que está fuera del rango especificado, la función debe emitir el mensaje Error: el valor no está dentro del rango permitido (min..max) y solicitará al usuario que ingrese el valor nuevamente.
* Si el valor de entrada es válido, será regresado como resultado.

**Datos de Prueba**

Prueba tu código cuidadosamente.

Así es como la función debería reaccionar ante la entrada del usuario:

Ingresa un número entre -10 a 10: 100

Error: el valor no está dentro del rango permitido (-10..10)

Ingresa un número entre -10 a 10: asd

Error: entrada incorrecta

Ingresa un número entre -10 a 10: 1

El número es: 1

**LABORATORIO** 5.1.9.18

**Tiempo Estimado**

20-25 minutos

**Nivel de dificultad**

Medio

**Objetivos**

* Mejorar las habilidades del alumno al trabajar con cadenas.
* Utilizar los métodos incorporados de Python para las cadenas.

**Escenario**

Ya sabes como funiona el método split(). Ahora queremos que lo pruebes.

Tu tarea es **escribir tu propia función, que se comporte casi como el método original**split(), por ejemplo:

* Debe aceptar únicamente un argumento: una cadena.
* Debe devolver una lista de palabras creadas a partir de la cadena, dividida en los lugares donde la cadena contiene espacios en blanco.
* Si la cadena está vacía, la función debería devolver una lista vacía.
* Su nombre debe ser misplit().

Usa la plantilla en el editor. Prueba tu código con cuidado.

**Salida esperada**

['Ser', 'o', 'no', 'ser', 'esa', 'es,', 'la', 'pregunta']

['Ser', 'o', 'no', 'ser,esa', 'es', 'la', 'pregunta']

[]

['abc']

[]

**LABORATORIO** 5.1.10.6

**Tiempo Estimado**

30 minutos

**Nivel de dificultad**

Medio

**Objetivos**

* Mejorar las habilidades del alumno para trabajar con cadenas.
* Usar cadenas para representar datos que no son texto.

**Escenario**

Seguramente has visto un *display de siete segmentos*.

Es un dispositivo (a veces electrónico, a veces mecánico) diseñado para presentar un dígito decimal utilizando un subconjunto de siete segmentos. Si aún no sabes lo qué es, consulta la siguiente liga en Wikipedia [artículo](https://en.wikipedia.org/wiki/Seven-segment_display).

Tu tarea es escribir **un programa que puede simular el funcionamiento de un display de siete segmentos**, aunque vas a usar LEDs individuales en lugar de segmentos.

Cada dígito es construido con 13 LEDs (algunos iluminados, otros apagados, por supuesto), así es como lo imaginamos:

# ### ### # # ### ### ### ### ### ###

# # # # # # # # # # # # # #

# ### ### ### ### ### # ### ### # #

# # # # # # # # # # # # #

# ### ### # ### ### # ### ### ###

Nota: el número 8 muestra todas las luces LED encendidas.

Tu código debe *mostrar* cualquier número entero no negativo ingresado por el usuario.

Consejo: puede ser muy útil usar una lista que contenga patrones de los diez dígitos.

**Datos de prueba**

Entrada de muestra:

123

Salida de muestra:

# ### ###

# # #

# ### ###

# # #

# ### ###

Entrada de muestra:

9081726354

Salida de muestra:

### ### ### # ### ### ### ### ### # #

# # # # # # # # # # # # # #

### # # ### # # ### ### ### ### ###

# # # # # # # # # # # # #

### ### ### # # ### ### ### ### #

**LABORATORIO** 5.1.11.6

**Tiempo Estimado**

30-45 minutos

**Nivel de Dificultad**

Difícil

**Prerrequisitos**

Módulo 5.1.11.1, Módulo 5.1.11.2

**Objetivos**

* Mejorar las habilidades del alumno para manipular cadenas.
* Convertir caracteres en código ASCII y viceversa.

**Escenario**

Ya estás familiarizado con el cifrado César, y es por eso que queremos que mejores el código que te mostramos recientemente.

El cifrado César original cambia cada caracter por otro *a* se convierte en *b*, *z* se convierte en *a*, y así sucesivamente. Hagámoslo un poco más difícil y permitamos que el valor desplazado provenga del rango 1 al 25.

Además, dejemos que el código conserve las mayúsculas y minúsculas (las minúsculas permanecerán en minúsculas) y todos los caracteres no alfabéticos deben permanecer intactos.

Tu tarea es escribir un programa el cual:

* Le pida al usuario una línea de texto para encriptar.
* Le pida al usuario un valor de cambio (un número entero del rango 1 al 25, nota: debes obligar al usuario a ingresar un valor de cambio válido (¡no te rindas y no dejes que los datos incorrectos te engañen!).
* Imprime el texto codificado.

Prueba tu código utilizando los datos que te proporcionamos.

**Datos de Prueba**

Entrada Muestra:

abcxyzABCxyz 123

2

Salida Muestra:

cdezabCDEzab 123

Entrada Muestra:

The die is cast

25

Salida Muestra:

Sgd chd hr bzrs

**LABORATORIO** 5.1.11.7

**Tiempo Estimado**

10-15 minutos

**Nivel de Dificultad**

Fácil

**Objetivos**

* Mejorar las habilidades del alumno para manipular cadenas.
* Alentar al alumno a buscar soluciones no obvias.

**Escenario**

¿Sabes qué es un palíndromo?

Es una palabra que se ve igual cuando se lee hacia adelante y hacia atrás. Por ejemplo, "kayak" es un palíndromo, mientras que "leal" no lo es.

Tu tarea es escribir un programa que:

* Le pida al usuario algún texto.
* Compruebe si el texto introducido es un palíndromo e imprima el resultado.

Nota:

* Supón que una cadena vacía no es un palíndromo.
* Trata las letras mayúsculas y minúsculas como iguales.
* Los espacios no se toman en cuenta durante la verificación: trátalos como inexistentes.
* Existe más de una solución correcta: intenta encontrar más de una.

Prueba tu código utilizando los datos que te proporcionamos.

**Datos de Prueba**

Entrada Muestra:

Ten animals I slam in a net

Salida Muestra:

Es un palíndromo

Entrada Muestra:

Eleven animals I slam in a net

Salida Muestra:

No es un palíndromo

**LABORATORIO** 5.1.11.8

**Tiempo Estimado**

10-15 minutos

**Nivel de Dificultad**

Fácil

**Objetivos**

* Mejorar las habilidades del alumno para trabajar con cadenas.
* Convertir cadenas en listas, y viceversa.

**Escenario**

Un anagrama es una nueva palabra formada al reorganizar las letras de una palabra, usando todas las letras originales exactamente una vez. Por ejemplo, las frases "rail safety" y "fairy tales" son anagramas, mientras que "I am" y "You are" no lo son.

Tu tarea es escribir un programa que:

* Le pida al usuario dos textos por separado.
* Compruebe si los textos ingresados son anagramas e imprima el resultado.

Nota:

* Supongamos que dos cadenas vacías no son anagramas.
* Tratar las letras mayúsculas y minúsculas como iguales.
* Los espacios no se toman en cuenta durante la verificación: trátalos como inexistentes.

Prueba tu código utilizando los datos que te proporcionamos.

**Datos de Prueba**

Entrada de Ejemplo:

Listen

Silent

Salida del Ejemplo:

Anagramas

Entrada de Ejemplo:

modern

norman

Salida del Ejemplo:

No son Anagramas

**LABORATORIO** 5.1.11.9

**Tiempo Estimado**

10-15 minutos

**Nivel de Dificultad**

Fácil

**Objetivos**

* Mejorar las habilidades del alumno para trabajar con cadenas.
* Convertir enteros en cadenas, y viceversa.

**Escenario**

Algunos dicen que el *Dígito de la Vida* es un dígito calculado usando el cumpleaños de alguien. Es simple: solo necesitas sumar todos los dígitos de la fecha. Si el resultado contiene más de un dígito, se debe repetir la suma hasta obtener exactamente un dígito. Por ejemplo:

* 1 Enero 2017 = 2017 01 01
* 2 + 0 + 1 + 7 + 0 + 1 + 0 + 1 = 12
* 1 + 2 = 3

*3* es el dígito que buscamos y encontramos.

Tu tarea es escribir un programa que:

* Le pregunta al usuario su cumpleaños (en el formato AAAAMMDD o AAAADMM o MMDDAAAA; en realidad, el orden de los dígitos no importa).
* Da como salida *El Dígito de la Vida* para la fecha ingresada.

Prueba tu código utilizando los datos que te proporcionamos.

**Datos de Prueba**

Entrada de Ejemplo:

19991229

Salida del Ejemplo:

6

Entrada de Ejemplo:

20000101

Salida del Ejemplo:

4

**LABORATORIO** 5.1.11.10

**Tiempo Estimado**

15-20 minutos

**Nivel de Dificultad**

Medio

**Objetivos**

* Mejorar las habilidades del alumno para trabajar con cadenas.
* Emplear el método find() para realizar busquedas dentro de las cadenas.

**Escenario**

Vamos a jugar un juego. Le daremos dos cadenas: una es una palabra (por ejemplo, "dog") y la segunda es una combinación de un grupo de caracteres.

Tu tarea es escribir un programa que responda la siguiente pregunta: **¿Los caracteres que comprenden la primera cadena están ocultos dentro de la segunda cadena?**

Por ejemplo:

* Si la segunda cadena es "vcxzxduybfdsobywuefgas", la respuesta es si.
* Si la segunda cadena es "vcxzxdcybfdstbywuefsas", la respuesta es no (ya que no están las letras "d", "o", "g", ni en ese orden).

Consejos:

* Debes usar las variantes de dos argumentos de las funciones pos() dentro de tu código.
* No te preocupes por mayúsculas y minúsculas.

Prueba tu código utilizando los datos que te proporcionamos.

**Datos de Prueba**

Entrada de Ejemplo:

donor

Nabucodonosor

Salida del Ejemplo:

Si

Entrada de Ejemplo:

donut

Nabucodonosor

Salida del Ejemplo:

No

**LABORATORIO** 5.1.11.11

**Tiempo Estimado**

60 minutos

**Nivel de Dificultad**

Difícil

**Objetivos**

* Mejorar las habilidades del estudiante para trabajar con cadenas y listas.
* Convertir cadenas en listas.

**Escenario**

Como probablemente sabes, *Sudoku* es un rompecabezas de colocación de números jugado en un tablero de 9x9. El jugador tiene que llenar el tablero de una manera muy específica:

* Cada fila del tablero debe contener todos los dígitos del 0 al 9 (el orden no importa).
* Cada columna del tablero debe contener todos los dígitos del 0 al 9 (nuevamente, el orden no importa).
* Cada subcuadro de 3x3 de la tabla debe contener todos los dígitos del 0 al 9.

Si necesitas más detalles, puedes encontrarlos [aquí](https://en.wikipedia.org/wiki/Sudoku).

Tu tarea es escribir un programa que:

* Lea las 9 filas del Sudoku, cada una con 9 dígitos (verifica cuidadosamente si los datos ingresados son válidos).
* Da como salida Si si el Sudoku es válido y No de lo contrario.

Prueba tu código utilizando los datos que te proporcionamos.

**Datos de Prueba**

Entrada de Muestra:

295743861

431865927

876192543

387459216

612387495

549216738

763524189

928671354

154938672

Salida de la Muestra:

Yes

Entrada de Muestra:

195743862

431865927

876192543

387459216

612387495

549216738

763524189

928671354

254938671

Salida de la Muestra

No

**LABORATORIO** 6.1.9.15

**Tiempo Estimado**

30 minutos

**Nivel de dificultad**

Medio

**Objetivos**

* Mejorar las habilidades del estudiante al operar con la lectura archivos.
* Utilizar colecciones de datos para contar datos numerosos.

**Escenario**

Un archivo de texto contiene algo de texto (nada inusual) pero necesitamos saber con qué frecuencia aparece cada letra en el texto. Tal análisis puede ser útil en criptografía, por lo que queremos poder hacerlo en referencia al alfabeto latino.

Tu tarea es escribir un programa que:

* Pida al usuario el nombre del archivo de entrada.
* Lea el archivo (si es posible) y cuente todas las letras latinas (las letras mayúsculas y minúsculas se tratan como iguales).
* Imprima un histograma simple en orden alfabético (solo se deben presentar recuentos distintos de cero).

Crea un archivo de prueba para tu código y verifica si tu histograma contiene resultados válidos.

Suponiendo que el archivo de prueba contiene solo una línea con:

aBc

el resultado esperado debería verse de la siguiente manera:a -> 1

b -> 1

c -> 1

**Tip**:

Creemos que un diccionario es un medio perfecto de recopilación de datos para almacenar los recuentos. Las letras pueden ser las claves mientras que los contadores pueden ser los valores.

**LABORATORIO** 6.1.9.16

**Tiempo Estimado**

15-20 minutos

**Nivel de dificultad**

Medio

**Prerrequisitos**

05\_9.15.1

**Objetivos**

* Mejorar las habilidades del estudiante para operar con archivos en modo (lectura/escritura).
* Emplear lambdas para cambiar el ordenamiento.

**Escenario**

El código anterior necesita ser mejorado. Está bien, pero tiene que ser mejor.

Tu tarea es hacer algunas enmiendas, que generen los siguientes resultados:

* El histograma de salida se ordenará en función de la frecuencia de los caracteres (el contador más grande debe presentarse primero).
* El histograma debe enviarse a un archivo con el mismo nombre que el de entrada, pero con la extensión '.hist' (debe concatenarse con el nombre original).

Suponiendo que el archivo de prueba contiene solo una línea con:

cBabAa

El resultado esperado debería verse de la siguiente manera:a -> 3

b -> 2

c -> 1

**Tip**:

Emplea una lambda para cambiar el ordenamiento.

**LABORATORIO** 6.1.9.17

**Tiempo Estimado**

30 minutos

**Nivel de dificultad**

Medio

**Objetivos**

* Mejorar las habilidades del alumno para operar con archivos en modo lectura.
* Perfeccionar las habilidades del estudiante para definir y usar excepciones y diccionarios autodefinidos.

**Escenario**

El profesor Jekyll dirige clases con estudiantes y regularmente toma notas en un archivo de texto. Cada línea del archivo contiene 3 elementos: el nombre del alumno, el apellido del alumno y el número de puntos que el alumno recibió durante ciertas clases.

Los elementos están separados con espacios en blanco. Cada estudiante puede aparecer más de una vez dentro del archivo del profesor Jekyll.

El archivo puede tener el siguiente aspecto:

John Smith 5

Anna Boleyn 4.5

John Smith 2

Anna Boleyn 11

Andrew Cox 1.5

Tu tarea es escribir un programa que:

* Pida al usuario el nombre del archivo del profesor Jekyll.
* Lea el contenido del archivo y cuenta la suma de los puntos recibidos para cada estudiante.
* Imprima un informe simple (pero ordenado), como este:

Andrew Cox 1.5

Anna Boleyn 15.5

John Smith 7.0

Nota:

* Tu programa debe estar completamente protegido contra todas las fallas posibles: la inexistencia del archivo, el vacío del archivo o cualquier falla en los datos de entrada; encontrar cualquier error de datos debería causar la terminación inmediata del programa, y lo erróneo deberá presentarse al usuario.
* Implementa y usa tu propia jerarquía de excepciones: la presentamos en el editor; la segunda excepción se debe generar cuando se detecta una línea incorrecta y la tercera cuando el archivo fuente existe pero está vacío.

**Tip**:

Emplea un diccionario para almacenar los datos de los estudiantes.