Desafío - Ciclos

- Para realizar este desafío debes haber estudiado previamente todo el material disponible en el LMS correspondiente a la unidad.
- Una vez terminado el desafío, comprime la carpeta que contiene el desarrollo de los requerimientos solicitados y sube el *.zip* en el LMS.
- Puntaje total: 10 puntos.
- Desarrollo desafío: Individual o Grupal.
- Tiempo: 120 minutos

Capítulos

- Ciclo While.
- Ciclo For.
- Python Comprehensions

Habilidades a evaluar

- Codifica un programa en Python utilizando ciclos de instrucciones iterativas combinadas con sentencias if/elif/else para resolver un problema.
- Codifica un programa en Python utilizando diagramas de flujo con iteraciones para la implementación de un algoritmo que resuelve un problema.
- Codifica un programa en Python utilizando ciclos anidados y condiciones de salida para resolver un problema.

Instrucciones:

El siguiente desafío presenta 4 casos en los que se requiere emplear Ciclos para su resolución:

- Primeros Auxilios
- Filtrado Compacto
- Fuerza Bruta
- Módulo 11

Cada desafío presentará requerimientos independientes de modo de obtener un programa con su resolución.

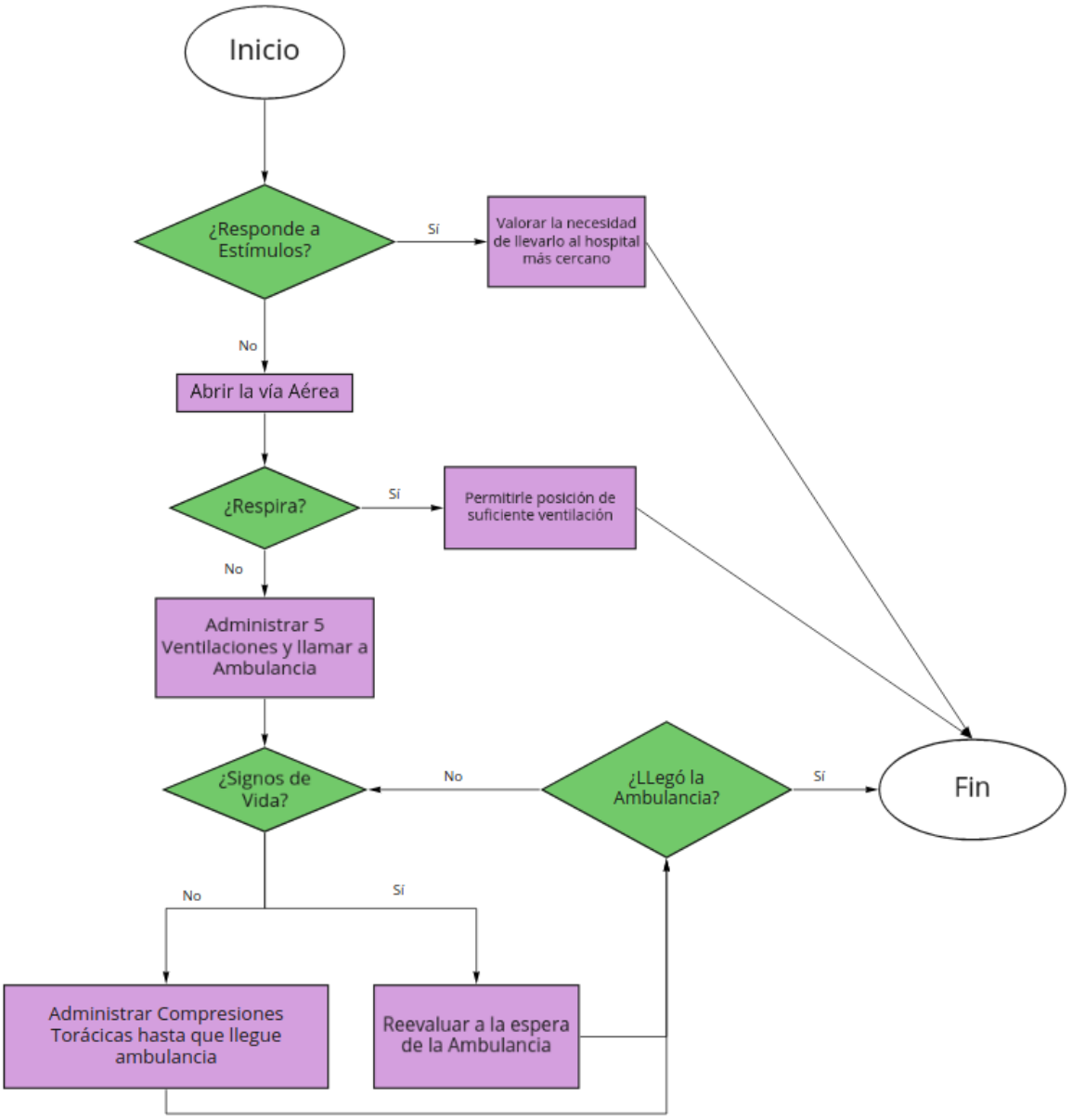
Primeros Auxilios

En cualquier momento puede haber una emergencia y hay que estar preparados ¿sabrías cómo reaccionar en caso de que alguien necesite de primeros auxilios?

Es muy probable que mucha gente no conozca cuáles son los pasos a seguir en caso de emergencia. Es por eso que se le solicita construir una aplicación que permita indicar los pasos a seguir ante una emergencia. Debido a que no se espera que usted sea un experto en el tema se le provee de un diagrama que explica las distintas instancias a la que se está sometido durante una emergencia.

II

I



Se requiere la construcción de una aplicación interactiva `primeros_auxilios.py` que entregue los distintos pasos a seguir dependiendo de las respuestas que el usuario entrega en tiempo real.

iltrado compacto

Una empresa provee de los balances del año anterior en un diccionario como se muestra a continuación:

```
ventas = {
   "Enero": 15000,
   "Febrero": 22000,
   "Marzo": 12000,
   "Abril": 17000,
   "Mayo": 81000,
   "Junio": 13000,
   "Julio": 21000,
   "Agosto": 41200,
   "Septiembre": 25000,
   "Octubre": 21500,
   "Noviembre": 91000,
   "Diciembre": 21000,
}
```

Se solicita devolver un informe resumido que exponga los meses que superan un cierto umbral. El programa `mayor_a.py` debe retornar un diccionario con el mes y el valor asociado siempre y cuando superen el umbral especificado. Ejemplo:

```
python mayor_a.py 40000
```

```
{'Mayo': 81000, 'Agosto': 41200, 'Noviembre': 91000}
```

Fuerza bruta

¿Qué tan seguro es tu password? ¿Intentemos hackear un password? Mediante el siguiente desafío se busca utilizar un algoritmo muy sencillo, llamado fuerza bruta para determinar cuántos intentos son necesarios para encontrar combinaciones numéricas en minúscula.

Para ello se ingresará un password oculto. Este password puede contener sólo combinaciones de letras y se requiere determinar su seguridad. Un mayor número de intentos implica un password más seguro:

El programa `fuerza_bruta.py` debe intentar todas las combinaciones de letras posibles, en orden alfabético, hasta que la combinación de letras sea igual a la de la contraseña indicada. Deberá hacer este proceso letra por letra, de izquierda a derecha.

Consideraciones:

- Utilizar `from string import ascii_lowercase`
  - `ascii_lowercase` es un string con todas las letras del abecedario en minúsculas (sin la ñ).
- No considerar la ñ.
- Considera mayúsculas y minúsculas como una misma letra.
- Se considera "**intento**" cada vez que se compara una letra.

Ejemplo:

- Usuario ingresa "abc"
- El computador compara:
  - **a** es igual a **a** => Sí (1 intento), continúa con la siguiente letra.
  - **b** es igual a **a** => No (2 intentos), prueba otra comparación.
  - **b** es igual a **b** => Sí (3 intentos), continúa con la siguiente letra.
  - **c** es igual a **a** => No (4 intentos), prueba con otra comparación.
  - **c** es igual a **b** => No (5 intentos), prueba con otra comparación.
  - **c** es igual a **c** => Sí (6 intentos), continúa con la siguiente letra.
  - No hay más letras. Se adivinó la palabra en 6 intentos.

NOTA: A modo explicativo se mostrará la contraseña a buscar pero la idea es que ésta se ingrese de manera oculta.

```
python fuerza_bruta.py
Ingrese la contraseña: gato
```

```
La contraseña fue forzada en 43 intentos
```

Módulo 11

Módulo 11 es un algoritmo con el cual se calcula el dígito verificador (o DV, es el número que va después del guión), para los RUT en chile. Este número evita que cualquier persona pueda inventar un RUT aleatoriamente.

Se pide crear un programa en Python llamado `dv.py` que permita ingresar el número de RUT sin puntos ni DV y devuelva el DV correspondiente.

Para ello se provee un ejemplo de cálculo, una especie de pseudo código que explica su funcionamiento:

- Dado el RUT 12 345 678-5.
- Llamaremos ***Número*** a 12 345 678 y *Dígito Verificador* (DV) a 5

1. Se toma la siguiente **serie numérica**: 2, 3, 4, 5, 6, 7.
2. Se multiplicará cada dígito del ***Número*** por su correspondiente en la **serie numérica**, partiendo desde el final del número. En caso de tener más números en la **serie numérica**, ésta se reinicia.
3. Ejemplo de Multiplicación
    - 8 x 2 = 16
    - 7 x 3 = 21
    - 6 x 4 = 24
    - 5 x 5 = 25
    - 4 x 6 = 24
    - 3 x 7 = 21
    - 2 x 2 = 4
    - 1 x 3 = 3
4. La suma de cada todas las multiplicaciones es: 16+21+24+25+24+21+4+3 = 138
5. El siguiente paso es aplicar el módulo 11 (por esto se llama módulo 11). Sólo se toma en cuenta el cuociente entero.
6. 138 / 11 = 12 **el resto (o módulo) es** 6, ya que 11*12 + 6 es 138.
7. Para calcular el dígito verificador se resta el resto (6) a 11, lo cual es: 11 - 6 = **5**
8. **Finalmente se aplican las reglas del algoritmo:**
    - **Si el resultado es igual a 10, el DV es K**
    - **Si el resultado es 11, el DV es cero,**
    - **En cualquier otro caso, no se modifica.**
9. En este caso el DV sigue siendo 5.

Se puede verificar el correcto funcionamiento del programa ingresando su propio número de RUT. El programa se espera que funcione de la siguiente manera:

```
python dv.py
Ingresa tu RUT sin puntos ni dígito verificador: 12345678
```

```
Su dígito verificador es 5
```