EXAMEN FINAL FUNDAMENTOS PROGRAMACIÓN
JUNIO 2024/2025

Normas:

- *Duración prevista del examen: 1:30 h.*
- *El fichero entregado debe ser un fichero válido de Python (NO una sesión interactiva de Jupyter), es decir, debe tener la extensión .py.*
- *Debéis subir todas las funciones de los ejercicios en un solo fichero.*
- *Las funciones implementadas deben respetar la signatura definida en el ejercicio (nombre, parámetros y valores de retorno correctos).* ***Aseguraos de probar esto con el código de prueba que se os entrega en el fichero plantilla.py para evitar errores 'tontos' que perjudiquen vuestra evaluación. NO SE ARREGLARÁ NINGÚN ERROR DURANTE LA REVISIÓN DEL EXAMEN.***
- *La entrega se encuentra abierta en https://pracdlsi.dlsi.ua.es/. Está prohibido acceder a cualquier otra URL durante el examen.*
- *Recordad que la red puede ir lenta, y que el servidor se cierra exactamente a la hora de finalización del examen: reservad al menos 5 minutos al final del examen para la entrega.*
- *No olvidéis indicar al inicio del fichero vuestro* ***dni*** *y nombre mediante un comentario Python*
- *En el cuerpo de las funciones* ***no debe haber ninguna sentencia de tipo input ni print.****Si queréis que el ejercicio incluya vuestro código de prueba, lo debéis poner al final del archivo .py, dentro del bloque, incluido en la plantilla del examen:*

```
if __name__ == "__main__":
        #aquí vuestro código de prueba
```

  - *Notad como tanto name como main van precedidos y sucedidos de DOS guiones bajos.*

- *No entreguéis código que contenga* ***errores sintácticos o código de prueba que haga abortar el corrector (e.g. bucles infinitos) fuera del bloque main****: abortará el corrector y eso supondrá un 0 en el examen.*
- *Para evitar pérdidas de código por bloqueo de los ordenadores del laboratorio,* ***trabajad en la unidad D: o E:*** *(cambia según el ordenador usado). De este modo, si se os reiniciase la máquina por cualquier motivo, no se perderá vuestro trabajo. No te olvides de borrar todo el contenido de esa unidad al acabar el examen. Si decidís trabajar en C:, recordad que, si se bloqueara el ordenador, no podréis recuperar vuestro trabajo, y el profesorado de la asignatura no podrá hacer nada al respecto.*
- *Si se detecta que el código entregado ha sido plagiado o generado con IA se procederá a actuar según el artículo 14 del Reglamento para la Evaluación de los Aprendizajes de la Universidad de Alicante (BOUA 9/12/2015)*
- *Queda explícitamente prohibido llevarse el enunciado del examen ni realizar copia de cualquier fichero entregado. Los alumnos que así lo deseen tendrán acceso a su código cuando se realice la revisión del examen.*
- *También* ***está totalmente prohibida la tenencia de cualquier dispositivo electrónico*** *(teléfonos, relojes inteligentes, etc.) durante el examen. Si se descubre a alguien con un dispositivo de estas características en su poder, se le expulsará automáticamente del examen y se procederá a actuar según el artículo 14 del Reglamento para la Evaluación de los Aprendizajes de la Universidad de Alicante (BOUA 9/12/2015)*
- *Importante:* ***El primer listado de notas que se publique deberá considerarse como meramente orientativo****, ya que el corrector podrá sufrir modificaciones tras las oportunas revisiones en base a las correcciones realizadas sobre vuestro código para garantizar su total fiabilidad. La lista definitiva de calificaciones se hará pública una vez concluido dicho proceso de verificación.*

Ejercicio 1: Análisis de Registros de pHmetría

Una clínica de digestivo monitoriza el pH esofágico de sus pacientes mediante pHmetrías de 24 horas. Los resultados se guardan en ficheros de texto individuales por paciente. Cada fichero registra el nivel de pH, la actividad del paciente (Tumbado, De pie, Comiendo, Durmiendo) y los síntomas reportados (Acidez, Tos, Disfagia, Odinofagia, etc.) en diferentes momentos.

Formato del Fichero de Entrada: Cada fichero contiene múltiples líneas, cada una representando una medición, con el siguiente formato:

```
"MM:SS";pH;Actividad;[Síntoma1,Síntoma2,...]
```

- **"MM:SS"**: String representando el tiempo (Minuto:Segundo) desde el inicio de la prueba. Las comillas son parte del formato.
- **pH**: Valor (de tipo float) del pH medido.
- **Actividad**: String describiendo la actividad del paciente (los valores posibles son `Tumbado`, `De pie`, `Comiendo`,`Durmiendo`).
- **Síntomas**: (Opcional) Uno o más síntomas separados por comas. Si no hay síntomas, esta parte puede estar ausente (aparecerá el ; sin nada detrás).

Ejemplo: fichero phm_ejemplo.txt (notad cómo el fichero puede contener líneas con distintos errores, que habrá que ignorar durante el procesado del fichero).

```
"00:00";7.0;Tumbado;
"00:05";6.7;De pie;
"00:10";-;De pie;Tos
"00:15";7;Comiendo;Acidez
"00:20";3.5;Tumbado;Acidez,Tos

"00:25";6.0;Tumbado;
"00:30";3.8;Comiendo;
```

Se pide:

A) Función de lectura y procesamiento inicial del fichero (3 puntos)

Crea una función `leer_phmetria(ruta_fichero: str) -> dict:`

- **Entrada:** La ruta completa de un fichero de pHmetría.
- **Proceso:**
  - Abre y lee el fichero con encoding=”utf_8”.
  - Parsea cada línea válida (con cuatro elementos, pudiendo ser el último cadena vacía). Extrae el pH (float), la actividad (str) y la lista de síntomas (list[str]). Ignora el campo del timestamp.
  - Maneja posibles errores:
    - `FileNotFoundError` si el fichero no existe -> debe devolver un diccionario vacío.
    - Número incorrecto de campos separados por ':' -> se ignora esa línea y se continúa procesando el resto del fichero
    - pH no numérico (int o float)-> se ignora esa línea y se continúa procesando el resto del fichero
    - Actividad contiene algún valor que no es ninguno de estos valores: Tumbado, De pie, Comiendo, Durmiendo-> se ignora esa línea y se continúa procesando el resto del fichero
  - Los síntomas deben extraerse como una **lista de strings**. Si no hay síntomas, la lista debe estar vacía (`[]`). Asegúrate de eliminar espacios en blanco y cualquier otro carácter no imprimible que pueda existir alrededor de los nombres de los síntomas.
- **Salida:** Devuelve un **diccionario** donde:
  - Las **claves** son los nombres de las actividades (str) encontradas en el fichero.

    - Los **valores** del diccionario son **listas** de **tuplas**. Cada tupla representa una medición para esa actividad y contiene `(ph: float, sintomas: list[str])`.
- **Ejemplo:** Para el fichero de ejemplo anterior, la función debería devolver el siguiente diccionario:

```
{
    'Tumbado': [(7.0,[]), (3.5,['Acidez', 'Tos']), (6.0,[])],
    'De pie': [(6.7,[])],
    'Comiendo': [(7.0, ['Acidez']), (3.8, [])]
}
```

B) Función de Análisis de Frecuencia de Actividad Durante Reflujo Ácido (2 puntos)

Crea una función `frecuencia_reflujo_por_actividad(datos_ph: dict) -> dict:`

- **Entrada:** El diccionario devuelto por la función `leer_phmetria (tenéis un diccionario de ejemplo en la plantilla para poder desarrollar esta función de manera independiente a la parte A)`. Si el diccionario de entrada está vacío, el diccionario de salida también lo estará.
- **Proceso:**
    - Define "reflujo ácido" como cualquier medición con **pH < 4.0**.
    - Para cada actividad (Comiendo, Durmiendo, etc) presente en el diccionario, cuenta cuántas de sus mediciones tienen un pH < 4.0. Cuenta también cuántas mediciones hay en total para cada actividad.
    - Calcula, para cada actividad, las veces que ha aparecido el reflujo ácido (*frecuencia de reflujo ácido por actividad)*. Esto es: Número de mediciones con pH < 4.0 para la Actividad X / Número total de mediciones de esa actividad) * 100. El número debe estar redondeado a dos decimales.
- **Salida:** Devuelve un **diccionario** donde:
    - Las **claves** son los nombres de las actividades (str).
    - Los **valores** son la frecuencia calculada (float) de reflujo en esa actividad, redondeada a 2 decimales.
- **Ejemplo:** Usando el resultado del ejemplo de la Parte A, la función debería devolver:

```
{'Tumbado': 33.33, 'De pie': 0.0, 'Comiendo': 50.0}
```

Ejercicio 2 (3 puntos): Evaluación de riesgo renal

Estás desarrollando un módulo para un sistema de **monitorización remota de pacientes con insuficiencia renal crónica**. Este módulo recibe **mediciones diarias** de **creatinina sérica** (en mg/dL) y debe generar un informe que evalúe el estado del paciente según los valores y determine el **riesgo de progresión de enfermedad renal**.

Los valores de creatinina se interpretan así (según la edad y el sexo del paciente):

Sexo	Edad	Creatinina Normal (mg/dL)
Hombre	< 60 años	0.6 – 1.2 (ambos incluidos)
Hombre	≥ 60 años	0.7 – 1.3 (ambos incluidos)
Mujer	< 60 años	0.5 – 1.1 (ambos incluidos)
Mujer	≥ 60 años	0.6 – 1.2 (ambos incluidos)

Estos valores se guardan en el siguiente diccionario (que tienes también en la plantilla del examen):

```
dcreatinina_normal = {
    "M": {
        "<60": (0.6, 1.2),
        ">=60": (0.7, 1.3)
    },
```

```
    "F": {
        "<60": (0.5, 1.1),
        ">=60": (0.6, 1.2)
    }
}
```

Crea una función evaluar_riesgo_renal (edad, sexo, lista_valores, drangos) que cumpla con los siguientes requisitos:

1. Reciba como parámetro la edad (un entero), el sexo (‘M’ o ‘F’), la lista de valores diarios de creatinina (en mg/dL, mínimo 5 mediciones) y el diccionario con la interpretación de las condiciones. **Por defecto**, el valor de drangos será el diccionario dcreatinina_normal.

   Nota: Esto quiere decir que yo le podría pasar como argumento a la función cualquier otro diccionario con la misma estructura pero distintos límites para los rangos, o podría no pasarle nada, en cuyo caso asume que el diccionario para calcular el resultado es el diccionario dcreatinina_normal de la plantilla.

   - Si la edad no es un entero >= 0, la función devolverá -1.
   - Si el sexo no es un string con los valores M o F, la función devolverá -2
   - Si la lista de valores diarios de creatinina contiene algún valor que no es entero>0 ni float >0, **ese valor se ignorará**.
   - Si la lista final (después de descartar valores no válidos), tiene menos de cinco valores, la aplicación devolverá -3.
2. Si todo es correcto, la función calculará el promedio de creatinina para ese paciente **redondeado a DOS decimales.**
3. A continuación, comparará el promedio con los valores límite del diccionario según edad y sexo.
4. Por último, devolverá un **diagnóstico personalizado**:
   - 1 si el nivel de ese paciente está dentro del rango normal para su sexo y edad
   - 2 si el nivel de creatinina es demasiado bajo para su sexo y edad
   - 3 si el nivel de creatinina es demasiado alto para su sexo y edad

Fíjate en que la función devuelve un entero positivo (1,2 o 3) si es posible calcular el nivel del paciente, y un entero negativo (-1, -2 o -3) si hay algún error.

Ejemplos de llamadas:

```
codigo_nivel=evaluar_riesgo_renal(50, 'F', [0.9, -1.0, "x", 1.0, 0.8, 1.0],
dcreatinina_normal) #-3


codigo_nivel=evaluar_riesgo_renal(63, 'F', [0.9, -1.0, "x", 1.0, 0.8, 1.0,
1.2],dcreatinina_normal) #1
```