Instrucciones

- Tendrás que hacer un commit cuando te lo indique el profesor con el mensaje **checkpoint** y un número.
- **Al final del examen** harás un push en tu repositorio remoto.
- En la entrega del examen, tendrás que copiar el link de tu repositorio.
- Ante dos soluciones que aparenten ser copia, se calificarán a los implicados con un 0. En determinados casos, se podrá realizar una entrevista personal, pudiendo preguntar problemas diferentes.

Ejercicio 1 (7 puntos)

ANTES QUE NADA: Descarga los ficheros `wordle.py`, `palabras_extended.txt` y `palabras_reduced.txt` de la plataforma y cópialos en el directorio donde tienes el repositorio.

Existe un juego muy popular entre los alumnos de GTI llamado Wordle, que consiste en:

- Adivinar una palabra de 5 letras en 6 intentos.
- Cada intento debe ser una palabra de 5 letras.
- Después de cada intento, se marcan las letras que se han acertado (en mayúsculas las que coincidan en la misma posición y en minúsculas las que estén en la palabra pero en una posición distinta).

En este ejercicio se pide que implementes una versión simplificada de este juego. A continuación, tienes una traza de lo que debería hacer el programa:

```
Palabra a adivinar: COBRE
Introduce una nueva palabra: GAMBA
---b-
Introduce una nueva palabra: CABRA
C-BR-
Introduce una nueva palabra: CEBRA
CeBR-
Introduce una nueva palabra: COBRE
COBRE
HAS GANADO!!
```

Para implementar esta versión del juego, en el fichero **wordle.py** ya tienes un código principal que hace el bucle. **Lo que debes implementar son las siguientes funciones**, cuya explicación detallada está descrita en el docstring de cada una:

- `choose_secret` que elige una palabra aleatoria de un fichero **(1.5 puntos)**. Utiliza el fichero **palabras_reduced.txt** para probar el funcionamiento.
- `compare_words` que compara dos palabras **(1 punto)**.
- `print_word` que crea un string según las letras acertadas **(1 punto)**.

Además, para optar a más nota, se piden las siguientes funcionalidades:

- Implementar la función `choose_secret_advanced` que recibe el nombre de un fichero (utiliza **palabras_extended.txt**) y filtra solo las palabras de 5 letras que no tengan acentos (á,é,í,ó,ú). De estas palabras, se seleccionan 15 aleatoriamente sin repetición y una de estas 15, se selecciona aleatoriamente como palabra secreta (**1 punto**). Modifica el **main** para llamar a esta función.
- Implementar la función `check_valid_word` que recibe una lista de palabras y va preguntando al usuario que introduzca una palabra hasta que introduzca una que está en la lista. Deberás modificar el **main** para que la petición de palabras ya no se haga ahí (**1 punto**).
- Control de errores emitiendo un error de tipo **ValueError** desde las funciones y la gestión try/except desde el **main** cuando se produzcan las siguientes situaciones (**1.5 puntos**):
  - Si el fichero recibido por `choose_secret` no tiene palabras.
  - Si el fichero recibido por `choose_secret_advanced` no tiene al menos 15 palabras de 5 letras sin acentos.
  - Si la longitud de las palabras recibidas por `compare_words` no es la misma.
  - Si `same_position` o `same_letter` recibidos por `print_word` no son listas.
  - Si `same_position` o `same_letter` recibidos por `print_word` contienen algún valor negativo o mayor que la longitud de la palabra.

Tiempo estimado: 80 minutos

ENTREGA: Los ficheros y la url de tu repositorio.

Ejercicio 2 (3 puntos)

ANTES QUE NADA: Descarga los ficheros `funciones.py` y `test_funciones.py` de la plataforma y cópialos en el directorio donde tienes el repositorio.

El fichero `funciones.py` contiene 3 funciones diferentes, que se explican en el docstring de cada una. El fichero `test_funciones.py` contiene 3 testing para validar el correcto funcionamiento de las 3 funciones. Mirando ambos ficheros, puedes hacerte una idea clara de lo que se espera que haga cada función.

Si ejecutas el testing, verás que **las 3 funciones fallan**. **NOTA**: Puedes ejecutar el testing de una única función añadiendo su nombre en la llamada, como por ejemplo:

```
python -m pytest -k "test_encontrar_menores"
```

En este ejercicio se pide que **modifiques el fichero** `funciones.py` para corregir los errores que impiden que se ejecuten correctamente los tests. Para ello, **en el mismo código de cada función** de funciones.py deberás **explicar claramente** (1) a qué se debe cada error y (2) cómo se ha solucionado. Añade las explicaciones mediante comentarios en el mismo código.

Tiempo estimado: 40 minutos

ENTREGA: Los ficheros y la url de tu repositorio.