1. Ejercicios (ahora siempre hay que testear)

1. Crea una función generadora de Fibonacci
2. Pasa eslint y prettier por todos tus scripts y mira qué cosas y fallas te salen.
3. Crea un generador del producto de los elementos de un array
4. Crea un generador que devuelva el máximo, el mínimo valor de un objeto iterable (array: [2, 4, 3, 5] -> (2, 2) -> (2,4) -> (2, 4) -> (2, 5)
5. Crea un iterador que devuelva infinitamente el valor de un número más la suma de un valor step cada vez. (valor:10, step:2 -> 10, 12, 14, 16….)
6. Crea un iterador infinito que devuelva todas las keys de un objeto y vuelva a empezar
7. Crea tres excepciones diferentes y un código que las utilice en diferentes puntos.
8. Crea dos excepciones y un código que controle dichas excepciones en diferentes puntos.
9. Crea un iterador de números naturales
10. Crea un iterador de números primos (Ojo!)
11. Crea un iterador de números pares
12. Crea un iterador de números múltiples de 3 impares
13. Intenta crear en un iterador las siguientes sucesiones:
    i. 1, 0, 0, 0, 0, 0, 1, 0, 2, 1, 1, 0,...
    ii. 1, 2, 3, 5, 10, 19, 20, 30, 1000...
    iii. 6, 2, 5, 5, 4, 5, 6, 3, 7,...
    iv. 1, 4, 9, 61, 52, 63, 94...
14. Write a function called divide that takes two parameters: a numerator and a denominator. Your function should return the result of numerator / denominator. However, if denominator is zero you should throw the error, "Attempted to divide by zero."
15. Ejercicio complet de https://education.launchcode.org/intro-to-professional-web-dev/chapters/exceptions/exercises.html:

A teacher has created a `gradeLabs` function that verifies if student programming labs work. This function loops over an array of JavaScript objects that *should* contain a `student` property and `runLab` property.

The `runLab` property is expected to be a function containing the student's code. The `runLab` function is called and the result is compared to the expected result. If the result and expected result don't match, then the lab is considered a failure.

```
function gradeLabs(labs) {
   for (let i=0; i < labs.length; i++) {
      let lab = labs[i];
      let result = lab.runLab(3);
      console.log(`${lab.student} code worked: ${result === 27}`);
   }
 }

let studentLabs = [
   {
      student: 'Carly',
      runLab: function (num) {
         return Math.pow(num, num);
      }
   },
   {
      student: 'Erica',
      runLab: function (num) {
         return num * num;
      }
   }
];

gradeLabs(studentLabs);
```

The `gradeLabs` function works for the majority of cases. However, what happens if a student named their function incorrectly? Run `gradeLabs` and pass it `studentLabs2` as defined below.

```
let studentLabs2 = [
   {
      student: 'Blake',
      myCode: function (num) {
         return Math.pow(num, num);
      }
   },
   {
      student: 'Jessica',
      runLab: function (num) {
         return Math.pow(num, num);
      }
   },
   {
      student: 'Mya',
      runLab: function (num) {
         return num * num;
      }
   }
];

gradeLabs(studentLabs2);
```

Upon running the second example, the teacher gets `TypeError: lab.runLab is not a function`.

Add a `try/catch` block inside of `gradeLabs` to catch an exception if the `runLab` property is not defined. If the exception is thrown, `result` should be set to the text `"Error thrown"`.

16. Create a function called safe_int() that takes a single argument i. If possible, the function converts i to int and returns it. If not possible (i.e. if an Exception occurs), the function returns None.
17. Define a function capitalize_last_name() that accepts as argument a string with a (single) first and a (single) last name, and returns a string in which only the first letter of the first name is uppercase, whereas all letters of the last name are uppercase; in otherwords, 'marisa tomei' becomes 'Marisa TOMEI'. (Tip: use str.split() to split a str into separate words.) If something other than a str object is passed as an argument, the function should raise a TypeError. (Tip: you can use isistance() to check whether an object is of a particular type.) If the str does not consist of exactly two words, the function should raise a ValueError.
18. Localiza el error en el siguiente bloque de código. Crea una excepción para evitar que el programa se bloquee y además explica en un mensaje al usuario la causa y/o solución:

```
lista = [1, 2, 3, 4, 5]
lista[10]
```

19. Realiza una función llamada **agregar_una_vez(lista, el)** que reciba una lista y un elemento. La función debe añadir el elemento al final de la lista con la condición de no repetir ningún elemento. Además si este elemento ya se encuentra en la lista se debe invocar un error de tipo *ValueError* que debes capturar y mostrar un mensaje en su lugar

BIBLIOGRAFÍA

1. https://www.w3resource.com/python-exercises/itertools/index.php, 2022/06/24
2. http://www.rodoval.com/heureka/seriesnum.html, 2022/06/24
3. https://education.launchcode.org/intro-to-professional-web-dev/chapters/exceptions/exercises.html, 2022/06/25
4. https://pythontutorials.eu/basic/exceptions/, 2022/06/25
5. https://docs.hektorprofe.net/python/errores-y-excepciones/ejercicios/, 2022/0625