

## Tareas iniciales

El objetivo de las tareas iniciales es completar la creación de cuentas y configurar el software y servicios necesarios para el desarrollo de la asignatura.

**Tarea 1** Cree una cuenta en *Github*. Github es un servicio de repositorios digitales de software, que usaremos para variadas tareas en la materia, y le servirá para mantener sus soluciones organizadas, así como para colaborar con otros estudiantes. Visite el sitio de Github (<https://github.com>), cree una cuenta y, por supuesto, recuerde sus credenciales de acceso.

**Tarea 2** En su estación de trabajo, ejecute las instrucciones de las páginas 45 a 47 del libro *“Intro to Python for Computer Science and Data Science”* (Deitel & Deitel 2019), para instalar el software necesario para la práctica del curso. Alternativamente, puede instalar Python 3.x junto con entornos de desarrollo de su preferencia, como PyCharm o VisualStudio Code.

## Expresiones, variables y tipos

**Tarea 3** Escriba expresiones en intérprete Python interactivo para convertir temperaturas de Fahrenheit a Celsius y viceversa. Utilice estas expresiones para convertir temperaturas específicas de ejemplo. Compruebe manualmente que los resultados obtenidos son correctos, y chequee tanto precedencia de operadores como el tipo de operaciones (y tipos de datos) utilizados.

**Tarea 4** Escriba programas Python para convertir temperaturas Fahrenheit a Celsius y viceversa. Las temperaturas a convertir deben ser pasadas por el usuario. Utilice tanto entradas interactivas como entradas por línea de comandos.

**Tarea 5** Escriba programas para manipulación de texto: conversión de texto de mayúscula a minúscula, y cálculo de prefijos y sufijos de cierto tamaño. Las entradas para estos programas deben ser pasadas por línea de comandos, y los resultados impresos por pantalla.

## Colecciones y estructuras de control

**Tarea 6** Escriba programas Python para imprimir tablas de conversión de temperaturas Fahrenheit a Celsius y viceversa. Los programas reciben temperatura inicial, temperatura final, y paso (cantidad de grados a avanzar entre una temperatura y otra). Las entradas se deben pasar por línea de comandos y los resultados se deben imprimir en pantalla.

**Tarea 7** Escriba un programa Python interactivo para carga de un diccionario. El programa permite cargar una entrada en el diccionario, buscar el valor correspondiente a una clave, y eliminar el valor correspondiente a una clave. Elija letras para cada opción (comandos), incluyendo una opción para salir. Al salir se debe imprimir el contenido del diccionario. Las claves y valores usados puede elegirlos como parte del diseño de su programa; pueden ser de cualquier tipo, pero no pueden ser ambas strings.

## Programas sobre colecciones

**Tarea 8** Utilizando el programa `read_iris.py` como referencia, escriba un programa Python que lea la información contenida en el archivo `iris_data.txt`, y almacene esta información en una lista de tuplas. Los valores deben estar representados con los tipos de datos que considere más adecuados.

**Tarea 9** Utilizando como base su solución al ejercicio anterior, calcule media, mediana, y moda, de cada uno de los atributos del dataset `iris_data.txt`. Cuál es la especie más común en el dataset? Cuál es el rango de cada uno de los atributos numéricos? Vea la Sección 3.17 de *“Intro to Python for Computer Science and Data Science”* (Deitel & Deitel 2019).

**Tarea 10** Extienda los ejercicios anteriores para organizar los datos del dataset `iris_data.txt` en un diccionario, que separe la información por especie. El diccionario tendrá tres claves (una por cada especie), y los valores serán listas, o conjuntos, de tuplas que corresponden a esa especie.

## Clases

**Tarea 11** Defina una clase `IrisFlower`, con campos para modelar el ancho y largo de pétalos y sépalos. Añada un atributo booleano `especie_conocida`, que indica si la especie es conocida o no, es decir, si ya cuenta con información de especie. Puede añadir métodos y atributos para almacenar la especie, teniendo en cuenta que cuando la especie no es conocida, tal información no formará parte del objeto.

**Tarea 12** Cambie su ejercicio anterior que almacena el contenido del dataset `iris_data.txt` en una lista de tuplas, para que realice este almacenamiento en una lista de objetos de tipo `IrisFlower`.

**Tarea 13** Defina una clase `IrisRandomClassifier`. Esta clase debe contar con un método `classify()`, que toma un objeto de tipo `IrisFlower`, y produce, aleatoriamente, una clasificación para el mismo, es decir, una de las tres especies posibles de flores iris (Iris-virginica, Iris-versicolor, o Iris-setosa).

**Tarea 14** Escriba un programa Python que calcule la tasa de falsos positivos, falsos negativos, positivos verdaderos y negativos verdaderos, generados por el clasificador aleatorio, para el total de flores del dataset `iris_data.txt`. Vuelva a calcular estas métricas, en promedio, para 30 ejecuciones diferentes.

**Tarea 15** Averigüe las definiciones de *precision* y *recall*, y cómo computar éstas a partir de tasa de falsos positivos, falsos negativos, positivos verdaderos y negativos verdaderos. Calcule estos valores en base a los datos obtenidos en el ejercicio anterior, extendiendo el programa del ejercicio anterior.

**Tarea 16** Defina una clase `IrisConstantClassifier`. Esta clase debe contar con un método `classify()`, que toma un objeto de tipo `IrisFlower`, y produce como predicción de su clasificación la especie que más se repite en el dataset. Calcule *precision* y *recall* de este clasificador.

**Tarea 17** Escriba un programa Python que genere aleatoriamente un objeto de tipo `IrisFlower`, usando los valores mínimo y máximo de cada atributo numérico como rango. El objeto generado no tendrá especie conocida. Utilice el predictor aleatorio para predecir especies para estos objetos aleatorios.