# Tecnología Digital 1: Introducción a la Programación
# **Trabajo Práctico 2**

Escuela de Negocios, UTDT - Primer semestre de 2025

El objetivo de este Trabajo Práctico es escribir programas para procesar y efectuar consultas sobre datos públicos de las Pruebas Aprender 2023, como parte de un (imaginario) sistema informático usado para el análisis de resultados en evaluaciones educativas estandarizadas.

Las Pruebas Aprender son un programa nacional de evaluación en Argentina que permite medir los conocimientos de los estudiantes en distintas áreas, como Lengua y Matemática. Los resultados que analizaremos son los obtenidos por estudiantes de sexto grado de la escuela primaria de todo el país en 2023.

El archivo adjunto `Aprender2023_curado.csv` contiene datos descargados el 22/05/2025 de la página web del gobierno nacional `https://www.argentina.gob.ar/aprender-2023`. Originalmente, el dataset tenía información de más de 642.000 estudiantes, incluyendo las notas obtenidas, el nivel socioeconómico, la provincia en la que viven, si la escuela es de gestión estatal o privada, si es de ámbito rural o urbano, y otros. Para este TP, excluimos los estudiantes con información incompleta, resultando en un total de 583.967 estudiantes.

El objetivo de este Trabajo Práctico es escribir programas para modelar, procesar y efectuar algunas consultas sobre estos datos, con la idea de que estos mismos programas puedan ser utilizados para modelar datasets similares con diferentes cantidades de estudiantes y provincias. En concreto, se pide:

1. Implementar una clase `Estudiante` que modele los resultados y otros datos de un estudiante individual, así como las consultas que se pueden hacer sobre los mismos. Debe tener al menos los siguientes atributos y métodos:

   - `Estudiante(...)`: construye un nuevo objeto de la clase `Estudiante`, con los parámetros que se consideren necesarios.
   - `e.provincia`: la provincia de la escuela en la que estudia el estudiante, de tipo `str`.
   - `e.puntaje_matematica`: puntaje en matemática del estudiante, de tipo `float`.
   - `e.puntaje_lengua`: puntaje en lengua del estudiante, de tipo `float`.
   - `e.puntaje_nse`: el puntaje del nivel socioeconómico del estudiante, `float`.
   - `e.ambito`: ámbito de la escuela del estudiante, de tipo `str`. En este dataset existen dos ámbitos: `'rural'` y `'urbano'`.
   - `e.sector`: tipo de gestión de la escuela del estudiante, de tipo `str`. En este dataset existen dos sectores: `'estatal'` y `'privado'`.
   - `str(e)`: devuelve una representación como string del estudiante `e` (esto se logra definiendo el método `__repr__`), con el siguiente formato: `<Mat:`*`FLOAT`*`, Len:`*`FLOAT`*`, NSE:`*`FLOAT`*`, `*`AMBITO`*`, `*`SECTOR`*`, `*`PROVINCIA`*`>`, donde *`FLOAT`* son números con exactamente dos dígitos decimales, *`AMBITO`* es un string (`'Rural'` o `'Urbano'`), *`SECTOR`* es un string (`'Estatal'` o `'Privado'`), y *`PROVINCIA`* es un string (`'MZA'`, `'SFE'`, `'CHU'`, etc.).
     Ejemplos:
     `<Mat:507.25, Len:488.68, NSE:-0.26, Rural, Privado, MIS>`
     `<Mat:466.76, Len:550.32, NSE:0.01, Urbano, Estatal, MZA>`
     `<Mat:492.72, Len:542.52, NSE:0.34, Rural, Estatal, CRR>`
   - `e1 == e2`: devuelve `True` si `e1` y `e2` tienen provincia, puntajes, ámbito y sector iguales; y `False` en caso contrario. En este contexto, cnsiderar que dos valores `float` son *iguales* si la diferencia absoluta entre ellos es menor que 0.001.

2. Implementar una clase `Resumen` que encapsule varias estadísticas de interés, calculadas sobre algún conjunto de estudiantes. Debe tener al menos los siguientes atributos y métodos:

   - `Resumen(...)`: construye un nuevo objeto de la clase `Resumen`, con los parámetros que se consideren necesarios.
   - `r.cantidad`: un valor entero que representa la cantidad de estudiantes considerada en este resumen.
   - `r.promedio_matematica`: un valor de tipo `float` que representa el promedio en matemática.
   - `r.promedio_lengua`: un valor de tipo `float` que representa el promedio en lengua.
   - `r.promedio_nse`: un valor de tipo `float` que representa el nivel socioeconómico promedio.
   - `r.proporcion_ambito_rural`: un valor de tipo `float` (entre 0.0 y 1.0) que representa la proporción de estudiantes que van a una escuela en el ámbito rural.
   - `r.proporcion_sector_estatal`: un valor de tipo `float` (entre 0.0 y 1.0) que representa la proporción de estudiantes que van a una escuela del sector estatal.
   - `str(r)`: devuelve una representación como string del resumen `r`, con el siguiente formato: `<Mat:`*`FLOAT`*`, Len:`*`FLOAT`*`, NSE:`*`FLOAT`*`, Rural:`*`FLOAT`*`, Estado:`*`FLOAT`*`, N:`*`INT`*`>`, donde *`FLOAT`* son números con exactamente dos dígitos decimales, e *`INT`* es un número entero.
     Ejemplos:
     `<Mat:481.97, Len:503.54, NSE:0.37, Rural:0.06, Estado:0.69, N:2663>`
     `<Mat:517.04, Len:536.54, NSE:0.54, Rural:0.00, Estado:0.51, N:30478>`
     `<Mat:462.57, Len:497.03, NSE:0.42, Rural:0.02, Estado:0.78, N:4356>`
   - `r1 == r2`: devuelve `True` si `r1` y `r2` tienen cantidad, promedios y proporciones iguales; y `False` en caso contrario. En este contexto, cnsiderar que dos valores `float` son *iguales* si la diferencia absoluta entre ellos es menor que 0.001.

3. Implementar una clase `Pais` que modele el dataset de resultados de un país (es decir, una colección de datos de muchos estudiantes) y las consultas que se pueden hacer sobre los mismos. Debe tener, al menos, los siguientes atributos y métodos:

   - `Pais(archivo_csv)`: construye un nuevo objeto de la clase `Pais` a partir del archivo CSV pasado como argumento. Por ejemplo, `Pais('mis-datos.csv')` construye un objeto de la clase `Pais` con los datos contenidos en el archivo `'mis-datos.csv'`.
   - `p.tamano()`: devuelve la cantidad de estudiantes en el dataset. Debe tener complejidad temporal $O(1)$ en el peor caso.
   - `p.provincias`: conjunto de provincias presentes en `p` (notar que es un atributo).
   - `p.resumen_provincia(provincia)`: devuelve un objeto de la clase `Resumen`, que encapsula ciertas estadísticas para la provincia dada. Debe tener complejidad temporal $O(N)$ en el peor caso, donde $N$ es la cantidad de estudiantes en `p`.
   - `p.resumenes_pais()`: devuelve un diccionario donde las claves son las provincias presentes en `p` y los valores son sus resúmenes correpondientes. Debe tener complejidad temporal $O(N * P)$ en el peor caso, donde $N$ es la cantidad de estudiantes y $P$ la cantidad de provincias en `p`.
   - `p.estudiantes_en_intervalo(categoria, x, y, provincias)`: dados una categoría (un string que puede tomar valor `'mat'`, `'len'` o `'nse'`), dos números (`x` e `y`) y un conjunto de provincias (todas presentes en `p`), devuelve la cantidad de estudiantes de las provincias indicadas que tienen un puntaje en matemática, lengua o nivel socioeconómico (según corresponda) mayor o igual que `x` y menor estricto que `y`. Debe tener complejidad temporal $O(N * P)$ en el peor caso, donde $N$ es la cantidad de estudiantes y $P$ la cantidad de provincias en `p`.
   - `p.exportar_por_provincias(archivo_csv, provincias)`: genera un nuevo archivo con nombre `archivo_csv`, con una fila por cada una de las provincias indicadas (todas ellas presentes en `p`). El archivo debe tener las siguientes columnas: `provincia` (código de 3 letras), `cantidad` (de estudiantes), `promedio_matematica`, `promedio_lengua`, `promedio_nse`, `proporcion_ambito_rural`, `proporcion_sector_estatal`. Los valores de las columnas corresponden a las estadísticas de interés descriptas en la clase `Resumen`.

## Modo de entrega:

Se deben entregar los siguientes archivos, respetando los nombres con exactitud:

- **`estudiante.py`**, **`pais.py`** y **`resumen.py`**, con las definiciones de las clases `Estudiante`, `Pais` y `Resumen`, respectivamente.
- **`estudiante_test.py`**, **`pais_test.py`**, **`resumen_test.py`**, con los tests de unidad de las clases `Estudiante`, `Pais` y `Resumen`, respectivamente.
- **`informe.pdf`**, un documento en el cual se explique por qué cada método cumple con la complejidad algorítmica requerida. Incluir también cualquier aclaración adicional que se considere necesaria sobre cualquier parte del trabajo. Se espera que este documento sea conciso, de tres o cuatro páginas, y en formato PDF.
- Los archivos CSV necesarios para testear la clase `Pais`.

La carpeta adjunta `templates` contiene archivos de ejemplo para usar como referencia.

## Observaciones:

- El trabajo se debe realizar en grupos de **tres personas**. La entrega consistirá en un trabajo original realizado íntegra y exclusivamente por las personas que integran el grupo. Se espera que las tres personas participen en toda la resolución. Se desaconseja fuertemente dividir y repartirse el trabajo.
- La fecha límite de entrega es el **martes 17 de junio a las 23:55**.
- Los archivos deben subirse a la *Entrega del TP2* en la página de la materia en el campus virtual.
- El código entregado debe ejecutarse correctamente en `Python3`, y sólo pueden usarse elementos de Python vistos en clase.
- Sólo está permitido importar las bibliotecas `unittest` (para los tests de unidad) y `csv` (para la lectura de archivos CSV; sugerimos usar la clase `csv.DictReader`).
- Es obligatorio especificar el tipo de todas las variables y funciones mediante sugerencias de tipos (*type hints*). Las funciones deben especificarse con el formato de *docstrings* visto en clase.
- Los archivos `estudiante.py`, `resumen.py` y `pais.py` no deben tener código principal; solamente definiciones de clases. El objetivo es que dichos archivos sean importados y usados por otros programas, como parte de un proyecto más grande.
- Durante el desarrollo y para el testing de unidad, se recomienda **evitar trabajar sobre el archivo CSV completo**. Por su tamaño, su lectura demora tiempo en cada ejecución y resulta difícil saber si los resultados son correctos. En cambio, conviene crear archivos CSV pequeños, conteniendo pocos registros tomados del archivo original, para tener mayor control de las ejecuciones.
- Puede suponerse que los objetos y sus métodos siempre se usarán de manera correcta. Es decir, si hay errores en la cantidad, tipo o valor de los argumentos, no se espera ningún comportamiento particular (la ejecución podría colgarse o terminar en un error, por ejemplo).
- **Sobre la evaluación del TP:** se tendrá en cuenta no solamente que el código sea correcto, sino también que sea claro, ordenado y modular, que esté comentado adecuadamente, que cumpla los órdenes de complejidad requeridos (con las justificaciones correspondientes), que el testing de unidad tenga un cubrimiento adecuado, que se respete con cuidado todo lo pedido en el enunciado, y que los archivos entregados sigan las indicaciones dadas.