Tecnología Digital 1: Introducción a la Programación
Trabajo Práctico 1

Escuela de Negocios, UTDT

Primer semestre 2021

Un número $n \in \mathbb{N}_0$ (donde $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$; es decir, los enteros mayor o iguales que 0) se considera *peculiar* si y sólo si n es múltiplo de 22 y además la representación en base decimal de n alterna dígitos pares e impares. Por ejemplo, el número 21890 es peculiar porque $21890 = 22 * 995$ y además alterna 2 (par), 1 (impar), 8 (par), 9 (impar), 0 (par). Otros ejemplos de números peculiares son 616 y 1034.[1]

El objetivo de este trabajo es crear un programa en Python que ofrezca las siguientes funcionalidades:

- dados $n, m \in \mathbb{N}_0$, determinar si n y m son ambos pares, o bien ambos impares;
- dado $n \in \mathbb{N}_0$, determinar si los dígitos de n alternan su paridad;
- dado $n \in \mathbb{N}_0$, determinar si n es peculiar (en particular, el 0 es un número peculiar);
- dado $n \in \mathbb{N}_0$, obtener el n-ésimo número peculiar (definimos al 0 como el 0-ésimo número peculiar);
- dados $n, m \in \mathbb{N}_0$, obtener la cantidad de números peculiares entre n y m (inclusive en ambos casos).

Para cada una de estas funcionalidades requeridas, se pide:

a) Escribir la especificación de una función que resuelva el problema.

b) Construir un conjunto de casos de test, que se consideren suficientes para ilustrar y poner a prueba el comportamiento esperado.

c) Pensar un algoritmo y llevarlo a un programa en Python que resuelva el problema.

d) Asegurarse de que la función pasa los casos de test construidos.

e) Justificar por qué los programas escritos hacen lo esperado. En particular, demostrar informalmente que los ciclos terminan y son correctos.

El programa presentado deberá poder ejecutarse desde la línea de comandos y recibirá como argumentos el nombre de la función que se desea ejecutar (por ejemplo, "`es_peculiar`") más los argumentos necesarios para dicha función.

A continuación se ilustran algunas ejecuciones del programa con el resultado esperado:

Comando a ejecutar por línea de comandos	Resultado impreso por pantalla
`python tp1.py misma_paridad 4 4`	`sí`
`python tp1.py misma_paridad 4 5`	`no`
`python tp1.py alterna_paridad 0`	`sí`
`python tp1.py alterna_paridad 1`	`sí`
`python tp1.py alterna_paridad 123450`	`sí`
`python tp1.py alterna_paridad 123455`	`no`
`python tp1.py es_peculiar 1078`	`sí`
`python tp1.py es_peculiar 1079`	`no`
`python tp1.py n_esimo_peculiar 0`	`0`
`python tp1.py n_esimo_peculiar 1`	`418`
`python tp1.py n_esimo_peculiar 15`	`1298`
`python tp1.py cant_peculiares_entre 100 200`	`0`
`python tp1.py cant_peculiares_entre 100 1000`	`6`

[1] Es posible probar que los números peculiares son infinitos. ¿Se les ocurre una demostración? (Esto **no** forma parte de la evaluación.)

Se deben entregar los siguientes cuatro archivos:

- **`peculiar.py`**, con el código de todas las funciones: `misma_paridad`, `alterna_paridad`, `es_peculiar`, `n_esimo_peculiar`, `cant_peculiares_entre`, y cualquier otra función auxiliar que se defina. Es obligatorio especificar el tipo de todas las variables y funciones mediante sugerencias de tipos (*type hints*). Las especificaciones de las funciones se deben incluir con el formato de *docstrings* presentado en clase.
- **`peculiar-test.py`**, con los tests de unidad de todas las funciones definidas.
- **`tp1.py`**, con el código principal para ejecutar el programa como se indica arriba. Este código es el encargado de leer los argumentos de la línea de comandos, invocar a las funciones definidas en `peculiar.py` y mostrar los resultados por pantalla.
- **`informe.pdf`**, un documento con las respuestas al punto e) enunciado arriba (justificaciones), y 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.

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 trabajo original realizado íntegra y exclusivamente por las personas que integran el grupo.
- La fecha límite de entrega es el **martes 6 de abril a las 23:55**. Los TPs entregados fuera de término serán aceptados hasta 48h pasada esta fecha, pero la demora incidirá negativamente en la nota.
- Los archivos deben subirse a la *Entrega del TP1* en la página de la materia en el campus virtual.
- El programa entregado debe correr correctamente en `Python3`.
- Sólo está permitido importar la biblioteca `sys` (para la lectura de los argumentos con `sys.argv` en `tp1.py`) y `unittest` (para correr los tests en `peculiar-test.py`).
- Como ayuda para usar `sys.argv`, tener en cuenta el apunte “Argumentos mediante la consola” publicado en la página de la materia.
- La función `print` solamente puede invocarse en el cuerpo principal del código, en el archivo `tp1.py`. Es decir, no puede usarse en las funciones requeridas ni en las funciones auxiliares.
- Sólo pueden usarse las instrucciones y estructuras de control vistas en clase. En particular, no pueden usarse listas ni iteradores. Tampoco está permitido usar slices de strings.
- Puede suponerse que el usuario siempre invocará al programa de manera correcta. Es decir, si hay errores en la cantidad, tipo o valor de los argumentos de entrada, no se espera comportamiento alguno del programa (podría colgarse o terminar en un error, por ejemplo).