Ejercicios en Python

Desarrollo de Aplicaciones 5to T.I.C.

por Julián Zylber

Última modificación: 30 de abril de 2022

Índice

1. Funciones elementales, variables y tipos de datos

1.1. Cliché

Escribir un programa que imprima por pantalla "`hello, world`". Para no ser igual a todos los cursos de arrancar a programar, propongo en vez que sea "`Oh Hi Mark`"[1]. La idea es que se familiaricen con la idea de escribir y luego correr un programa. No hace falta que guarden este.

1.2. Echo

En Linux, hay una función del shell (consola), llamada `echo`, que da como output en forma de texto lo que le hayamos pasado. Por ejemplo, `echo Oh Hi Doggy`[2] devuelve "`Oh Hi Doggy`". Es muy usada para enviar texto por pantalla o para escribir en archivos/pipes[3]. Vamos a hacer un programa que haga algo similar a `echo`. Pedir por consola input del usuario como "`echo `" y luego imprimir por pantalla lo que me ingresó el usuario

1.3. Esta vuelta Joey aprendió

Existe un meme template de *Friends* que se trata de que Phoebe le explica algo lento a Joey pero después Joey vuelve al error. A continuación un ejemplo:

Armen un programa que imite la dinámica, pero esta vuelta Joey dice lo correcto al final, sin que Phoebe le repita la frase completa. Osea, pidan input del usuario 3 veces (el usuario es Phoebe), y entre cada input Joey (nuestro programa) repite lo que Phoebe le dice. Al tercer ingreso de Phoebe, Joey no solo dice el ultimo fragmento, sino que también dice la frase completa, **sin esperar que Phoebe diga la frase completa**.

Extra: Armar un meme con el template de Phoebe/Joey. Después votamos el mejor, y ponemos al ganador a reemplazar al que está en esta guía de ejercicios. Pueden armarlo acá: https://imgflip.com/memegenerator/231451789/Phoebe-Joey

1.4. Steam Sale Salada

Steam es un plataforma para comprar juegos de computadora. Comprar juegos en Steam es barato, pero no tan barato como parece. Al precio que aparece, se le debe agregar:

- impuesto a las ganancias (35 %)
- impuesto PAIS (8 %)
- IVA (21 %)

Hacer un programa en python que le pueda ingresar el precio de un juego en Steam, y me devuelva por pantalla, **detallado**, cuanto pago por cada impuesto y cuanto termino pagando en total. No solo se debe decir los valores, se debe imprimir a que corresponden **en la misma línea**. **TIP:** la función `str()` convierte cosas a texto.

[1] The Room es la peor y mejor película a la vez
[2] Si, también es The Room, acepto sugerencias para cambiarlo
[3] No tienen idea que es un pipe, pero lo voy poner igual por una cuestión de #completitud

2. Condicionales

2.1. Ser par o ser impar, esa es la pregunta

Hacer un programa que tome un número entero y diga si el número es par o impar. **Tip:** El operador % da el resto de la división entera. Ejemplo: $17 \div 5$ da da cociente 3 y resto 2. Osea, $17/5 = 3$ y $17\,\%5 = 2$, ya que % da el resto.

2.2. Mil horas (como un perro)

Se dice que los perros tienen 7 años por cada año humano. Sin embargo, esto no funciona considerando que los perros llegan a la adultez a los 2 años. Proponemos entonces, un mejor sistema: los primeros 2 años valen 10.5 años perro, y después, cada nuevo año vale por 4 años perro. Ejemplo:

Años humanos	1	2	3	4	n
Años perro	10.5	21	25	28	$21 + (n-2) \cdot 4$

Hacer un programa que tome años humanos y los convierta en años perro.
Desafío extra: ¿Y al revés? Hacer ese programa también.

2.3. Tarjeta de crédito

Las tarjetas de crédito no son simplemente dígitos al azar. Cada dígito tiene una función, y esto permite rápidamente obtener datos de la tarjeta y chequear que el usuario no se haya equivocado al ingresar el número. A continuación una versión simplificada de dicha validación:

- El primer dígito identifica la red de la tarjeta.
  - Las tarjetas que comienzan con 3 son American Express
  - Las tarjetas que comienzan con 4 son Visa
  - Las tarjetas que comienzan con 5 son Mastercard
- Los dígitos 2,3,4,5 y 6 representan el banco emisor.
- Los dígitos 7,8,9,10,11,12,13,14 y 15 son únicos para cada tarjeta de crédito
- El último dígito se usa para validar (Lo vamos a ver más adelante).

Primera Parte

Hacer un programa que tome un **número (no texto)** de tarjeta de crédito de 16 dígitos del usuario, e imprima por pantalla a que red pertenece la tarjeta. Si la tarjeta no pertenece a ninguna red, imprimir un mensaje de error que lo diga. Ejemplos:

- 4956618550028752 es una tarjeta Visa
- 5453680960593325 es una tarjeta Mastercard
- 7499297903955946 no pertenece a ninguna red

Tip: Para obtener los dígitos de un número podemos jugar con tomar cociente y/o resto de dividir por potencias de 10. Ejemplos:

- `4536 // 1000 = 4`
- `4536 % 1000 = 536`
- `4536 % 10 = 6`
- `4536 // 10 = 453`
- `(4536 // 10) % 100 = 53`
- `(4536 % 100) // 10 = 3`

Segunda Parte

Resulta que las American Express son de 15 dígitos, las de Visa de 14 y las de Mastercard de 16. Modificar el programa anterior para que no solo chequee el primer dígito sino la longitud del número ingresado.

Tip #1: Recuerden los `and` y `or`.

Tip #2: Para "medir la longitud" de un número pueden pensar en que significa tener n dígitos. Por ejemplo, los números de 4 dígitos son los mayores iguales a 1000 pero menores a 10000.

Desafío extra

Hacer el programa pero ahora para la versión no simplificada[4]: https://www.ibm.com/docs/en/order-management-sw/9.3.0?topic=cpms-handling-credit-cards. Hacer solamente la parte de *"Prefix and length validation"*, nada más, es decir, no solo la institución, sino que el número ingresado tenga la cantidad correcta de dígitos.

2.4. Astrology for dummies

Una de los elementos básicos de la astrología son los signos, que están determinados por el mes y día de nacimiento. Hacer un programa que tome el día y el mes de nacimiento (Sugerencias: Dos `input` distintos) y le diga al usuario su signo del zodíaco. Acá está la tabla de conversión:

Signo	Rango de Fechas
Capricornio	22 de Diciembre a 19 de Enero
Acuario	20 de Enero a 18 de Febrero
Piscis	19 de Febrero a 20 de Marzo
Aries	21 de Marzo a 19 de Abril
Tauro	20 de Abril al 20 de Mayo
Géminis	21 de Mayo al 20 de Junio
Cáncer	21 de Junio al 22 de Julio
Leo	23 de Julio al 22 de Agosto
Virgo	23 de Agosto al 22 de Septiembre
Libra	23 de Septiembre al 22 de Octubre
Escorpio	23 de Octubre al 21 de Noviembre
Sagitario	22 de Noviembre al 21 de Diciembre

Desafío extra: ¿Que pasa si el usuario pone un mes que no existe? (Ejemplo: *Germinal*[5]) ¿O un día de un mes imposible? (Ejemplo: 50 de septiembre, 30 de febrero). Arreglar el programa para que de error en caso de que se ingrese una fecha imposible.

2.5. La cuadrática

DISCLAIMER: Yo se que no es lo mas entretenido del mundo hacer cosas de matemática, pero es innegable que son útiles. Aparte, son cosas que conocen. Prometo que las temáticas de los ejercicios van a ser variadas

La idea de este ejercicio es hacer un programa que dada una cuadrática (polinomio de grado 2) nos diga sus raíces. Para eso, basta con aplicar la resolvente.

Primera parte

Hacer un programa que pida a, b y c, y calcule las raíces de la cuadrática.

$$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Recordatorio: Para saber si una cuadrática tiene 0,1 o 2 raíces se usa el **determinante** ($b^2 - 4ac$)

determinante	cantidad de raíces
< 0	0 (No tiene raíces reales)
0	1
> 0	2

[4] Bienvenidos a la vida real

[5] Para los que piensan "que ejemplo raro Julián", miren esto

Segunda parte

Bajar y descomprimir el archivo *"Cuadrática.zip"* en la carpeta `Archivos para ejercicios` del drive. Deberían ver 1 archivo, `main.py`, y la carpeta `aux` con otros archivos. Ahora seguir los siguientes pasos:

1. En VSC, abrir la carpeta "Cuadratica". Abrir la terminal y tipear `pip install -r requirements.txt` que va a instalar las librerías requeridas para correr dichos archivos (`ply` y `datclasses`).

2. Si no desean "contaminar" su instalación de Python con todas estas librerías, pueden armar un `virtualenv`. Esto no es necesario, pero para el que le interesa, puede resultar util: https://docs.python-guide.org/dev/virtualenvs/.

Listo! Ahora pueden escribir en el archivo `main.py` la cuadrática con un código similar al que hicieron en la primera parte. Completen debajo del comentario que dice "COMPLETEN USTEDES". Para correr, corran `main.py`.

Fíjense que ahora el input es texto, y ya les viene separado quien es a, b y c. Para ingresar polinomios, se usa '**' para potencias (igual que python).

Ejemplo: $3x^2 - 8x + 2$ se ingresa como `3x**2 - 8x + 2`

Breve explicación: Una librería/módulo/paquete es código hecho por otra persona que podemos usar libremente:

- `ply`**:** Lo uso para "interpretar" la cadena de texto.
- `dataclasses`**:** Lo uso para definir la idea de polinomio de una forma mas sencilla

Y los 3 archivos cumplen la siguiente función:

- `parser.py`**:** *parsea* el texto, es decir, interpreta el texto como un polinomio. Acá uso `ply` para ayudarme
- `polynomials.py`**:** Define la idea de polinomio, cual usamos para luego hacer la cuadrática
- `main.py`**:** Toma el input, y llama a las funciones correspondientes para poder correr su cuadrática.

DISCLAIMER: Los 3 archivos fueron escritos por mi, así que puede esperen[6] *bugs*.

3. Ciclos Acotados

3.1. Cocinando un huevo duro

Cuando alguien se va a vivir solo, se le suele decir que tiene que saber cocinar un huevo duro. Eso solo implica hervirlo cierta cantidad de tiempo. Para eso, necesitamos un *timer*. La idea es crear dicho timer, osea, dada una cantidad de segundos, que haga una cuenta regresiva de esa cantidad de segundos. **TIP:** para que la computadora espere 1 segundo, basta con el comando `time.sleep(1)`. Tienen que poner al principio del código `import time`, que trae el módulo necesario para que el comando descrito anteriormente funcione.

Desafío extra: ¿Que tal si en tiempo está en minutos? Modificar el programa para que tome minutos y no segundos. Puede ser minutos no enteros, por ejemplo, 0,5 minutos son 30 segundos.

3.2. Porto/Bariloche, no diciembre

Para poder pasar el año, me tiene que dar el promedio de notas mayor a 7. Hacer un programa al cual le pueda ingresar notas, y, no solo me calcule el promedio, sino que también determine si a la vuelta del viaje de egresados me quede seguir sufriendo o si soy libre[7]. **TIP:** A priori uno no sabe cuantas notas va a promediar, entonces no se cuantos `input` tendría que incluir. Pidámosle al usuario cuantas notas quiere promediar antes de empezar[8].

[6] Si, dice "puede esperen". Lo iba a corregir, pero después me di cuenta que es un excelente ejemplo de la clase de problemas que puede traer código no testado de un estudiante universitario.

[7] *como el sol cuando amanece, yo soy libreeeeee...*

[8] Más adelante vamos a ver formas menos cavernícolas de determinar esto.

4. Ciclos No Acotados

4.1. This is fine

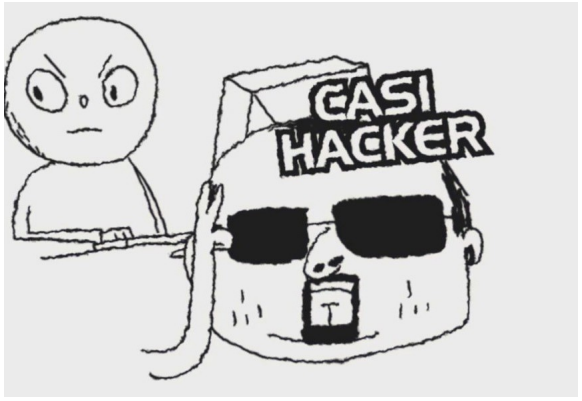
Hacer un programa que se cuelgue. Idealmente debe hacer algo infinitamente, no simplemente colgarse. Corten al programa después de un ratito (`Ctrl + C`) , no queremos incendiar la PC.

4.2. Las contraseña es 1234

Pedirle al usuario que ingrese una contraseña, y e imprimir por pantalla si ingresó la correcta. Ahora, si puso otra cosa, pedirle la contraseña nuevamente, y así sucesivamente hasta que no ingrese la correcta.

Desafío extra: Esto no es muy real. En general hay un límite a que tantos intentos podemos hacer, así un atacante no puede adivinar la contraseña a fuerza bruta. Agregarle a nuestro programa de contraseñas un límite de intentos.

4.3. Casi Hacker/The Secret



Nos dieron un archivo desconocido, el cual tiene un secreto[9]. Sin embargo, este archivo esta protegido por una contraseña especial para cada persona. Esta contraseña es un número de 0 a 999 inclusive, y cuando ingresemos la correcta, el archivo nos va a decir el secreto.

Para esto, nos vamos a convertir en casi hackers, y vamos a probar contraseñas a lo bruto, hasta encontrar la correcta. Para esto contamos con la función `probar_contraseña`, que toma 2 parámetros:

1. El nombre de la persona que quiere acceder
2. La contraseña(numérica) que se desea probar

La función devuelve `True` si la contraseña es correcta, `False` si no. Construir un programa que encuentre la contraseña deseada. **¡IMPORTANTE!** Este ejercicio usa archivos en el campus, bajarlos.

Tip: Si no tienen idea que significa el secreto(probable), googleenlo.

[9] The Secret

5. Listas

5.1. Where´s Wally?

'Donde está Wally?' es el famoso libro de dibujos donde hay que buscar a Wally, un personaje, entre dibujos muy cargados de cosas. Resulta ahora que Wally decidió salir a buscar un trabajo de programador de `python` y se terminó escondiendo en una lista.

Hacer un programa que dada una lista devuelva la posición en donde está. Si tengo la lista `['Hola','Otra cosa','Wally','Perón','ORT']`, el programa debe devolver `Wally está en la posición 2` (queda a criterio de ustedes si usan la indexación de `python` o si le suman uno para usar la más natural).

Desafío extra: Si Wally no está en la lista, tienen que informarlo también (una única vez, no por cada elemento).

5.2. Wanted

Agus armó una app para la fiesta de egresados que encuentra a las parejas más apropiadas. Lula le pasó sus datos a Agus, y Agus nos pasó dos listas, de igual longitud, una con nombres y otra con el porcentaje de compatibilidad. Las listas están en el mismo orden, de tal forma que al primer nombre le corresponde el primer porcentaje, al segundo nombre el segundo porcentaje, etc. Tu misión es hacer un programa que le diga a Lula quién es su *Wanted* para la fiesta de egresados. *Ejemplo:* Si Agus nos da las listas `['Lauti','Cami','Mile','Fran','Agus']` y `[34,56,48,78,70]`, deberíamos devolver `'Fran'`.

Tip: Fijense de recorrer la lista de porcentajes. Si encuentro el índice del máximo elemento, ya tengo gran parte del problem a hecho.

Desafío extra: Resulta que a veces hay gente que tiene la misma compatibilidad. Modificar el código para que considere que puede haber más de un máximo.*Ejemplo:* Si nos dan las listas `['Lauti','Cami','Mile','Fran','Agus']` y `[34,78,48,78,70]`, deberíamos devolver `'Cami','Fran'`.

5.3. Gauss Wars: A New Chore

La idea de esta serie de ejercicios es armar un programa que pueda factorizar polinomios usando Gauss y Ruffini. Ya se que no es lo más entretenido del mundo, pero ustedes sabrán mejor que nadie, es súper útil. Aparte empezamos a programar cosas de otras áreas y las empezamos a automatizar.

En esta primera parte, vamos a armar un programa que hace algo crucial al momento de encontrar los candidatos a raíces derivados del teorema de Gauss: los divisores de un número. Dado un número, devolver la lista de divisores de dicho número. Recuerden que un número d es divisor de n si $n \div d$ tiene resto 0. En Python, % da el resto de la división. Los divisores de un negativo son los mismos que los de su valor absoluto. Pueden tomar valor absoluto en python con `abs`. **Importante:** Gauss usa los divisores negativos también, que son iguales que los positivos pero negativos, agregar estos también.

Desafío extra: Si lo piensan, no es necesario recorrer los n números. Basta con recorrer la mitad[10], y agregar como divisor no solo al encontrado sino al cociente. *Ejemplo:* 3 es divisor de 36, pero como 3 es divisor de 36, 12 también lo es, ya que $3 \cdot 12 = 36$. Modificar el programa para que recorra los primeros $\frac{n}{2}$ números, y no todos.

[10]Incluso menos, podemos bajar a $\sqrt{n}$

6. Funciones

6.1. Jesus de Laferrere

Jesus de Laferrere es una importante figura de la comunidad rollinga. Con su poder, hace milagros que permiten sobrevivir de la amenaza careta.

Primera parte

Uno de los milagros mas importantes de Jesus de Laferrere es su capacidad de multiplicar panchos y birras. Crear una función que dado un string y un número n, devuelva una lista con ese string n veces. *Ejemplo:* si hago `milagro_rollinga('Pancho',3)` devuelva `['Pancho','Pancho','Pancho']`

Usar esta función para armar dos listas de `'Pancho'` y `'Birra'` de 100 elementos cada una.

Segunda Parte

Los rollingas se quejaron de que al pancho le faltaba algo. Entonces Jesús realizó otro milagro y creó una lluvia de papas fritas. Armar una función que a un string le agregue el string `' con papas'`. *Ejemplo:* si hago `lluvia_de_papas('Pancho')` devuelva `'Pancho con papas'`.

Aplicarle `lluvia_de_papas` a cada pancho de la lista construida en la primera parte.

6.2. Gauss Wars: The Divisor Strikes Back

El siguiente paso para factorizar polinomios es encontrar todos los candidatos a raíces racionales por Gauss.

> **Recordatorio:** El teorema de Gauss dice que los candidatos a raíces racionales son los divisores del término independiente (sin x) sobre divisores del coeficiente principal (multiplica al x de mayor grado). *Ejemplo:* Dado el polinomio $2x^2 + 4x - 4$, Gauss nos daría como candidatos a $1, -1, 2, -2, 4, -4, \frac{1}{2}, -\frac{1}{2}$

¡Pero ya hicimos los divisores![11] Traigan el código que hicieron, y pónganlo en una función. Lo vamos a usar acá. Ahora armen una función, que dado un polinomio representado como lista, de los candidatos a raíces por gauss. Por ahora, no pasa nada con los repetidos. Pero Julián, ¿Cómo representamos polinomios con una lista? Ejemplos:

$$x - 4 = [1, -4]$$
$$x^2 - 3x + 5 = [1, -3, 5]$$
$$-3x + 5x^3 + 2x^2 + 1 = [5, 2, -3, 1]$$
$$x^2 - 4 = [1, 0, -4]$$

Donde `polinomio[0]` es el coeficiente principal, y `polinomio[len(polinomio) - 1]` es el término independiente.

Desafío extra: Este procedimiento da bastantes repetidos. ¿Podremos eliminarlos? Armar un algoritmo que elimine repetidos de la lista, o googlear (y usar) `Set`, otro tipo de datos en Python.

6.3. Los mensajes secretos de Julio César

Julio César fue un brillante general y político romano. En su vida llevó a cabo múltiples campañas militares, y en ellas, desarrolló una de las primeras formas de criptografía. Para proteger mensajes de contenidos militares sensibles, los encriptaba de una forma que si el mensaje cayera en manos enemigas, el enemigo no pudiera saber su contenido.

El cifrado funciona de la siguiente manera[12]: por cada letra del mensaje, esta se reemplaza por otra letra que se encuentra un número fijo de posiciones más adelante en el alfabeto. Por ejemplo, con un desplazamiento de 3, la A

[11] Obi-wan murió por esos divisores, respeten.

[12] Si no entienden, lean acá: https://es.wikipedia.org/wiki/Cifrado_C%C3%A9sar

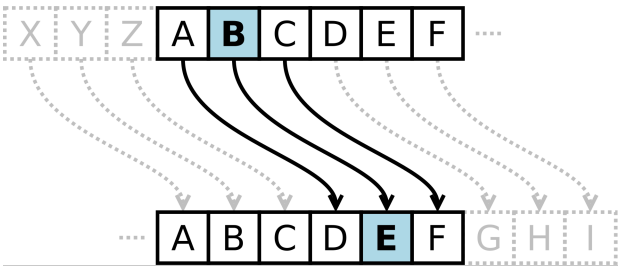


Figura 1: Cifrado de César de 3 espacios

sería sustituida por la D (situada 3 lugares a la derecha de la A), la B sería reemplazada por la E, etc. En la Figura 1 se puede ver el cifrado en acción.

Este ejercicio cuenta con archivos y tests. Bajarlos del Campus, y completar en el archivo `cesar.py`. Para correr los tests deben primero instalar pytest poniendo el siguiente comando por consola: `pip install pytest`. Para correr los tests se escribe en consola: `python -m pytest test_palabra.py`

Primera Parte

Completar la función `encode` en `cesar.py`, que dada una palabra, y un shift, dar su versión codificada. Verificar que los tests relativos a la codificación den bien. Podemos asumir que las palabras usan el alfabeto inglés (sin ñ y ll) y todo minúsculas. Los shifts pueden ser cualquier número **entero**.

Segunda Parte

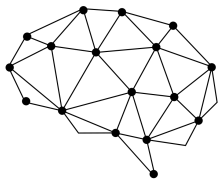
Completar la función `decode` en `cesar.py`, que dada una palabra codificada, y el shift al que fue sometida, dar su versión decodificada. Verificar que los tests relativos a la decodificación den bien. Podemos asumir que las palabras usan el alfabeto inglés (sin ñ y ll) y todo minúsculas. Los shifts pueden ser cualquier número **entero**.

Desafío Extra

Completar las funciones `encode_sentece` y `decode_sentece`, que codifican y decodifican **oraciones**, es decir, pueden contener espacios. Esta parte cuenta con tests propios, que se pueden acceder corriendo: : `python -m pytest test_oraciones.py`

No hace falta que tomen en cuenta otros signos de puntuación (, ; : .)ni mayúsculas. Igual, si están valientes inténtenlo, pero no hice tests (por ahora).

6.4. Consultora Política



Cambridge Analytica

¡Hola! Bienvenido a *Cambridge Analytica*[13]. Resulta que nos están ingresando unas encuestas para la siguiente elección legislativa en Argentina. Tenemos el total de votos por agrupación, pero nos interesa ver cuantas bancas les corresponde a cada partido. Vimos que en tu CV pusiste que sabés programar en `Python`, ¿armarías un script que calcule eso automáticamente?

¿Que no tenés idea como se asignan bancas en argentina? ¡No importa! Acá te dejo información sobre el sistema D'Hondt, que es el que se usa: https://es.wikipedia.org/wiki/Sistema_D%27Hondt

¡ESTE EJERCICIO CUENTA CON TESTS Y ARCHIVOS! Para eso, tienen que bajar el archivo *"votos.zip"* del Campus y descomprimir. **Recuerden correr el `pip install -r requirements.txt` para poder correr los tests.** A continuación, la consigna:

[13] Tenemos tus datos

DISCLAIMER: Los tests no son demasiado exhaustivos y los nombres no son demasiado buenos para ver lo que está pasando. Van a tener que leer los tests y el `data.py` para mas información, además de preguntarme. Sepan disculparme, en cuanto lo arregle este cartel desaparecerá[14].

Primera Parte

Completar la función `dhondt` en `main.py`, donde `votos` es una lista de votos por agrupación, `bancas` es el total de bancas a asignar. Por ejemplo, `dhondt([340000,280000,160000,60000,15000],7)` debería dar `[3,3,1,0,0]`. Correr los tests tipeando `python -m pytest test_dhondt.py` en la consola.

Segunda Parte

Si ven el cuarto test, estos no son los números exactos de la elección legislativa de la provincia de buenos aires. Se le asigna una banca a la agrupacion "+Valores" que en la vida real no se le asigno, y una menos al "Frente De Todos". Esto se debe a que el sistema argentino filtra las agrupaciones que tienen menos del 3 % de los votos respecto al padrón electoral, y luego se aplica D'Hondt.

Completar la función `bancas_por_agrupacion`, que tiene casi los mismos parámetros que dhondt. Se le agrega uno más que es el total del padron. Correr los tests tipeando `python -m pytest test_bancas.py` en la consola.

Desafío Extra

Crear una función que haga lo mismo que bancas por agrupación, pero que además diga el partido que recibió cada banca. Pónganle un nombre apropiado, y prueben que ande mas o menos bien[15]. Si se animan aun mas, creen un archivo de tests similar a los que tengo armados yo, y cópienle un poco el formato.

6.5. Gauss Wars: Return of Ruffini

Antes de arrancar, para este ejercicio tienen que bajar código mío. Al igual que con la cuadrática, van a `Archivos para ejercicios`, bajan y descomprimen al archivo *"Gauss-Ruffini.zip"*. Se corre desde `main.py`, y se ingresan polinomios de la misma forma que en la cuadrática.

¡ESTE EJERCICIO CUENTA CON TESTS! Si, me tomé el trabajo de hacer algunos tests para que puedan chequear ~~que tan mal lo hicieron~~ como les fue. Para correrlos, con el comando `python -m pytest` testean tanto la primer como segunda parte. Para correrlos individualmente pueden especificar el archivo a pytest, `testsRuffini.py` si están testeando la primera parte, `testsFactorización.py` para la segunda. Para más información, pueden ir a https://docs.pytest.org/en/7.1.x/.

Si no se acuerdan nada de factorizar polinomios, usen a mis notas de clase de mi era de docente de matemática como referencia: https://drive.google.com/file/d/1pSXDwc6OT3tZKTiKA6iywNrLETI9GSwo/view?usp=sharing

Primera parte

Tenemos que dividir a un polinomio. Para esto, vamos a completar la función `ruffini`, que toma un polinomio (en la forma de lista que ya usamos en Gauss Wars II) y al polinomio de grado 1, que como nos interesa solo la raíz, directo tomamos la raíz. Devolver el "renglón de abajo" de ruffini, que es cociente y resto a la vez. *Ejemplo: Dada la división de* $x^3 + 3x^2 - 4x - 12$ *por* $x - 4$*, que se ve:*

	1	3	-4	-12
4		4	28	96
	1	7	24	84

y devolveríamos la lista $[1, 7, 24, 84]$. **¡No se olviden de correr los tests!**

Segunda parte

Ahora si, podemos factorizar. Traigan las funciones gauss (y por lo tanto también tienen que traer divisores) que ya hicieron, y junto a la función de ruffini, completen la función `factorizar`, que devuelva las raíces racionales de un polinomio. No importa el orden ni que haya repetidas (los tests lo tienen en cuenta). Repito, si no se acuerdan nada, relean mis notas de clase o pregúntenme. **¡No se olviden de correr los tests!**

Usando el teorema del resto puede salir más fácil que usando Ruffini, pero no nos da la multiplicidad de dichas raíces, algo que es súper útil al momento de hacer la factorización y no simplemente obtener raíces.

Desafío extra: Si lo piensan, cuando llegan a grado 2 pueden usar la cuadrática, no hace falta seguir usando ruffini. ¡Y ya hicimos la cuadrática! Traerla, y modificar la función `factorizar` para que la incluya. **¡OJO!** Corran los tests y fíjense que no hayan roto nada, especialmente los de grado 1.

[14] Este cartel lo escribí en 2022, ¿Cuántos años habrán pasado?
[15] Mas o Menos Bien