Ejercicios complementarios del taller de Python

Talleres FIFA (Federación Interestudiantil de Físicos de Argentina)

Observación: En caso de que el IDE de sus computadoras no funcione, se puede probar programar online en la web: `https://www.python.org/shell/`

Observación 2: Estos ejercicios en general son menos guiados que los anteriores, y por ende más complejos. No teman buscar en internet.

Ejercicios propuestos

1. `es_primo(`n`)`: devuelve True si el input es un número primo. False en caso contrario.
2. `divisores(`n`)`: devuelve una lista con todos los divisores enteros del número n. Para generar la lista pueden buscar el método `append`.
3. `factorial(`n`)`: devuelve el factorial del input n. Recordar que el factorial se define: $n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 1$
4. `absoluto(`x`)`: devuelve el valor absoluto del input x. Si $x < 0$ debería devolver $-x$ y si $x \geq 0$ debería devolver x.
5. `norma(x, y, z)`: devuelve la distancia euclídea entre el punto elegido y el origen de coordenadas
   - modifique la función para que reciba un único parámetro v, que sea una tupla o lista de largo 3.
   - extienda esta función para que reciba dos puntos y calcule la norma del vector que los une.
   - puede extender la función para que calcule otras normas no euclídeas, por ejemplo la norma Manhattan o la norma infinito (si no las conocen, Wikipedia les cuenta.).
6. `perp(v1, v2)`: devuelve True si los vectores son perpendiculares y False si no lo son (recordar que es fácil ver si dos vectores son perpendiculares viendo su producto interno: $\vec{v} \cdot \vec{w} = v_1 w_1 + v_2 w_2 + v_3 w_3$.)
   - Generalice a dos vectores de largo N, con un N arbitrario.
7. `prod_vectorial(v, w)` devuelve el producto vectorial $\vec{v} \times \vec{w} = (v_2 w_3 - v_3 w_2; v_3 w_1 - v_1 w_3; v_1 w_2 - v_2 w_1)$. Para verificar la función pueden verificar que $(1;0;0) \times (0;1;0) = (0;0;1)$ (esto es la regla de la mano derecha: $\hat{i} \times \hat{j} = \hat{k}$, para los que la conozcan).
8. Considere la serie de Taylor de la función exponencial $f(x) = e^x = \sum_{n=0}^{\infty} (x^n)/n!$. Como esta suma es infinita se imaginarán que tarda infinito tiempo en converger al valor con error nulo. No obstante, cada término que se calcula acerca la estimación al valor real. El objetivo de este ejercicio es ver la convergencia de esta serie, para realizarlo les conviene tener algunas de las funciones de los ejercicios anteriores a mano.
   Escriba una función `exp_taylor(`x`, `N`)` que reciba como parámetros el valor de x donde evaluar la

exponencial y N siendo el orden de precisión deseado (es decir que la sumatoria en vez de ser hasta ∞ será hasta N) y devuelva el resultado de la sumatoria parcial obtenida.
Luego compare con el valor real"de la función (que pueden obtener, por ahora, usando una calculadora o Google o Mathematica. La próxima clase veremos de donde podemos obtener el valor desde python). Por ejemplo: Se quiere ver la convergencia para $x = 4$, entonces buscamos el valor posta $e^4 \approx 54{,}5981500331$, y para ver el error hacemos: error = $|\,54{,}5981500331 -$ `exp_taylor(4, N)` $|$. La pregunta que se quiere saber es, a qué orden necesito hacer la cuenta para encontrar una aproximación que difiera del valor real en menos de 0,1. Y 0,01? 0,001? etc....
Esto va a estar bueno cuando aprendamos a hacer gráficos, vamos a poder ver el error en función del orden de aproximación.

9. `verificar_nombre(str)`: Esta función va a recibir un string, y va a verificar si es compatible como nombre de archivo devolviendo True o False. Vamos a aceptar nombres solamente si *no* contienen los caracteres: [ ] ( ) { } , / ; . *
Ejemplo: *secretos_fifa* es un nombre válido para un archivo, pero *expedientes-secretos;fifa* no, porque tiene un ';'.

10. Distancia de Hamming: Se quiere realizar una función que compute la distancia de Hamming entre dos strings. Esta se define como la cantidad de caracteres diferentes entre los dos strings de igual largo.
Ejemplo: La distancia de hamming entre `lamento` y `mamerto` es 2, ya que los únicos caracteres diferentes son el primero y el quinto.

11. ¡Adivina un número del uno al diez! Defina un número del 1 al 10, luego pídale al usuario que ingrese un número. Si adivina el número que muestre un mensaje de felicitaciones y si no que siga intentando.
Para esto puede usar la función `input`. Deberá buscar en internet o la documentación de esta función, en particular para ver qué tipo de dato devuelve esta función (si es un `int, str, ...`). Además, buscar cómo elegir un número aleatorio (o `random`) usando python.