Tarea_4

May 24, 2022

1 Tarea #4 (Corregido)

1.0.1 Universidad de Costa Rica

Escuela de Ciencias de la Computación e Informática

CI0202 - Principios de Informática. Grupo G02,G13

Prof. Jose Pablo Ramírez Méndez

I-2022 Enunciado adaptado del profesor Leonardo Villalobos

Objetivo: Familiarizar al estudiante con los conceptos vistos en clase de **Funciones (Subrutinas)**

Formato: Individual

Entregar: Debe entregar su código fuente ya sea en 1 archivo por cada ejercicio con extensión (.py) o 1 cuaderno de Jupyter para todos lo ejercicios de la tarea (Puede utilizar este como base para su entregable)

- En esta asignación, se debe resolver los ejercicios mencionados posteriormente. Para cada enunciado distinto, se deberá entregar un archivo de código fuente en Python 3 que lleve a cabo la tarea especificada (con extensión .py). Puede también y se recomienda entregar el laboratorio en un cuaderno de Jupyter.
- No olvide incluir su **nombre completo y carnet** en su entrega, ya sea como comentarios de códigos o bien, como celdas de textos si decide utilizar cuadernos de Jupyter.
- La tarea es de elaboración y entrega individual. No se aceptarán entregas después de la fecha y hora límite. Tareas elaboradas en grupos serán invalidadas; y serán considerados como **plagio** aquellas que sean idénticas o casi idénticas.
- No olvide aplicar cada una de las etapas de resolución de problemas vista en clase. Se recomienda que en la fase de Diseño e implementación, escriba su algoritmo en psudocódigo como comentarios de línea en su código.

1.0.2 Evaluación

Para cada ejercicio, se evaluará lo siguiente: - **Resultado esperado:** 25% - Lo que el programa realiza coincide con el enunciado. Para una entrada tal, el programa produce el resultado esperado. El programa solicita los datos adecuados y muestra los datos adecuados. - **Resolución del problema:** 65% - Internamente, el programa resuelve o intenta resolver el problema estatado en

el enunciado. Esto significa que, aunque su programa tenga errores de cálculo o formato, si se entiende bien lo que están tratando de hacer, los puntos de esta sección serán otorgados. - **Buen formato:** 10% - Utilice nombres de variables significativos (por ejemplo "coordX" lugar de "x" o "nombre" en lugar de "n"). Haga manejo de entrada y salida con texto descriptivo. Muestre la cantidad de decimales correcta.

Cada ejercicio tiene un puntaje asignado. La nota final es la sumatoria de multiplicar el puntaje de cada ejercicio por la nota obtenida en ese ejercicio, por un máximo de 100 puntos.

Puntos extra

Para cada problema planteado en la tarea adjunte el resultado de la etapa de *Diseño* del flujo del proceso resolución de problemas visto en clase como algoritmo en pseudocódigo. Puede agregarlo como comentarios de código al principio de su implementación o bien, como un documento a parte para todos los ejercicios de la tarea. Puede obtener un máximo de 10 puntos. Por ejemplo, si obtiene un 95 en la nota de la tarea, pero realizó los diseños correspondientes y corresponden **directamente** a la solución implementada, obtiene 10 puntos y su nota sube a 105. Cada diseño tiene valor proporcional a los puntos del ejercicio respecto al máximo de 10 puntos mencionado anteriormente.

2 Ejercicio 1. Valor 100 pts

3 Taylor: Generalización

Ya hemos utilizado las implementaciones de seno y coseno de la biblioteca math, y hemos visto que se pueden aproximar mediante algunas de las series de Taylor. La fórmula que ya hemos visto antes es la de seno para x radianes:

$$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \tag{1}$$

No obstante, habíamos visto que tenían mucho error de cálculo. Esto es porque estas son una forma acotada de la series. Realmente, la serie es infinita. Si se calculara con infinitos factores, se obtendría el verdadero valor de estas funciones. Las verdaderas series (acotadas a M factores) son las siguientes:

$$sin(x) = \sum_{n=0}^{M} \frac{(-1)^n}{(2n+1)!} x^{2n+1} \tag{2}$$

sea x un ángulo en radianes. Cuando M es lo suficientemente grande, esta aproximación se vuelve más precisa. Idealmente tendríamos $M = \infty$, pero como bien saben, las computadoras no disfrutan los ciclos infinitos.

Nuevamente, esta aproximación solo es válida para ángulos en grados en el intervalo $x = [-180, 180]$ (o $x = [-\pi, \pi]$ en radianes).

Así mismo, con sumatorias de Taylor es posible expresar la función exponencial e^x de tal forma que:

$$e^x = \sum_{n=0}^{K} \frac{x^n}{n!} \tag{3}$$

donde x es cualquier número real y K la cantidad de factores en la sumatoria

3.0.1 Instrucciones

1. Cree un programa que pida un ángulo (**en grados**) y una cantidad de factores M. Con base a esa información, su programa deberá aproximar el seno utilizando las fórmulas de arriba (no olvide que estas están en **radianes**). La salida de su programa deberá ser las aproximación del seno. Para ello defina e invoque una función **obtener_serie_taylor_seno** cuyos parámetros serán un ángulo en radianes y la cantidad M de factores. **Retorne** la aproximación de la serie al final de su función.

- **Nota:** La operación del signo de exclamación (!) en las fórmulas se denomina el factorial. El factorial de n (denominado n!) se calcula multiplicando el numero por todos sus antecesores. Por ejemplo, 4! = 4 x 3 x 2 x 1 = 24. Desde la biblioteca math puede importar factorial, que es una función que realiza esta operación. Note que se usa así: `factorial(4)`.
- **Nota 2:** Factorial es una operación computacionalmente pesada. A veces siendo incalculable para valores más allá de 40. Por esto, no utilice más de 20 factores a la hora de correr el programa.

2. Cree un programa que pida el valor real y una cantidad K de factores. Con base a esa información su programa deberá aproximar la función exponencial utilizando las formulas de arriba. La salida de su programa deberan ser la aproximación de la función exponencial. Para ello defina e invoque una función ***obtener_serie_taylor_exp*** *parámetros serán un valor real y la cantidad K de factores.* **Retorne** la aproximación de la serie al final de su función.

- **Nota**: Recuerde que la biblioteca math ofrece muchas funciones útiles para el desarrollo. Utilice la función **exp** para calcular el valor de la función exponencial de un número real.

3. Modifique su programa para que además muestre el error de la aproximación, para ambos seno y función exponencial. Calcule los valores «verdaderos» del seno y función utilizando la biblioteca math (recuerde que igualmente funcionan para ángulos en radianes). El error se calcula como valor_real – valor_aproximado. Un ejemplo final del programa:

```
Ingrese el ángulo en grados: 90
Ingrese el valor real: 8.3
Ingrese la cantidad de factores M a utilizar: 15
Ingrese la cantidad de factores K a utilizar: 19

La aproximación del seno es: 1.000000
El error en esta aproximación es de: -0.000000

La aproximación de la función exponencial es: 4022.263607
El error en esta aproximación es de: 1.608786
```

Nota: Las funciones `obtener_serie_taylor_seno` y `obtener_serie_taylor_exp` no deben realizar ninguna salida de datos en consola, solo calcular las aproximaciones y retornar los valores

correspondientes. Sino estaría asignando más tareas a las funciones lo que disminuye la modularidad y que sean reutilizables.

[1]:
```
from math import factorial, sin, exp

print(factorial(5))
print(exp(8.3))
```

```
120
4023.872393822313
```

[ ]:
```

```