Tarea 3

May 3, 2022

1 Tarea #3

1.0.1 Universidad de Costa Rica

Escuela de Ciencias de la Computación e Informática

CI0202 - Principios de Informática. Grupos G02,G13

Prof. Jose Pablo Ramírez Méndez

I-2022 Objetivo: Repasar, mediante ejercicios prácticos, los conceptos estudiados en clase de **Control de flujo**

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.

1.0.3 Enunciado

Para cada uno de los siguientes ejercicios, elabore un programa que lo resuelva.

1.1 Ejercicio 1. Valor 60 pts

1.1.1 Función cuadrática

El criterio de una función cuadrática esta dado por:

$$ax^2 + bx + c = 0 \tag{1}$$

Donde a, b y c son coeficientes reales y $a \neq 0$.

Escriba un programa que reciba los 3 coeficientes de una ecuación cuadrática y calcule sus raices, o valores que la resuelven (sea igual a 0). Las raices se pueden calcular con las fórmulas:

$$r_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \tag{2}$$

$$r_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \tag{3}$$

Como ya se pueden imaginar, es posible que las raíces de la ecuación no sean reales. Recordemos que esto ocurre cuando el discriminante dado por: $\Delta = b^2 - 4ac$ es negativo. En este caso, la raíz en las fórmulas no tendrían solución. Si llega a ocurrir, su programa debe mostrar un mensaje que señale que la función no tiene soluciones reales. Además, cuando estas raíces sean iguales, **no repita** las soluciones y solo muestre una.

Igualmente, es posible que se introduzca un valor de $a = 0$. Este caso no corresponde a una ecuación cuadrática, pero igualmente mostraremos la solución de la ecuación tal y como lo haríamos con una función lineal de la siguiente forma:

$$r = \frac{-c}{b} \tag{4}$$

En el caso que $a = 0$ y $b = 0$, no se tiene una ecuación puesto que la función sería una constante. En este caso, simplemente muestre un mensaje de error y termine con el programa. Note que es válido que $b = 0$ siempre y cuando $a \neq 0$.

Ejemplos: (Las entradas de datos van indicadas con una flecha (>))

```
a =
> 1
b =
> 2
c =
> -35
Las raices de la ecuación f(x) = 1.00x^2 +2.00x -35.00 = 0 son x_1 = 5.00 y x_2 = 7.00
```

```
a =
> 1
b =
> 1
c =
> 35
La ecuación f(x) = 1.00x^2 +1.00x +1.00 = 0 no tiene solución real
```

```
a =
> 0
b =
> 3
c =
> 9
La raiz de la ecuación f(x) = 3.00x +9.00 = 0 es x = -3.00
```

```
a =
> 0
b =
> 0
c =
> 1
Los valores ingresados no corresponden a una ecuación válida
```

1.2 Ejercicio 2. Valor 40 pts

1.2.1 Factores de un número

Cree un programa que muestre los factores de un número. Los factores son los números por los cuales se es enteramente divisible. Por ejemplo, los factores de 12 son 1, 2, 3, 4, 6 y 12.

¿Cómo hacemos esto en Python? Anteriormente vimos que un número es par si es divisible entre 2, en otras palabras, si al dividirlo entre 2, el residuo es cero. Para ello, utilizamos el operador módulo (%) de python que permite obtener el residuo de la división. Veamos los siguientes ejemplos:

[1]:
```
# 7 no es factor de 12
print(12%7)
```

```
5
```

[2]:
```
# 3 es factor de 12
print(12%3)
```

```
0
```

Otra cuestión es que los factores de un número **siempre** son menores o iguales a él. Además, estamos trabajando con números enteros positivos. Dicho de otra manera, deberíamos revisar todos los números entre 1 y nuestro número. Si el residuo de dividir nuestro número entre alguno de estos, entonces ese es un factor. Su programa debe mostrar número total de factores y listarlos como se muestra a continuación.

Ejemplo de entrada de datos:

```
Introduzca un número: 12
```

Ejemplo de salida de datos:

```
El  número 12 tiene 6 factores que se listan a continuación:
Factor 1: 1
Factor 2: 2
Factor 3: 3
Factor 4: 4
Factor 5: 6
Factor 6: 12
```

Analice detenidamente cómo resolvería usted este problema ya que conlleva un proceso de **repetición**

Nota final: No olvide hacer manejo de excepciones para la entrada. Si se ingresa algo que no es un número al programa, debería mostrarse un mensaje de advertencia y terminar.

A partir de esta entrega, puede utilizar tanto para laboratorios como tareas, las funciones para lectura segura de un valor entero o un valor flotante. No profundizaremos aún en la definiciones de funciones, de momento aprovecharemos el código y el llamado. Estas funciones trabajan igual que la función **input** que ya conocemos. Observe los siguientes ejemplos:

[ ]:
```
# Copie las funciones al inicio de su código para hacer la lectura segura de un␣
↪valor flotante o entero

# Inicio función
def pedir_numero_real(mensaje="Ingrese un número real: ", mensaje_error="Error,␣
↪debe ser un número real."):
    while True:
```

```
        try:
            return float(input(mensaje))
        except:
            print(mensaje_error)
# Fin función

#Inicio función
def pedir_numero_entero(mensaje="Ingrese un número entero: ",␣
↪mensaje_error="Error, debe ser un número entero."):
    while True:
        try:
            return int(input(mensaje))
        except:
            print(mensaje_error)
# Fin función


#Luego de copiarlas al inicio de su código, las puede probar de la siguiente␣
↪manera

unEntero   = pedir_numero_entero(mensaje= "Ingrese un número entero cualquiera:␣
↪", mensaje_error= "Siga instrucciones, debe ser entero")
unFlotante = pedir_numero_real(mensaje = "Ingrese un número flotante que quiera:
↪ ", mensaje_error= "Porfa, que sea flotante")
print( "\nSin personalizar mensaje\n" )
otroEntero   = pedir_numero_entero()
otroFlotante = pedir_numero_real()
```

[ ]:
```
# Material adaptado del profesor Leonardo Villalobos
```