EJERCICIOS PYTHON NIVEL 1.

EJERCICIO 1

Diseña un programa para que, a partir de un número x ingresado, calcule el valor de la función *y*. La expresión de la función y es:

$$\cfrac{1}{x+\cfrac{1}{x+\cfrac{1}{x+\cfrac{1}{x}}}}$$

Datos de Prueba para probar el código:

Entrada de muestra: 1 **Salida esperada:** y = 0.6000000000000001
Entrada de muestra: 10 **Salida esperada:** y = 0.09901951266867294
Entrada de muestra: 100 **Salida esperada:** y = 0.009999000199950014
Entrada de muestra: -5 **Salida Esperada:** y = -0.19258202567760344

EJERCICIO 2

Sabiendo que 1 milla equivale aproximadamente a 1,61kilómetros, diseña un programa que convierta de:

- Millas a kilómetros
- Kilómetros a millas.

Debe aceptar dos valores de punto flotante: las millas y los kilómetros a convertir.

Devolverá la equivalencia en Km para las millas, y la equivalencia en millas para los Km.

Utiliza la función round() para mostrar los valores obtenidos con dos dígitos decimales. Se emplea así: *round*(variable, num_posiciones_decimales)

Datos de Prueba:

Entrada: 7,38 millas y 12,25 Km **Salida esperada**:

```
7.38 millas son 11.88 kilómetros
12.25 kilómetros son 7.61 millas
```

EJERCICIO 3

El impuesto en País se evalúa utilizando la siguiente regla:

- si el ingreso del ciudadano no es superior a 85.528 €, el impuesto es igual al 18% del ingreso menos 556,20 € (la llamada *exención fiscal*).
- si el ingreso es superior a esta cantidad, el impuesto es igual a 14.839,20 €, más el 32% del excedente sobre 85.528 €.

Tu tarea es escribir una **calculadora de impuestos**.

- Debe aceptar un valor de punto flotante: el ingreso.
- A continuación, debe imprimir el impuesto calculado, redondeado a € totales. Utiliza la función round() para redondear el valor.

Si el impuesto calculado es menor que cero, solo significa que no hay impuesto .

Datos de Prueba para probar el código

Entrada de muestra: 10000	**Salida esperada:** El impuesto es: 1244.0 €
Entrada de muestra: 100000	**Salida esperada:** El impuesto es: 19470.0 €
Entrada de muestra: 1000	**Salida esperada:** El impuesto es: 0.0 €
Entrada de muestra: -100	**Salida esperada:** El impuesto es: 0.0 €

EJERCICIO 4

Como seguramente sabrás, debido a algunas razones astronómicas, el año puede ser *bisiesto* o *común*.

Desde la introducción del calendario Gregoriano (en 1582), se utiliza la siguiente regla para determinar el tipo de año:

- si el número del año no es divisible entre cuatro, es un *año común*.
- de lo contrario, si el número del año no es divisible entre 100, es un *año bisiesto*.
- de lo contrario, si el número del año no es divisible entre 400, es un *año común*.
- de lo contrario, es un *año bisiesto*.

Realiza un programa que lee un número de año y según las reglas anteriores muestre uno de los dos mensajes posibles, que son *Año Bisiesto* o *Año Común*, según el valor ingresado.

Sería bueno verificar si el año ingresado cae en la era Gregoriana y emitir una advertencia de lo contrario: *Fuera del período del calendario Gregoriano*.

Datos de Prueba para probar el código**:**

Entrada de muestra: 2000	**Salida esperada:** Año bisiesto
Entrada de muestra: 2015	**Salida esperada:** Año común
Entrada de muestra: 1999	**Salida esperada:** Año común
Entrada de muestra: 1996	**Salida esperada:** Año bisiesto
Entrada de muestra: 1580	**Salida esperada:** Fuera del período del calendario gregoriano

EJERCICIO 5

Escribir un programa que realice un juego que consiste en adivinar un número (puedes inicializar una variable con ese valor). Para ello se pedirá al usuario que ingrese un número entero; comprobará si el número ingresado por el usuario es el mismo que el número elegido. Si los números son diferentes, el usuario debería ver el mensaje "¡Frío, frío!" y se le solicitará que ingrese un número nuevamente. Si el número ingresado por el usuario coincide con el número elegido, el número debe imprimirse en la pantalla, junto con las palabras: "¡Bien hecho!"

EJERCICIO 6

Escribir un programa para calcular las 15 primeras potencias de 2

EJERCICIO 7

La instrucción break se implementa para salir/terminar un bucle.

Diseña un programa que use un bucle while y le pida continuamente al usuario que ingrese una palabra a menos que ingrese "chupacabra" como la palabra de output secreta, en cuyo caso el mensaje "Has dejado el bucle con éxito." debe imprimirse en la pantalla y el bucle debe terminar.

No imprimas ninguna de las palabras ingresadas por el usuario. Utiliza el concepto de ejecución condicional y la sentencia break.

EJERCICIO 8

La sentencia continue se usa para omitir el bloque actual y avanzar a la siguiente iteración, sin ejecutar las sentencias dentro del bucle. Se puede usar tanto con bucles while y for.

Escribe un programa que use:

un bucle for;

el concepto de ejecución condicional (if-elif-else).

la sentencia continue.

Tu programa debe ser un Devorador de vocales (bastante feo) con los siguientes pasos:

- pedir al usuario que ingrese una palabra.
- Utiliza user_word = user_word.upper() para convertir la palabra ingresada por el usuario a mayúsculas;
- Utiliza la ejecución condicional y la instrucción continue para "devorar" las vocales de la palabra ingresada.
- Imprime las letras restantes en la pantalla, en una misma línea

Prueba tu programa con los datos de prueba:

Entrada de muestra: Gregory	**Salida esperada**: GRGRY
Entrada de muestra: abstemious	**Salida esperada**: BSTMS
Entrada de muestra: IOUEA	**Salida**:

EJERCICIO 9

Mejora el programa anterior. Escribe un programa que use:

- un bucle for.
- el concepto de ejecución condicional (*if-elif-else*).
- la instrucción continue.

Tu programa debe:

- pedir al usuario que ingrese una palabra.
- utilizar user_word = user_word.upper() para convertir la palabra ingresada por el usuario a mayúsculas
- emplea la ejecución condicional y la instrucción continue para "devorar" las siguientes vocales *A* , *E* , *I* , *O* , *U* de la palabra ingresada.
- asigna las letras no consumidas a la variable word_without_vowels e imprime la variable en la pantalla.

Prueba tu programa con los **Datos de Prueba:**

Entrada de muestra: Gregory **Salida esperada:** GRGRY
Entrada de muestra: abstemious **Salida esperada:** BSTMS
Entrada de muestra: IOUEA **Salida esperada:**

Utiliza este código de punto de partida:

```
palabra_sin_vocales = ""

# Indicar al usuario que ingrese una palabra
# y asignarla a la variable palabra_usuario.


for letter in palabra_usuario:
    # Completa el cuerpo del bucle.

# Imprimir la palabra asignada a palabra_sin_vocales.
```

EJERCICIO 10

Diseña un programa para que, a partir de un número N ingresado, calcule la suma de la serie:

$$\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{N}$$

Para obtener solo dos decimales en la salida, por ahora utilizaremos la función:

round(variable, posiciones_decimales)

Datos de Prueba:

Entrada de muestra: 1 **Salida esperada:** 1.0
Entrada de muestra: 3 **Salida esperada:** 1.83
Entrada de muestra: 10 **Salida esperada:** 2.93
Entrada de muestra: 1000 **Salida Esperada:** 7.49

EJERCICIO 11

Dos niños juegan con bloques de madera. Están construyendo una pirámide o en realidad una pared en forma de pirámide. La pirámide se apila de acuerdo con un principio simple: cada capa contiene un bloque más que su capa inferior.

Bloques: 6

Tu tarea es escribir un programa que lea la cantidad de bloques que tienen los niños, y generar la altura de la pirámide que se puede construir utilizando esos bloques.

Nota: La altura se mide por el número de **capas completas** - si los niños no tienen la cantidad suficiente de bloques y no pueden completar la siguiente capa, dan por terminada la pared.

Datos de Prueba

Entrada de muestra: 6	Salida esperada: La altura de la pirámide es: 3
Entrada de muestra: 20	Salida esperada: La altura de la pirámide: 5
Entrada de muestra: 1000	Salida esperada: La altura de la pirámide: 44
Entrada de muestra: 2	Salida esperada: La altura de la pirámide: 1

EJERCICIO 12

Modificación del ejercicio 5 de la serie de nivel 1: programa para adivinar un número que la máquina ha pensado (número entre 0 y 100)

Se pedirá al usuario que ingrese números enteros hasta encontrar el número correcto:

- si el número ingresado por el usuario es menor que el número a adivinar, aparecerá el mensaje "Más alto, sube sin miedo"
- si el número ingresado por el usuario es mayor que el número a adivinar, aparecerá el mensaje "No tan alto, es menor"
- si el número ingresado por el usuario coincide con el número elegido, el número debe imprimirse en la pantalla, junto con las palabras: "¡Buen trabajo! Lo has conseguido en x intentos" (sustituyendo x por el número de intentos jugados)

Para generar el número a adivinar utiliza la función — incluída en la librería random— **randint(x,y)** que genera un número entero entre los valores x e y.

Añade al principio de tu código una de estas instrucciones:

```
from random import randint
from random import *
```

EJERCICIO 13

En 1937, un matemático alemán llamado Lothar Collatz formuló una hipótesis intrigante (aún no se ha comprobado) que se puede describir de la siguiente manera:

1. toma cualquier número entero que no sea negativo y que no sea cero y asígnale el nombre c0
2. si es par, evalúa un nuevo c0 como c0 ÷ 2
3. de lo contrario, si es impar, evalúa un nuevo c0 como 3 * c0 + 1
4. si c0 ≠ 1, salta al punto 2.

La hipótesis dice que, independientemente del valor inicial de c0, el valor siempre tiende a 1.

Por supuesto, es una tarea extremadamente compleja usar una computadora para probar la hipótesis de cualquier número natural (incluso puede requerir inteligencia artificial), pero puedes usar Python para verificar algunos números individuales. Tal vez incluso encuentres el que refutaría la hipótesis.

Escribe un programa que lea un número natural y ejecute los pasos anteriores siempre que c0 sea diferente de 1. También queremos que cuente los pasos necesarios para lograr el objetivo. Tu código también debe mostrar todos los valores intermedios de c0.

Sugerencia: la parte más importante del problema es como transformar la idea de Collatz en un bucle while, esta es la clave del éxito.

Datos de Prueba:

Entrada de muestra: 15

Salida esperada: *46 - 23 - 70 - 35 - 106 - 53 - 160 - 80 - 40 - 20 - 10 - 5 - 16 - 8 - 4 - 2 - 1 - Pasos: 17*

Entrada de muestra: 16

Salida esperada: *8 - 4 - 2 - 1 - Pasos: 4*

Entrada de muestra: 1023

Salida esperada: *3070 - 1535 - 4606 - 2303 - 6910 - 3455 - 10366 - 5183 - 15550 - 7775 - 23326 - 11663 - 34990 - 17495 - 52486 - 26243 - 78730 - 39365 - 118096 - 59048 - 29524 - 14762 - 7381 - 22144 - 11072 - 5536 - 2768 - 1384 - 692 - 346 - 173 - 520 - 260 - 130 - 65 - 196 - 98 - 49 - 148 - 74 - 37 - 112 - 56 - 28 - 14 - 7 - 22 - 11 - 34 - 17 - 52 - 26 - 13 - 40 - 20 - 10 - 5 - 16 - 8 - 4 - 2 - 1 - Pasos: 62*