Ejercicios para trabajar con funciones no recursivas

Ejercicio 01:

Diseñe una función de nombre "**la_mas_larga()**" que recibe un número indeterminado de parámetros que representan palabras para buscar entre ellas la más larga. La función debe retornar dos valores, el primero la longitud de la palabra más larga y el segundo una lista donde esta/están la o las palabras más largas de todas las recibidas, tenga en cuenta que puede ser más de una. La lista de palabras que se retorna debe estar ordenada alfabéticamente.

Ejercicio 02:

Diseñe un programa en Python que funcione como un conversor de temperaturas. El programa deberá mostrar el siguiente menú:

1. ºC a ºF
2. ºC a ºK
3. ºK a ºC
4. ºK a ºF
5. ºF a ºC
6. ºF a ºK
9. Salir

Deberá diseñar funciones para cada una de las opciones de la 1 a la 6 del menú. Cada una de esas funciones recibe como parámetro la temperatura que se quiere convertir y devolverá la temperatura convertida en la escala asociada a la opción del menú.

Además, el programa deberá tener dos funciones adicionales:

La primera: leer_temperatura(escala)

Esta función recibe un parámetro indicando la escala en la que deberá leerse la temperatura (asumir que 1 es ºC, 2 es ºF y 3 es ºK) y acorde a ese parámetro deberá adecuar el mensaje que se pone dentro del input().

Si el parámetro es 1, el mensaje sería: "Introduzca el valor de temperatura en ºC:"
Si el parámetro es 2, el mensaje sería: "Introduzca el valor de temperatura en ºF:"
Si el parámetro es 3, el mensaje sería: "Introduzca el valor de temperatura en ºK:"

La función retorna el valor leído.

La segunda: escribir_temperatura(escala, temp, escala_conv, temp_conv)

Esta función recibe 4 parámetros. Los dos primeros **escala** y **temp** que indica la temperatura en la escala inicial, los dos últimos **escala_conv** y **temp_conv** la escala y temperatura en la escala a la que se convirtió. Los valores escala y escala_conv toman los valores siguientes: 1 es ºC, 2 es ºF y 3 es ºK.

Esta función escribirá el siguiente mensaje acorde a los valores de escala:

temp escala equivale a **temp_conv escala_conv**

donde temp y temp_conv son los valores de las temperaturas y escala y escala_conv se deben escribir acorde a su valor, por ejemplo para estos valores: temp = 32, temp_conv = 100, escala = 2 escala_conv = 1

el mensaje a mostrar sería: 32 ºF equivale a 100 ºC

Ejercicio 03:

Se sabe que al lanzar dos dados normales (valores de 1 al 6) se pueden tener valores de suma obtenida que van desde el 2 hasta el 12.

Se pide que diseñe un programa en Python con las dos funciones siguientes:

calculo_combinaciones()

esta función se encarga de calcular el número de maneras distintas en obtener cada uno de los valores de la suma de los dos dados.

Por ejemplo, el valor 2, sólo se obtiene de sacar 1 en los dos dados, el valor 4 se puede obtener de 3 maneras distintas:

Dado1: 1 Dado2: 3,
Dado1: 2 Dado2: 2,
Dado1: 3 Dado2: 1.

Esta función debe retornar un diccionario donde la **clave** representa la suma de los puntos de los dos dados (2 al 12) y el valor el número de maneras distintas en que se puede obtener esa puntuación.

calculo_probabilidades(ocasiones_por_valor)

esta función recibe como parámetro el diccionario que se ha retornado de la función anterior y debe calcular la probabilidad de que al lanzar los dos dados salga cada uno de los valores del 2 al 12.

Para ello se debe recordar que la probabilidad de que un suceso ocurra se calcula como

P = numero de casos posibles / total de casos probables

En el diccionario se tiene guardado en numero de casos posibles para cada uno de los posibles valores de la suma.

Esta función debe retornar un diccionario similar al de la función anterior, salvo que ahora el **valor** es la probabilidad de que salga cada uno de las claves.

El programa principal se encarga de las llamadas a las funciones y de escribir los valores de los diccionarios con los resultados obtenidos.

Ejercicio 04:

La multiplicación rusa:

Supongamos que se quiere multiplicar x * y (x es el multiplicador e y es el multiplicando). El método ruso de multiplicar consiste en multiplicar sucesivamente por 2 el multiplicando y dividir por 2 el multiplicador hasta que el multiplicador tome el valor 1. Luego, se suman todos los multiplicandos correspondientes a los multiplicadores impares.

Dicha suma es el producto de los dos números. La siguiente tabla muestra el cálculo realizado para multiplicar 37 por 12, cuyo resultado final es 12 + 48 + 384 = 444.

Multiplicador	Multiplicando	Multiplicador impar	Suma
37	12	sí	12
18	24	no	
9	48	sí	60
4	96	no	
2	192	no	
1	384	sí	444

Diseñe un programa que implemente mediante una función de nombre

multiplicación_rusa(multiplicando, multiplicador) y retorne el resultado de la multiplicación.

El programa principal debe leer los números a multiplicar.

Ejercicio 05:

Un analista financiero lleva un registro del precio del euro día a día, y desea saber cuál fue la mayor de las alzas en el precio diario a lo largo de ese período.

Escriba un programa que pida al usuario ingresar el número *n* de días, y luego el precio del dólar para cada uno de los *n* días.

El programa debe entregar como salida cuál fue la mayor de las alzas de un día para el otro.

Si en ningún día el precio subió, la salida debe decir: No hubo alzas.

Para ello el programa debe implementar dos funciones:

lectura_cotizacion()

se encarga de leer las cotizaciones para un número de días determinados.

mayor_alza()

esta función trabaja con la lista de cotizaciones y calcula

Ejercicio 06:

El día juliano correspondiente a una fecha es un número entero que indica los días que han transcurrido desde el 1 de enero del año indicado. Queremos crear un programa principal que al introducir una fecha nos diga el día juliano que corresponde. Para ello podemos hacer las siguientes subrutinas:

• LeerFecha : Nos permite leer por teclado una fecha (día, mes y año).
• DiasDelMes : Recibe un mes y un año y nos dice los días de ese mes en ese año.
• EsBisiesto : Recibe un año y nos dice si es bisiesto.
• Calcular_Dia_Juliano : recibe una fecha y nos devuelve el día juliano.

El calendario juliano es un método para identificar el día actual a través de la cuenta del número de días que han pasado desde una fecha pasada y arbitraria. El número de días se llama *día juliano*, abreviado como DJ. El origen, DJ=0, es el 1 de enero de 4713 A.C. (o 1 de enero de -4712, ya que no hubo año 0). Los días julianos son muy útiles porque hacen que sea muy sencillo determinar el número de días entre dos eventos, solo con restar los números de sus días julianos. Hacer ese cálculo con el calendario normal (gregoriano) es muy difícil, ya que los días se agrupan en meses, que contienen un número variable de días, complicado además por la presencia de los años bisiestos.

La conversión entre el calendario normal (gregoriano) y los días julianos y viceversa, es mejor que sea realizada por un programa escrito concretamente para ello, como el que proporciona KStars Calculadora astronómica. Sin embargo, para quien pueda estar interesado, este es un ejemplo sencillo de conversión entre los calendarios gregoriano y juliano:

DJ = D - 32075 + 1461*(Y + 4800 + (M - 14) / 12) / 4 + 367*(M - 2 - (M - 14) / 12 * 12) / 12 - 3*((Y + 4900 + (M - 14) / 12) / 100) / 4

donde D es el día (1-31), M es el mes (1-12) y A es el año (1801-2099). Tenga en cuenta que esta fórmula solo funciona entre los años 1801 y 2099. Otras fechas anteriores requieren transformaciones más complicadas.

Un día juliano de ejemplo es: DJ 2440588, que corresponde al 1 de enero de 1970.

Ejercicio 07:

Crea un función “ConvertirEspaciado”, que reciba como parámetro un texto y devuelve una cadena con un espacio adicional tras cada letra. Por ejemplo, “Hola, tú” devolverá “H o l a , t ú “. Crea un programa principal donde se use dicha función.

Ejercicio 08:

Definir una función superposicion(lista1, lista2) que tome dos listas y devuelva True si tienen al menos 1 miembro en común o devuelva False en caso contrario. Escribir la función usando un bucle for anidado.

Deberá definir también una función leer_datos() que permita leer los datos y guardarlos en una lista. La función debe retornar la lista creada.

El programa principal debe realizar la lectura de los datos y ejecutar la función superposición.

Ejercicio 09:

Modificar la función anterior para que ahora la función superposición() devuelva una lista con todos los elementos que sean comunes a las dos listas

Ejercicio 10:

Escribir dos funciones que permitan calcular:

• La cantidad de segundos en un tiempo dado en horas, minutos y segundos.
• La cantidad de horas, minutos y segundos de un tiempo dado en segundos.

Escribe un programa principal con un menú donde se pueda elegir la opción de
convertir a segundos, convertir a horas,minutos y segundos o salir del programa.

Ejercicio 11:

Un gimnasio pretende realizar un sistema informático para gestionar los cobros de los servicios que presta a sus clientes, por lo que se pide implementar la siguiente función:

def realizar_cobro (tipo_servicio, nombre_cliente, saldo, edad, categoria):

El gimnasio define sus precios en base a 4 categorías: básico, especial, premium y complementario, este último es gratis para todos los clientes.

Cada cliente tiene una tarjeta donde carga un monto a gastar y desde donde se le descuentan los servicios que utiliza de acuerdo al precio del servicio y ciertos descuentos. Estos descuentos dependen de la categoría del cliente al que pertenece (descuento porcentual) y la edad del cliente (descuento bruto, que se aplica luego del descuento porcentual). Si un cliente no tiene suficiente carga en su tarjeta, no de la realiza el cobro, ya que no se le permite usar el servicio. Se pide escribir el código para las 6 funciones siguientes que facilitarán el proceso del cobro:

1. def valor_servicio (tipo_servicio):

   Retorna el valor base de un servicio

2. def descuento_proporcional_por_categoria (categoria):

   Retorna el descuento proporcional por categoría.
   Por ejemplo, si el descuento es del 25 % debe retornar 0.25
   La categoría viene dada como string .

3. def descuento_bruto_por_edad (edad):

   Retorna el descuento bruto por edad

4. def valor_cobro (tipo_servicio, edad, categoria):

   Retorna el valor de cobro por servicio incluyendo descuentos

5. def realizar_cobro (tipo_servicio, nombre_cliente, saldo, edad, categoria) :

   Realiza cobro según tipo de servicio, y retorna el saldo restante en la tarjeta
   Aplica los descuentos correspondientes y despliega mensaje indicando ¿cuanto cobró a la persona identificada por su nombre?

6. def consumos_posibles (nombre_cliente, saldo, edad, categoria :

   Despliega mensaje indicando cuántas veces puede consumir una persona (identificada por su nombre) cada uno de los servicios no gratuitos
   Por ejemplo, una persona con saldo 50.000 de 20 años y de categoría "200" podría consumir 6 veces el servicio básico, 2 veces el especial y 1 vez el premium

Los servicios, descuentos por categoría y descuentos por edad son los siguientes:

Servicio	Valor
básico	$ 10.000
premium	$ 25.000
super premium	$ 45.000
complementario	$ 0

Categoría	Descuento
400	50 %
300	25 %
200	10 %
otros	0 %

Edad	Descuento
Menor a 18 o 60 o más	$ 5.000
Entre 18 y 30	$ 1.000
otros	$ 0

Los valores de los servicios, los descuentos por categoría y los descuentos por edad se deben implementar con un diccionario para cada uno de ellos.

Realizar en Python todas las funciones indicadas.

Ejercicio 12:

Escribe una función para invertir un diccionario. Debe aceptar un diccionario como parámetro y devolver un nuevo diccionario creado de la forma siguiente:

Las claves en el nuevo diccionario, serán los valores del diccionario de entrada y sus valores, será una lista de las claves del diccionario recibido.

Por ejemplo:

Si pasamos como parámetro el diccionario de abajo:

```
{ "key1" : "value1", "key2" : "value2", "key3" : "value1" }
```

should return this dictionary:

```
{ "value1" : ["key1", "key3"], "value2" : ["key2"] }
```

Ejercicio 13:

Queremos crear un programa que trabaje con fracciones a/b. Para representar una fracción vamos a utilizar dos enteros: numerador y denominador .

Vamos a crear las siguientes funciones para trabajar con funciones:

• Leer_fracción : La tarea de esta función es leer por teclado el numerador y el denominador. Cuando leas una fracción debes simplificarla.

• Escribir_fracción : Esta función escribe en pantalla la fracción. Si el dominador es 1, se muestra sólo el numerador.

• Calcular_mcd : Esta función recibe dos número y devuelve el máximo común divisor.

• Simplificar_fracción : Esta función simplifica la fracción, para ello hay que dividir numerador y dominador por el MCD del numerador y denominador.

• Sumar_fracciones : Función que recibe dos funciones n1/d1 y n2/d2, y calcula la suma de las dos fracciones. La suma de dos fracciones es otra fracción cuyo numerador=n1*d2+d1*n2 y denominador=d1*d2 . Se debe simplificar la fracción resultado.

• Restar_fracciones : Función que resta dos fracciones: numerador=n1*d2-d1*n2 y denominador=d1*d2 . Se debe simplificar la fracción resultado.

• Multiplicar_fracciones : Función que recibe dos fracciones y calcula el producto, para ello numerador=n1*n2 y denominador=d1*d2 . Se debe simplificar la fracción resultado.

• Dividir_fracciones : Función que recibe dos fracciones y calcula el cociente, para ello numerador=n1*d2 y denominador=d1*n2 . Se debe simplificar la fracción resultado.

Crear un programa que utilizando las funciones anteriores muestre el siguiente menú:

1. Sumar dos fracciones: En esta opción se piden dos fracciones y se muestra el resultado.
2. Restar dos fracciones: En esta opción se piden dos fracciones y se muestra la resta.
3. Multiplicar dos fracciones: En esta opción se piden dos fracciones y se muestra la producto.
4. Dividir dos fracciones: En esta opción se piden dos fracciones y se muestra la cociente.
5. Salir

Ejercicio 14

Vamos a crear un programa para trabajar con una pila. Una pila es una estructura de datos que nos permite guardar un conjunto de variables. La característica fundamental es que el último elemento que se añade al conjunto es el primero que se puede sacar (LIFO).

Para representar una pila vamos a utilizar una lista de cadenas de caracteres.

Vamos a crear varias funciones para trabajar con la pila:
• longitudPila : Función que recibe una pila y devuelve el número de elementos que tiene.
• estaVaciaPila : Función que recibe una pila y que devuelve si la pila está vacía, no tiene elementos.
• estaLlenaPila : Función que recibe una pila y que devuelve si la pila está llena.
• añadirElementoPila : función que recibe una cadena de caracteres y una pila, y añade la cadena a la pila, si no está llena. Si esta llena muestra un mensaje de error.
• sacarDeLaPila : Función que recibe una pila y devuelve el último elemento añadido y lo borra de la pila. Si la pila está vacía muestra un mensaje de error.
• EscribirPila : Función que recibe una pila y muestra en pantalla los elementos de la pila.

Realiza un programa principal que nos permita usar las funciones anterior, que nos muestre un menú, con las siguientes opciones:

1. Añadir elemento a la pila
2. Sacar elemento de la pila
3. Longitud de la pila
4. Mostrar pila
5. Salir

Ejercicio 15

Vamos a realizar un programa similar al anterior para trabajar con una cola. Una cola es una estructura de datos que nos permite guardar un conjunto de variables. La característica fundamental es que el primer elemento que se añade al conjunto es el primero que se puede sacar.

Ejercicio 16

1. El ministro de educación de Pythonia tiene una política para atender las necesidades que tienen los colegios que están bajo su jurisdicción. Cada vez que se requiere hacer reparaciones en la infraestructura en un colegio, el ministro les sugiere organizar un bingo para que se puedan recaudar los fondos necesarios. Para jugar al bingo, se utilizan cartones que contienen `N` números **distintos** y **desordenados**, representados por strings de la forma `'numero1-numero2-...-numeroN'` con el caracter guión `'-'` como separador de los números. Tener en cuenta que los números x del cartón están en el rango $0 < x \leq 99$, es decir, tienen solo 2 dígitos.

En base a lo anterior, se le solicita:

1. Escriba la función `verificar_numero(numero,carton)` que recibe como parámetro un número entero y un string con un carton de bingo. Esta funcion deberá retornar `True` si el número ingresado está contenido en el cartón de bingo y `False` en caso contrario.

```
>>> verificar_numero(2,'1-12-54-5')
False
>>> verificar_numero(54,'1-12-54-5')
True
```

2. Escriba un programa que simule un juego de bingo para dos jugadores y muestre al ganador. Deberá ingresar como entrada los cartones de cada jugador y luego los números que vayan saliendo en la tómbola. El ganador será aquel jugador que logre marcar todos los números de su cartón. Si ocurre un empate deberá indicar que ambos jugadores ganaron.

   Observaciones: Los datos de la tómbola que se van ingresando serán siempre correctos y no se debe considerar mal uso del programa, como por ejemplo, ingresar números repetidos de la tómbola.

```
Carton del Jugador 1: 1-12-54-5
Carton del Jugador 2: 3-6-12-2
Ingrese numero de la tombola: 1
Ingrese numero de la tombola: 6
Ingrese numero de la tombola: 4
Ingrese numero de la tombola: 3
Ingrese numero de la tombola: 54
Ingrese numero de la tombola: 8
Ingrese numero de la tombola: 5
Ingrese numero de la tombola: 12
El ganador es el Jugador 1
```

```
Carton del Jugador 1: 1-12-54-5
Carton del Jugador 2: 3-6-12-2
Ingrese numero de la tombola: 1
Ingrese numero de la tombola: 6
Ingrese numero de la tombola: 4
Ingrese numero de la tombola: 3
Ingrese numero de la tombola: 54
Ingrese numero de la tombola: 2
Ingrese numero de la tombola: 5
Ingrese numero de la tombola: 12
Ambos jugadores ganan
```

Ejercicio 17:

Una modificación al ejercicio anterior es generar los números de la tómbola de forma aleatoria mediante una función que retorna un valor en el rango establecido para los números.

Ejercicio 18:

2. El servicio de inteligencia de Pythonia ha estado monitoreando los mensajes de texto que recibe el líder del país de Javapolis, y ha detectado un patrón en éstos, per no ha podido decifrar qué dicen los mensajes, puesto que requieren de un servicio rápido. Como han estado celebrando sus fiestas patrias, nadie ha tenido tiempo de implementar una solución, por lo que han acudido a ud para que los apoye en esta importante misión.

Ellos detectaron que el líder recibe muchas palabras, pero sólo las que contienen una **clave** en alguna parte, son consideradas en el mensaje. Además, notaron que cuando quieren dar por finalizado un mensaje, usan la palabra **out**. Notar que en Javapolis, sólo trabajan con letras minúsculas.

El servicio de inteligencia logró crear la función `get_pos(palabra, clave)`, la cual devuelve la posición en la cual comienza la clave dentro de la palabra y en caso contrario retorna `-1`. Notar que las posiciones comienzan desde `0`. A continuación, se presentan ejemplos de cómo funciona `get_pos`.

```
>>> get_pos('foniwida', 'iwi')
3
>>> get_pos('primavera','iwi')
-1
```

En base a lo descrito anteriormente se le solicita a usted lo siguiente:

1. Escriba la función `get_palabra(palabra, clave)` la cual recibe 2 textos. La función debe retornar la palabra desencriptada, es decir, sin la palabra clave en su interior. Si no existe la palabra clave en su interior debe retornar -1. Asuma que la palabra contiene **a lo más una vez** la clave.

```
>>> get_palabra('foniwida', 'iwi')
fonda
>>> get_palabra('iwien','iwi')
en
>>> get_palabra('fiesta','iwi')
-1
```

2. Desarrolle un programa que solicite el ingreso de la clave que se usó para encriptar las palabras y luego solicite el ingreso de palabras hasta que se ingrese el texto **out**. En dicho momento, dejará de pedor palabras y desplegará por pantalla el mensaje oculto, desifrado.

```
Ingrese clave: iwi
Ingrese palabra: i wana be
Ingrese palabra: feliiwices
Ingrese palabra: programar
Ingrese palabra: public
Ingrese palabra: stdout
Ingrese palabra: iwilos
Ingrese palabra: class
Ingrese palabra: primavera
Ingrese palabra: cuatroiwi
Ingrese palabra: cueca
Ingrese palabra: i win
Ingrese palabra: out
El mensaje oculto es: felices los cuatro
```

Ejercicio 19:

3. La comisión organizadora de la semana mechona creó un concurso on line, en el cual sus 3 alianzas deben participar. El concurso consta de una serie de turnos, en los cuales cada alianza debe ingresar una palabra. La palabra con mayor cantidad de vocales gana. Si dos o más alianzas igualan en el máximo de vocales, nadie gana esa ronda. El concurso termina cuando **una** alianza logre ganar x juegos (meta), donde x debe definirse al inicio del juego.

En base a lo anterior, la comisión solicita a usted implementar lo siguiente:

1. Escriba la función `ganador(c1,c2,c3,meta)`, la cual recibe 4 parámetros, la cantidad de juegos ganados por la alianza 1, 2 y 3 y la meta a lograr en el juego. La función debe retornar el número de la alianza ganadora. En caso de no existir ganador, debe retornar el valor entero 0 (cero).

   Nota: Tener en cuenta que no pueden existir empates en las alizanzas al momento de alcanzar la meta.

```
>>> ganador(2,5,3,5)
2
```

```
>>> ganador(1,3,3,5)
0
```

2. Escriba la función `contar(palabra)` que reciba un string `palabra`. La función debe retornar la cantidad de vocales existentes en la `palabra` recibida.

```
>>> contar('paralelepipedo')
7
```

```
>>> contar('str')
0
```

3. Desarrolle un programa que solicite la meta del Juego, y luego solicite las palabras de cada alianza por turno, hasta que exista una alianza que logre la meta. **Asuma que las palabras pueden ser ingresadas en mayúsculas o minúsculas**. A continuación se presenta un ejemplo de cómo debería lucir el programa.

```
Ingrese meta del juego: 3
alianza 1: Paralelepipedo
alianza 2: reunA
alianza 3: salIda
alianza 1: Tartamudo
alianza 2: Mauricio
alianza 3: semana
alianza 1: mUrciElAgO
alianza 2: rescate
alianza 3: Salvavida
alianza 1: temperatura
alianza 2: Sala
alianza 3: Certamen
La alianza ganadora es 1
```