Actividades Parte IV: Funciones

Actividad 63

Escribir una función `mas_larga()` que tome una lista de palabras y devuelva la más larga.

Actividad 64

Escribir una función `filtrar_palabras()` que tome una lista de palabras y un entero `n`, y devuelva las palabras que tengan más de `n` caracteres.

Actividad 65

Escribir un programa que le diga al usuario que ingrese una cadena. El programa tiene que evaluar la cadena y decir cuántas letras mayúsculas tiene.

Actividad 66

Construir un pequeño programa que convierta números binarios en enteros.

Actividad 67

Escribir un pequeño programa donde:

- Se ingresa el año en curso.
- Se ingresa el nombre y el año de nacimiento de tres personas.
- Se calcula cuántos años cumplirán durante el año en curso.
- Se imprime en pantalla.

Actividad 68

Definir una tupla con 10 edades de personas. Imprimir la cantidad de personas con edades superiores a 20.

Puedes variar el ejercicio para que sea el usuario quien ingrese las edades.

Actividad 69

Definir una lista con un conjunto de nombres, imprimir la cantidad de comienzan con la letra a.

También se puede hacer elegir al usuario la letra a buscar.

Actividad 70

Crear una función **`contar_vocales(),`** que reciba una palabra y cuente cuantas letras "a" tiene, cuantas letras "e" tiene y así hasta completar todas las vocales.

Se puede hacer que el usuario sea quien elija la palabra.

Actividad 71

Escriba una función **`es_bisiesto()`** que determine si un año determinado es un año bisiesto.

Un año bisiesto es divisible por 4, pero no por 100. También es divisible por 400

Actividad 72

Dado un mensaje, se debe calcular su costo para enviarlo por telégrafo. Para esto se sabe que cada letra cuesta $10, los caracteres especiales que no sean letras cuestan $30 y los dígitos tienen un valor de $20 cada uno. Los espacios no tienen valor.

Tu mensaje debe ser un **string**, y las letras del castellano (ñ, á, é, í, ó, ú) se consideran caracteres especiales.

```
Mensaje: Feliz Aniversario!
Su mensaje cuesta $190
```

Actividad 73: Análisis de correo electrónico

La empresa RawInput S.A. desea hacer una segmentación de sus clientes según su ubicación geográfica. Para esto, analizará su base de datos de correos electrónicos con el fin obtener información sobre el lugar de procedencia de cada cliente.

En una dirección de correo electrónico, el **dominio** es la parte que va después de la arroba, y el **TLD** es lo que va después del último punto. Por ejemplo, en la dirección `fulanito@alumnos.cesur.com`, el dominio es `alumnos.cesur.com` y el TLD es `.com`

Algunos TLD no están asociados a un país, sino que representan otro tipo de entidades. Estos TLD genéricos son los siguientes:

```
genericos = {'com', 'gov', 'edu', 'org', 'net', 'mil'}
```

Se pide:

1. Escriba la función **`obtener_dominios(correos)`** que reciba como parámetro una lista de correos electrónicos, y retorne la lista de todos los dominios, sin repetir, y en orden alfabético.

2. Escriba la función **`contar_tld(correos)`** que reciba como parámetro la lista de correos electrónicos, y retorne un diccionario que asocie a cada TLD la cantidad de veces que aparece en la lista. No debe incluir los TLD genéricos.

Aquí teneis un ejemplo del funcionamiento de la aplicación:

```
>>> c = ['fulano@usm.cl', 'erika@lala.de', 'li@zi.cn', 'a@a.net',
...      'gudrun@lala.de', 'otto.von.d@lorem.ipsum.de', 'org@cn.de.cl',
...      'yayita@abc.cl', 'jozin@baz.cz', 'jp@foo.cl', 'dawei@hao.cn',
...      'pepe@gmail.com', 'ana@usm.cl', 'polo@hotmail.com', 'fer@x.com',
...      'ada@alumnos.usm.cl', 'dj@foo.cl', 'jan@baz.cz', 'd@abc.cl']

>>> obtener_dominios(c)
['abc.cl', 'alumnos.usm.cl', 'baz.cz', 'cn.de.cl', 'foo.cl',
'hao.cn', 'lala.de', 'lorem.ipsum.de', 'usm.cl', 'zi.cn']

>>> contar_tld(c)
{'cz': 2, 'de': 3, 'cn': 2, 'cl': 8}
```