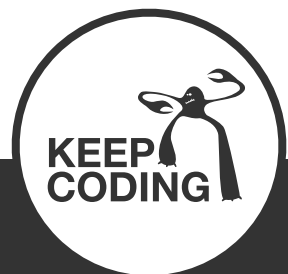




■ Parte 3

Listas, Tuplas y diccionarios



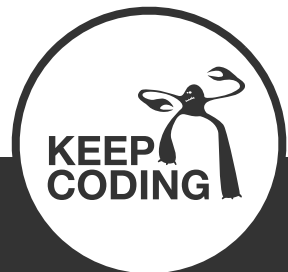
Crear un programa que contenga una lista de 20 números y muestre un rango de la lista. El inicio y fin del rango serán introducidos por el usuario y el programa deberá validar que sean valores válidos.

Ejemplo 1 (entrada no válida)

```
>> Valores: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20  
>> Entradas: 11 y 23  
>> Salida: El rango debe ser de 1 a 20
```

Ejemplo 2 (entrada válida)

```
>> Valores: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20  
>> Entrada: 11 y 15  
>> Salida: 12, 13, 14, 15.
```



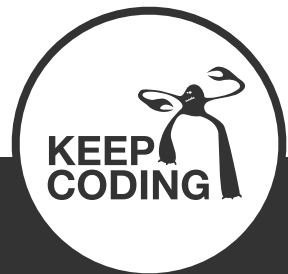
■ KC_EJ14

Crear un programa que reciba un texto y muestre su longitud

Ejemplo

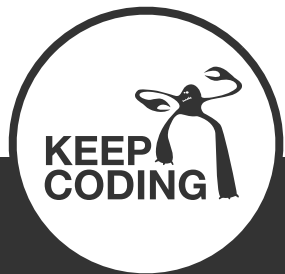
>> Entrada: Hola mundo

>> Salida: Longitud 10



■ KC_EJ15

Crear un programa que reciba el nombre y las calificaciones de 3 personas. Para cada persona deberá guardar la información en una tupla. El programa no mostrará resultados de salida.

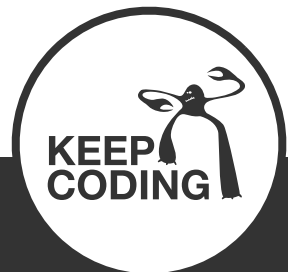


■ KC_EJ16

Crear un programa que contenga una Lista de nombres. Solicitar un índice de la lista y mostrar el valor del índice. El programa deberá validar que el índice es válido.

Ejemplo

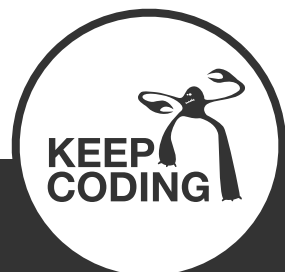
- >> Lista predeterminada: “Pedro Picapiedra”, “Pablo Marmol”, “Bob Esponja”, “Patricio”
- >> Entrada: 2
- >> Salida: Bob Esponja



Crear un programa que contenga un diccionario con nombres y correos electrónicos. Solicitar el nombre de una persona y mostrar su correo electrónico. Indicar con un mensaje apropiado cuando no se encuentre un resultado válido.

Ejemplo

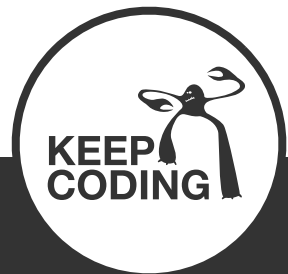
```
>> Diccionario predeterminado:  
    Pedro - pedro.picapiedra@gmail.com  
    Pablo - pmarmol123@gmail.com  
    Bob - bob@gmail.com  
  
>> Entrada: Pablo  
>> Salida: pmarmol123@gmail.com
```





■ Parte 4

Bucles



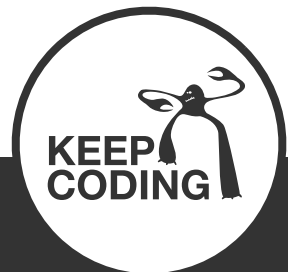
■ KC_EJ18

Crear un programa que muestre los números naturales del 1 al N, donde N será dado por el usuario.

Ejemplo

>> Entrada: 8

>> Salida: 1, 2, 3, 4, 5, 6, 7, 8



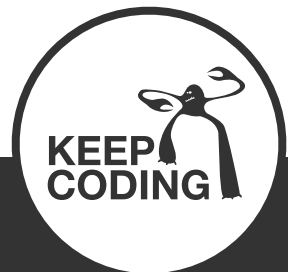
■ KC_EJ19

Crear un programa que almacene 10 números dados por el usuario y muestre únicamente los pares.

Ejemplo

>> Entrada: 10, 3, 6, 12, 20, 5, 9, 7, 11, 40

>> Salida: Son pares 10, 6, 12, 20, 40.



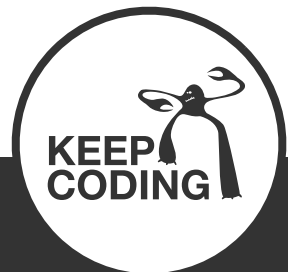
■ KC_EJ20

Crear un programa que escriba la suma de los números del 1 a N, donde N será dado por el usuario.

Ejemplo

>> Entrada: 8

>> Salida: $1+2+3+4+5+6+7+8 = 36$



Crear un programa que reciba un número natural entero y muestre su tabla de multiplicar del 1 al 10.

Ejemplo

>> Entrada: 5

>> Salida:

5x1=5

5x2=10

5x3=15

5x4=20

5x5=25

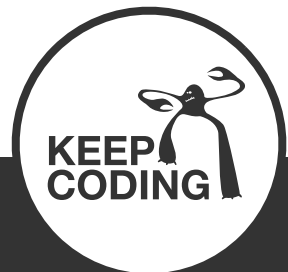
5x6=30

5x7=35

5x8=40

5x9=45

5x10=50



Crear un programa que reciba los nombres y edades de 10 personas. Mostrar únicamente los nombres de las personas que tienen derecho a votar (mayores a 18 años).

Ejemplo

- >> Entradas: Pedro 23, Bob 15, Patricio 7, Pablo 30, Betty 20, Pebbles 2 ...
- >> Salida: Tienen derecho al voto Pedro, Pablo y Betty.



Crear un programa que reciba los el nombre y las calificaciones de N personas, mientras que el usuario no escriba “terminar”. Al terminar deberá mostrar la media de calificaciones de cada persona.

Ejemplo

>> Entradas:

Pedro 6, 8, 7

Pablo 9, 9, 10

Bob 10, 10, 10

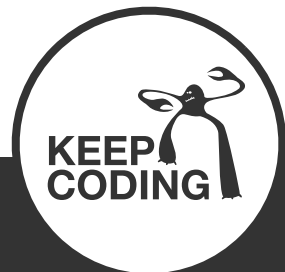
terminar

>> Salida:

Pedro 7

Pablo 9

Bob 10



Modificar el programa **KC_EJ23** para que muestre los resultados ordenados por la media, de forma descendente.

Ejemplo

>> Entradas:

Pedro 6, 8, 7

Pablo 9, 9, 10

Bob 10, 10, 10

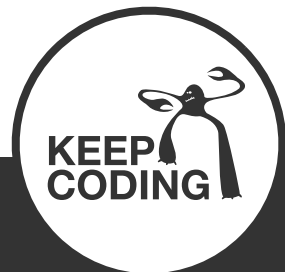
terminar

>> Salida:

Bob 10

Pablo 9

Pedro 7

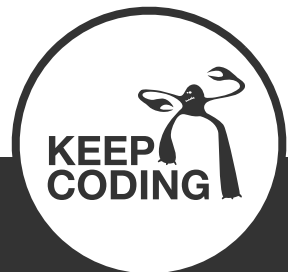


Crear un programa que contenga un número aleatorio del 1 al 100, sin mostrarlo, y permitir que el usuario intente adivinarlo. El usuario solamente tendrá 5 oportunidades, en cada oportunidad fallida se le darán pistas para saber si debe intentar con un número mayor o menor.

Ejemplo

>> Número a adivinar: 32

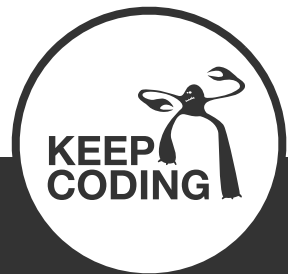
- Entrada: 10
- Salida: Intenta con un número mayor (te quedan 4 oportunidades)
- Entrada: 40
- Salida: Intenta con un número menor (te quedan 3 oportunidades)
- Entrada: 35
- Salida: Intenta con un número menor (te quedan 2 oportunidades)
- Entrada: 32
- Salida: ¡Bien, has adivinado! :D





■ Parte 5

Funciones



Crear un programa con una función `pintar_fila()` que arme las filas de una tabla html. Completar el programa con la estructura de una tabla e invocando a la función N veces, donde N es un valor introducido por el usuario.

Ejemplo

>> Entrada: 4

>> Salida:

```
<table>
```

```
    #ciclo invocando 4 veces a la función pintar_fila()
```

```
    <tr><td></td></tr>
```

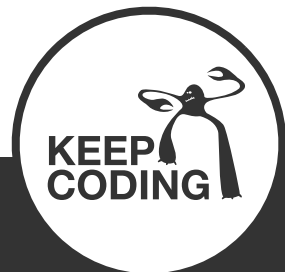
```
    <tr><td></td></tr>
```

```
    <tr><td></td></tr>
```

```
    <tr><td></td></tr>
```

```
    #- - - fin ciclo
```

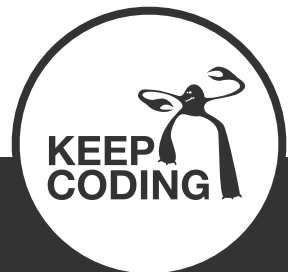
```
</table>
```



Crear un programa que contenga una función `es_palindromo(texto)` y determine si dicho texto es un palíndromo.

Ejemplo

>> Entrada: anita lava la tina
>> Salida: El texto ingresado ES palíndromo



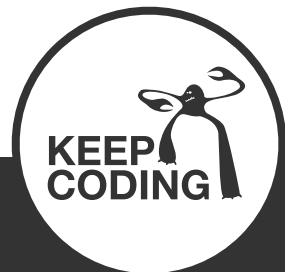
Crear un programa que contenga una función `pintar_rectangulo(lado, figura)`. Esta función deberá pintar en consola un cuadrilátero de lado x lado con la figura proporcionada.

Ejemplo:

>> Entrada: lado 10, figura +

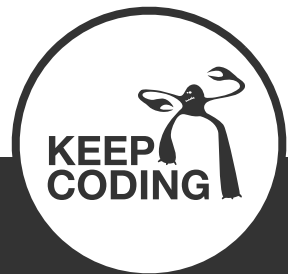
>> Salida:

```
+++++++  
+           +  
+           +  
+           +  
+           +  
+           +  
+           +  
+           +  
+           +  
+++++++
```



■ KC_EJ29

Modificar el programa **KC_EJ24** (promedio de alumnos) de forma tal que el cálculo del promedio se realice a través de una función.

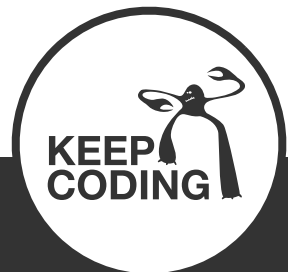


■ KC_EJ30

Crear un programa que simule una máquina expendedora de gaseosas con las siguientes características:

- Las bebidas disponibles son: Fanta, Pepsi, 7Up.
- Todas las bebidas tienen un costo de €1,0
- La máquina recibe monedas de 10, 20 y 50 cent, y €1,0

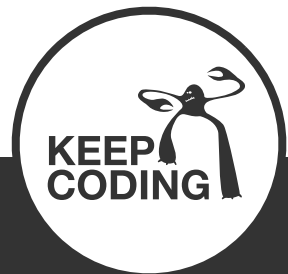
El programa deberá permitir que el usuario seleccione una bebida e ingrese una a una las monedas. El programa deberá detenerse cuando el importe de la gaseosa haya sido completado y, de ser necesario, determinar el sobrante.





■ Parte 6

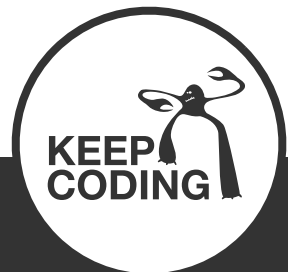
Classes



Crear una clase Alumno con los siguientes atributos:

>> numero_matricula, nombre, apellido, correo_electronico, estatus_inscrito.

La matrícula deberá ser numérica, mientras que correo_electronico, nombre y apellido como textos. El atributo estatus_inscrito deberá ser un valor booleano.



■ KC_EJ32

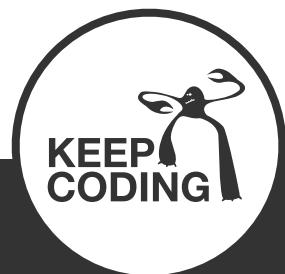
Crear una clase Módulo con los siguientes atributos:

- >> Listado_alumnos, fecha_inicio, fecha_fin.

El listado de alumnos deberá ser tipo Lista y contener objetos de tipo Alumno creado en el ejercicio *KC_EJ31*.

En la misma clase Módulo deberá implementar métodos para

- >> agregar objetos Alumno a la Lista
- >> buscar un alumno
- >> mostrar todos los alumnos con estatus_inscrito == True.





■ Parte 7

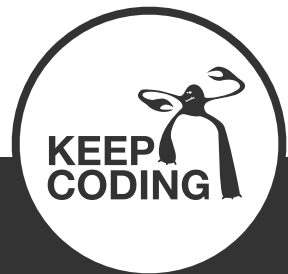
Objetos



■ KC_EJ33

Usando las clases de los ejercicios **KC_EJ31** y **KC_EJ32**, crear un programa que contenga una instancia de la clase Módulo, con instancias de alumnos predefinidos en init. El programa permitirá al usuario:

- >> Ver todos los alumnos enrolados.
- >> Buscar un alumno por matrícula.





■ Parte 8

Herencia



■ KC_EJ34

Crear las clases AlumnoRemoto y AlumnoPresencial, ambas subclases de la clase Alumno creada en el ejercicio **KC_EJ31**.

- >> AlumnoRemoto deberá contar con los atributos numero_matricula, nombre, apellido, correo_electronico, estatus_inscrito, skype, huso_horario.
- >> Mientras que AlumnoPresencial deberá definir los atributos numero_matricula, nombre, apellido, correo_electronico, estatus_inscrito, numero_asiento.

