Clase 19 - Repaso

"""
Ejercicio 1
Escribir una función que dada una cadena de caracteres divida la cadena en pares.
Si la cadena contiene un número impar de caracteres, debería reemplazar el segundo carácter faltante del par final con un guión bajo ('_').
Ejemplo:
de_pares('abc')
retorna ['ab', 'c_']
pares('abcdef')
retorna ['ab', 'cd', 'ef']
"""

```
def solution(s):
        _s = s if len(s) % 2 == 0 else s+"_"
        return [_s[x:x+2] for x in range(0,len(_s),2)]
```

"""
Ejercicio 2
Crear un método is_uppercase(string) que verifica si el string que se pasó por parámetro esta escrito todo en mayúscula.
is_uppercase("c") == False
is_uppercase("C") == True
is_uppercase("hello I AM DONALD") == False
is_uppercase("HELLO I AM DONALD") == True
is_uppercase("ACSKLDFJSgSKLDFJSKLDFJ") ==
False
is_uppercase("ACSKLDFJSGSKLDFJSKLDFJ") == True
"""

```
#UpperCase
def is_uppercase(inp):
        return inp.upper() == inp
```

"""
Ejercicio 3
Escribir la función tribonacci. Funciona igual que la función conocida Fibonacci pero en vez de sumar sus dos números anteriores suma de a tres.
Entonces si la secuencia comienza con [1, 1, 1] la función sigue: [1, 1 ,1, 3, 5, 9, 17, 31, …]
Si comenzamos con [0, 0, 1] entonces sigue con [0, 0, 1, 1, 2, 4, 7, 13, 24, …]
Escribir la función tribonacci que tome la secuencia de los primeros tres números de entrada como lista y la cantidad de elementos a mostrar.
Ejemplo:
Tribonacci([1, 1, 1], 10))
[1, 1, 1, 3, 5, 9, 17, 31, 57, 105])
Tribonacci([0, 0, 1], 10)
[0, 0, 1, 1, 2, 4, 7, 13, 24
Tribonacci([1, 1, 1], 1))
[1]
Tribonacci([300, 200, 100], 0))
[]

"""

```
#Tribonacci
def tribonacci(signature, n):
        if n == 0:
                  return []
          elif n == 1:
                return [signature[0]]
        else:
                return [signature[0]] + tribonacci([signature[1],signature[2],signature[0] + signature[1] +
signature[2]],n-1)
```

"""
Ejercicio 4
Escribir una dos funciones para cifrar/descifrar un string.
La función cifrar recibe un string y retorna el string cifrado.
La función descifrar recibe el string cifrado y lo devuelve descifrado.
El algoritmo para cifrar/descifrar es el siguiente:
Para cifrar y descifrar precisaremos una clave secreta.
La clave secreta es una palabra que debe estar almacenada en un archivo.
Si el caracter a cifrar se encuentra entre las letras de la clave, se sustituye por su posición dentro de la clave.
Si no, simplemente se deja la letra original.

Supongamos que la palabra claves es MURCIELAGO.
ABECEDARIO cifrado 7B535D7249
Trabajar con todas las palabras (tanto clave como string a cifrar/descifrar) en mayúsculas.
Se puede asumir que en la palabra clave no hay caracteres repetidos y su largo no es mayor que 10.
Tener en cuenta las funciones de Python:
upper,
index,
len,
is_digit
"""

```
def codificador(entrada, clave):
        salida = ""
        for letra in entrada:
                if letra in clave:
                        salida = salida + str(clave.index(letra))
                else:
                        salida = salida + letra
        return salida

def decodificador(entrada, clave):
        salida = ""
        for simbolo in entrada:
                if simbolo.isdigit():
                        salida = salida + clave[int(simbolo)]
                else:
                salida = salida + simbolo
        return salida

# leer de un archivo la clave
clave = "murcielago"
entrada = "buenos_dias"
cifrado = codificador(entrada, clave)
descifrado = decodificador(entrada, clave)
print("codificado", cifrado)
print("decodificado", descifrado)
print("------------------------------------------------")
```

"""
Ejercicio 5
Implementar una clase que se llame DefaultList y tenga dos atributos: una lista y un valor predeterminado.
La lista será la lista que corresponde a ese objeto y el valor predeterminado se devolverá cada vez que se llame a un índice de la lista fuera de rango.
En una lista común, cuando queremos acceder a un índice fuera de rango (es decir, i > len (lista) - 1 o i <-len (lista)), se lanza una excepción.
Esta clase debe extender list y para lograr el comportamiento deseado es necesario sobreescribir el método __getitem__(self,index)
"""

```
#DefaultList
class DefaultList(list):
        def __init__(self,lst,default):
                super().__init__(lst)
                self.lst = lst
        self.default = default

def __getitem__(self,index):
        if index < len(self.lst) and index >= -len(self.lst):
                return self.lst[index]
        else:
                return self.default


l = ["hola", 3]
dl = DefaultList(l, -1)

print(dl[0]) # hola
print(dl[2]) #-1
```

"""
Ejercicio 6
Se recomienda beber 1 vaso de agua por bebida con alcohol para no tener resaca a la mañana siguiente.
Supongamos que tenemos como entrada los siguiente bebidas en formato string. Devuelva una cadena que sugiera cuántos vasos de agua debe beber para no tener resaca.
Ejemplos
"1 cerveza" => "1 vaso de agua"
"1 trago, 5 cervezas y 1 copa de vino" => "7 vasos de agua"
Notas
Para simplificar las cosas, consideraremos que cualquier cosa con un número delante es una bebida: "1 oso" => "1 vaso de agua" o "1 motosierra y 2 piscinas" => "3 vasos de agua "
El número delante de cada bebida se encuentra en el rango [1; 9]
"""

```
#Hidratar
def hydrate(drink_string):
        glases = 0
        for x in drink_string:
                if x in ['1','2','3','4','5','6','7','8','9']:
                        glases +=int(x)
        return f'{glases} vasos de agua' if glases > 1 else f'{glases} vasos de agua'
```