cambiar de idioma

Windows + Espacio

algoritmo de ordenamiento burbuja

```
lista = [1,45,6,66,3,2,3,56]

def buble(lista):
    cambio = True
    it = 0
    n = len(lista)
    while cambio:
        cambio = False
        for i in range(n-1):
            if lista[i] > lista[i+1]:
                cambio = True
                lista[i],lista[i+1] = lista[i+1],lista[i]
            it += 1
    return lista,it

print(buble(lista))
```

ejemplo elementos

```
coches =["mercedes", "Alfa", "Romeo", "Fiat", "Lamborghini", "Maserati", "Honda", "Lexus",
"Nissan", "Toyota"]
```

valores no concretos.

```
def concatena(*lista):
    palabraSalida =""
    for i in lista:
          if (type(i) == str)
               palabraSalida += i
     else:
             palabraSalida += str(i)
return palabraSalida

print (concatena(valores))
```

_ : significa que no tiene importancia o que no es necesario un valor
\n : representa un salto de línea. Cuando se incluye \n en una cadena, le indica al programa que debe pasar a la siguiente línea al mostrar el texto.

Conjuntos

Es una coleccion de elementos no ordenada , se utiliza cuando no quieres tener un ordenas , no quieres indicides y ademas no se pueden duplicar los items dentro del conjunto.
Se utilizan llaves , y se pueden llamar a cada item con bucles "for "

No se pudeen modificar sin embargo si se pueden añadir y borrar items
Se puede utilizar el metodo len() , y para añadir el metodo add() o update() , con este podemos añadir elementos de otro conjunto a un nuevo conjunto. Remove() , discard() , pop() se utilizan para borrar elementos pero a distintos rangos. DEL para borrar el set y clear para limpiarlo . Si queremos copiarlo lo podemos hacer utilizando el metodo copy() , y para unir 2 conjuntos utilizariamos el metodo update() o union(), pero el metodo union paara lo que me devuelva uno existente. el metodo difference () ,nos hace un nuevo conjunto con los elementos no comunes de dos conjuntos anteriores. x.diferente.y. Tenemos el metodo diference_update() , se actualiza el conjunto el cual lo has llamado con la diferencia. El metodo intersecion te hace un conjunto de los items comunes. El metodo interseccion_update te actualiza en el conjunto ya existente con el item repetido o comun. Despues para saber de manera booleana si existen conjuntos comunes o no se utliza el metodo isdisjoint() , True cuando no haya elementos comunes y false cuando si haya almenos 1. El metodo issubset () el metodo comprueba si los sets que compara son un subconjunto de otro si lo es le pone TRUE sino FALSE . El metodo superset(). Simetric_difference () , devuelve un conjunto de los elementos no comunes.

ACTIVIDADES

#Desarrolla un programa que tome una cadena y reemplace todas las palabras de cuatro letras o más con asteriscos (*).

En esta actividad el problema que he tenido es de vision dado que salucion es utilizar una lista para las palabras que tengan asteriscos , despues utilizar un bucle que añada las palabras en una variable que compruebe cada una de las palabras del texto anterior que hemos divivido por palabras . Despues de esto hacemos un if que lo que hace es si la palabra contiene mas de 4 caracteres , entonces añade la palabra a la lista , y ademas la remplaza con asteriscos. Como hacemos esto , dicendo quiero que reemplaces por astericos , el numero de veces de la cantidad o longitud de palabras que tenga la variable palabra a la cual estamos definiendo antes. Si no pasara nada de esto entonces la dejamos intacta. Y para finalizar añadimos al texto las palabras de la lista.

```
def remplazar_letras(texto):
    palabras = texto.split()
    palabras_remplazadas = []
    for palabra in palabras:
        if len(palabra) >= 4:
            palabras_remplazadas.append("*" * len(palabra))
        else:
            palabras_remplazadas.append(palabra)
    return " ".join(palabras_remplazadas)

texto = "Esta es mi cadena con no demasiadas letras porque las estoy sustituyendo por asteriscos"
texto_remplazado = remplazar_letras(texto)
```

print(texto_remplazado)

ACTIVIDADES

"""Crea una función que tome una cadena de texto y devuelva una versión en la que las palabras estén ordenadas alfabéticamente, pero los caracteres en cada palabra se mantengan en su orden original"""

El problema que he tenido en este problema es que no he distinguido una lista de una cadena de texto , entonces en este caso debia de utilizar diferentes metodos primero dividir cada palabra de la cadena con el metodo split(), despues ordenarla alfabeticamente con el metodo sort() , y ademas ignorar las mayusculas y las minusculas con una lambda muy sencilla. Y al terminar unirlo todo con un join

```
def ordenadas_alfabeticamente(texto):
    palabras = texto.split()
    palabras.sort(key=lambda x: x.lower())
    return ' '.join(palabras)

texto = "Aqui esta la cadena de texto que tu necesitas"
texto_ordenado = ordenadas_alfabeticamente(texto)
print(texto_ordenado)
```

Diccionarios

un diccionario es un mapa de pares clave de valor. Cada clave es un identificador unico e inmutable, los valores son mutables , se utilizan en base de datos.

Para crear los diccionarios utilizamos las llaves { } , ejemplo dic1 = { "jose" : "ford" , "modelo" : "mustang", "año" : "2005" }

Otro metodo para acceder a un par clave valor es get() y la clave, dic1["años"] = 2022

Los diccionarios son mutables por lo tanto se pueden modificar , entonces podemos añadir y eleminar elementos . Para añadir es tan sencillo como dic1 ["color"] = "rojo" y para eleminarlos utilizamos el metodo pop, popitem o del , en el metodo popitem() eleminamos el ultimo elemento guardado

Para copiar un diccionario podemos utilizar el metodo copy() o el metodo dict() se utilizan igual.

Tambien se pueden hacer diccionarios anidados , ejemplo myfamily {
"child1" : {
"name" : "Emil",
"year" : 2004 } ,
"child2" : {
"name" : "Tobias",

"year" : 2007 },
"child3" : {
"name" : "Linus"
"year" : 2011 }
}

El metodo fromkeys() , nos permite crear un nuevo diccionario a partir de claves de un diccionario ya existente. Ej : dic1 = dict.fromkeys (x , y)

El metodo items() , nos devuelve un objeto vista el cual es una lista compuesta por tuplas.

El metodo keys() , nos devuelve un objeto vista que son las claves del diccionario ,

El metodo values() , nos devuelve un objeto vista, que es una lista de cuyos elementos que son valores del diccionario.

TUPLAS

Es una lista pero inmutable, es decir no se puede modificar. Lo interesante de las tuplas es que utilizan datos heterogeneos.

Si queremos modificar una tupla deberiamos crear una lista de la tupla modificar la lista y luego crear una tupla de esa lista

Para crear una tupla de un elemento sera necesario especificar una coma. ejemplo : lista = ("manzana",)

Para unir tuplas es igual que una lista utilizas el operador +

LISTAS:

Hablando de las listas podemos decir unas cuantas cosas:
- Las listas son mutables , es decir que se pueden modificar.

- Para definir una lista utilizamos [ ]

- Podemos jugar con el orden de los paremetros dentro de la lista con print (lista[1]) , recordando que en realidad la posicion 1 es la cero. Tambien podemos elegir el rango con print(lista[3:5]) , el o si no le indicamos el inicio empezara desde la posicion 0 hasta la que pongamos print(lista[:5]) , igualmente pasa al reves

- Para modificar una lista lo unico que tenemos que es añadir el elemento a la lista para esto deberiamos decir la posicion del parametro que queramos modificar por ejemplo lista [1]

- Si queremos comprobarlo podriamos utilizar check = 1 "in" lista

- Si queremos añadir un elemento a una lista podemos utilizar 2 metodos , append() , para añadir ele elemento al final de la lista en la ultima posicion "lista.append(2)" , o con el metodo insert() , que podemos decir como primer argumento en posicion queremos que se inserte el elemento lista.insert(1, 23)

- Para borrar argumentos podemos utilizar 3 metodos , remove() , lista.remove(23) , el motodo pop(), que borra el indice o el ultimo si no se especifica , lista.pop(1) , y el metodo del te permite borrar una lista entera del lista[0] , tambien la podemos vaciar con el metodo clear que ha diferenecia del del , esta despues de vaciarla sigue existiendo

- Para copiar listas utilizamos el operador + , para unir dos listas , " lista1 + lista2", también podemos utilizar el metodo append() , "list1.append(list2[0])" , o el metodo extend() , para unir dos listas diferentes , "list1.extend(list2)" , tambien podemos copiarlas con list3 = list2.copy

- Se puede cambiar el sentido con el metodo reverse() , list.reverse

- Para saber cuantas veces se repite un elemento en una lista se utiliza el metodo count() , list.count("a")

- Para ordenar la lista en orden ascendente o descendente se utiliza el metodo sort() , list.sort()

- Para averiguar el indice o la posicion de un elemento de la lista utilizamos el metodo index(), list.index("a")

https://docs.python.org/es/3/library/functions.html

LEARN ABOUT FUNCIONS

LAMDBA = tambien llamadas funciones anonimas,

KEYWORDS ARGUMENTS: es cuando defines el argumento , porque el orden importa.

SCOPE = veriamos lo que conoce como el alcance de una variable , basicamente cuanto duran dentro de la funcion dado que no se puede llamar a una variable creada dentro de una funcion desde fuera. LOCAL VARIABLE

VARIABLE GLOBAL = la que esta fuera de la funcion , a python no le importa que tengan el mismo nombre si una es local y otra global

ACTIVIDADES

#Escribe un programa que tome una cadena de texto y reemplace todas las letras vocales con números. Por ejemplo, "aeiou" se reemplazaría por "12345".

```
def remplazar_vocales(s):
    vocales = "aeiouAEIOU"
```

```
    numeros = "12345"
    resultado = ""
    return "".join(numeros[vocales.index(char)]if char in vocales else char for char in s)

cad1 = "Demasiado para mi"
resultado = remplazar_vocales(cad1)
print(resultado)
```

Explicacion
"return ''.join(numbers[vocales.index(char)] if char in vocales else char for char in s)"

for char in s: Esta parte itera sobre cada carácter char en la cadena original s.

if char in vocales else char: Esta parte verifica si el carácter char está en la cadena vocales. Si es así, devuelve el número correspondiente utilizando numbers[vocales.index(char)].Si no es así, devuelve el carácter original char.

numbers[vocales.index(char)]: Esta parte utiliza el índice de la vocal en la cadena vocales para obtener el número correspondiente de la cadena numbers.

''.join(...): Esta parte une todos los caracteres devueltos por la comprensión de lista en una sola cadena utilizando el método join().

ACTIVIDADES

#Desarrolla un programa que tome una cadena y elimine los caracteres no alfabéticos, dejando solo letras y espacios en blanco.

```
import re

def eliminar_no_alfabetico(s):
    return re.sub(r'[^a-zA-Z\s]', '', s)

cad1 = input("Escribe diferentes textos que quieras en una cadena, pueden contener caracteres no alfanumericos")
```

Explicacion

re.sub : viene de la biblioteca "re" , toma 3 argumentos : lo que queremos buscar ,la cadena que queremos remplazar , y la cadena original.

r'[^a-zA-Z\s]' : es lo que queremos buscar, la "r" es la cadena que indica que es un raw (q no se interpretan los caracteres especiales), para buscar los caracteres que quieren reemplazar.

[^a-zA-Z\s]: se lee como cualquier letra que no sea mayuscula o minuscula ni un espacio en blanco.

^ : indica que se busca el conjunto de caracteres que se especifican

a-zA-Z: indica que se buscan letras mayúsculas y minúsculas .

\s: indica que se buscan espacios en blanco. El \ indica que el caracter siguiente tiene un significado especial.

' ': es la cadena de reemplazo. En este caso, se reemplaza cada caracter no alfabético con una cadena vacía (''), lo que efectivamente los elimina.

s: es la cadena original que se pasa como argumento a la función

FUNCION LAMBDA

Debemos utilizar la llamada función lambda cuando no queremos volver a llamar a una función otra vez además también se utiliza para funciones simples.
Una función lambda (también conocida como función anónima) es una función pequeña y concisa que se define sin un nombre explícito.
Para funciones de orden superior , sort(), map(), filter(), reduce().

Sintaxis :
"lambda arguments: expression"
Donde arguments es una lista de variables que se pasan a la función, y expression es la expresión que se evalúa cuando se llama a la función.

Ejemplo:

```
sumar = lambda x, y: x + y
print(sumar(2, 3)) # Output: 5

numbers = [1, 2, 3, 4, 5]
squared_numbers = list(map(lambda x: x ** 2, numbers))
print(squared_numbers) # Output: [1, 4, 9, 16, 25]
```

map () , te los mapea , es un metodo que hace la funcion lambda para cada elemento de la lista que pertenece.

filter() , viene de filtrar y utiliza una condicional , si esto cumple lo que definimos en la funcion lamdba los "reserva"

ACTIVIDADES

#Desarrolla un programa que cuente cuántas palabras hay en una cadena de texto. Puedes asumir que las palabras están separadas por espacios en blanco

```
cad1 = "Este texto es de prueba realmente lo que quiero solo es terminar este trabajo sinceramente , de hecho estoy viendo un podcast mientras hago esto"
print("El numero de palabras que hay en esta cadena es: "+str(len(cad1.split())))
```

---

cómo funciona:

1. La variable cad1 contiene la cadena de texto que se va a analizar.
2. La función split() se utiliza para dividir la cadena en una lista de palabras separadas por espacios en blanco. Por defecto, split() separa la cadena en palabras utilizando cualquier cantidad de espacios en blanco como separador.
3. La función len() se utiliza para contar el número de elementos en la lista generada por split(). En este caso, el número de elementos es el número de palabras en la cadena.
4. La función str() se utiliza para convertir el resultado de len() (un número entero) a una cadena de texto. Esto es necesario porque se va a concatenar el resultado con una cadena de texto utilizando el operador +.
5. El resultado se imprime en la consola utilizando la función print().

La función str() es necesaria en este caso porque se va a concatenar un número entero con una cadena de texto. En Python, no se puede concatenar directamente un número entero con una cadena de texto utilizando el operador +. La función str() se utiliza para convertir el número entero a una cadena de texto, lo que permite la concatenación.

ACTIVIDADES

#Crea una función que reciba una cadena y cuente cuántas letras "a" contiene
string = "El autismo en esta clase es demasiado"
print("Numero de caracteres con a: "+str(string.lower().count('a')))

---

lower(): Convierte la cadena a minúsculas para que la función sea case-insensitive (no distingue entre mayúsculas y minúsculas). De esta forma, se cuentan tanto las letras "a" como las letras "A".
count('a'): Cuenta cuántas veces aparece la letra "a" en la cadena.

ACTIVIDADES

#Crea una función que tome una cadena de texto y devuelva la misma cadena sin vocales.
cad1 = "Esta cadena antes tenia vocales"
vocales = 'aeiouAEIOU'
print(''.join([caracter for caracter in cad1 if caracter not in vocales]))

Explicación:
Caracter for caracter in cadena: Esta es una lista de comprensión que itera sobre cada caracter caracter en la cadena de texto cadena. Es como un bucle for que recorre cada caracter de la cadena.

if caracter not in vocales: Esta es la condición que se aplica a cada caracter. Si el caracter no está en la cadena vocales (es decir, no es una vocal), se agrega a la lista.

[...]: La lista de comprensión se encierra entre corchetes []. La lista contiene todos los caracteres que no son vocales.

".join(...): La función join() se utiliza para unir todos los caracteres de la lista en una sola cadena. El primer argumento '' es la cadena vacía que se utiliza como separador entre los caracteres. En este caso, no se utiliza separador, por lo que los caracteres se concatenan directamente.

TIPOS DE FUNCIONES:
1. Función print() Imprime por consola

2. Función len() - Devuelve el numero de elementos de un objeto (la longitud)

3. Funcion max() - Devuelve el elemento , para que todos los elemento sean del mismo tipo , (el mas grande en un iterable)

4.Funcion min () - Devuelve el elemento , para que todos los elemento sean del mismo tipo , (el mas pequeño en un iterable)

5. Función divmod (a, b) - Devuelve el cociente y el sobrante de dividir dos numeros. Ej : >>> divmod(5,2)

6. Función pow(base, exp) - Retorna la base elevado al exponente.
Ej>>> pow(5,2) = 25

7. Funciones upper() - muestra todo en mayuscula

8. Funciones lower() - muestra todo en minuscula

9. Funciones split() - divide una cadena seperandalo por

10. Funciones Replace() - remplaza en una cadena los valores que nosotros le digamos

11. Funciones Format() - sustituye el formato de una cadena combinandolo por otro.
Ej >>>
var1 = "jose"
var2 = "Maria"
var3 = "Juan"
cad2 = "{alum1} se siente al lado de {alum2}, pero lejos de {alum3}"
print(cad2.format(alum1=var1, alum2=var2, alum3=var3))

12.Funciones Str() , Devuelve una versión recortada de la cadena

13.Funciones join() , Une los elementos de un iterable al final de la cadena

14.Funcion index() , se utiliza para encontrar el indice, El indice es un numero entero que indica la posicion del caracter dentro de la cadena

15. Funcion Sorted() , te ordena una lista, tuped , te la ordena alfabeticamente o si son numeros numericamente , de menor a mayor , tambien puedes cambiar la direccion y el orden (KEY) de manera decendiente.
16.

Python

Tipos de datos numéricos:
INT : abrv de entero ; 1.2.3.4.5.
- para calculos matematicos

Float : abrv flotante ; decimales 1.22
- para calculos matematicos con decimales

Complex : abrv complejo ; para numeros complejos 1x + 2

Tipos de datos de texto

STR : abrv cadena de caracteres , string.
- cadena de texto donde puedes poner comillas simples y dobles , (inmutable)

Tipos Booleano

booleano: true o false ; para las expresiones logicas

Tipos de secuencias:

List: abrv list ; una coleccion ordenada de elementos.
- Puedes almacenar mas de un valor en un solo objeto y acceder a ellos por su posicion de la lista. [ ]

Tuple : abrv tuplas ; similar a las listas , (inmutables) , Puedes almacenar mas de un valor en un solo objeto y acceder a ellos por su posicion de la tupla pero una vez creado no se pueden modificar.

Range: secuencia inmutable de numeros enteros (bucles) ,

Tipos de mapeo:

dict: abrv diccionario , estructura de datos que permite almacenar un conjunto de datos como pares clave-valor, donde cada valor es accesible a traves de una clave unica. LOS DICCIONARIOS PERMITEN ASOCIAR VALORES CON CLAVES.

- Por ejemplo:
{'nombre': 'Juan', 'edad': 25, 'ciudad': 'Madrid'}, {1: 'uno', 2: 'dos', 3: 'tres'}, {(1, 2): 'valor de la tupla', 'clave': [1, 2, 3]}

Tipos de conjuntos:

set: Coleccion de elementos unicos y e inmutables.,