//

//Challenge técnico Tecla.

//¿Cómo realizar el challenge?

// Se debe completar cada una de las funciones para que puedan ser probadas todas juntas al correr el programa en el navegador.

// Importante: No modificar ni el nombre ni los argumentos que reciben las funciones, sólo se debe escribir código dentro de las funciones ya definidas.

// Realizar las ejecuciones de cada una de las funciones a modo de prueba.

function devuelveTexto(array) {

// La funcion llamada 'devuelveTexto' recibe como argumento un arreglo de enteros y de cadena de caracteres llamado 'array'.

// Debe devolver un arreglo con solo las cadenas de caracteres.

// Ej:

// devuelveTexto([1, 'Tecla', 2, 'Teclers']) debe retornar [Tecla, Teclers]

// Tu código aca:

#######################################################################

devuelveTexto = input("hola porfabor ingrese el un texto: ")

devuelveTexto = input("buelba a ingrese el texto: ")

print(f"{texto} {texto2}")

#######################################################################

}

function concatenaTodos(array) {

// La funcion llamada 'concatenaTodos' recibe como argumento un array con dos o mas cadena de caracteres

// debe devolver la concatenación total entre todos los elementos recibidos

// ej:

// sumaTodos([soy,un,tecler]) => "soy un tecler"

// Nota: las cadenas deben estan ordenadas por tamaño de menor a mayor.

// Tu código aca:

#######################################################################

import json

from bson.son import RE\_TYPE

qwe = input('texto1: ')

rty = input('texto2: ')

uio = input('texto3: ')

concatenaTodos = {}

concatenaTodos['arraid'].append({

't1': qwe,

't2': uio,

't3': uio,

})

json\_format = json.dumps(concatenaTodos)

print (json\_format)

print (type(json\_format))

#######################################################################

}

function veoProductos(productos, item) {

// La funcion 'veoInventario' recibe como argumento un array de objetos llamado 'productos' y el nombre de un item llamado 'item'

// Cada objeto tiene una propiedad 'nombre' y 'cantidad'.

// La funcion debe devolver la cantidad de items que hay.

// Si el item no existe la funcion tiene que devolver 0 (cero).

// Ej:

// var inventario = [

// {

// nombre: 'monitor',

// cantidad: 9

// },

// {

// nombre: 'teclado',

// cantidad: 3,

// },

// ]

// veoProductos(productos, 'teclado') devuelve => 3

// Tu código aca:

#######################################################################

from re import S

from flask import Flask, jsonify, request

from flask\_pymongo import PyMongo

from flask\_cors import CORS

app = Flask(\_\_name\_\_)

app.config['MONGO\_URI'] = 'mongodb://localhost/trabajo'

mongo = PyMongo(app)

db =mongo.db.trabajo

@app.route('/pro', methods=['GET'])

def veoProductos():

trabajo = []

for doc in db.find():

trabajo.append({

'\_id': str(ObjectId(doc['\_id'])),

'productos': doc['productos'],

'item': doc['item']

})

return jsonify(trabajo)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=False)

#######################################################################

}

function resto(num) {

// La funcion llamada 'resto' recibe como argumento un numero entero 'num'

// Esta devuelve true o false dependiendo de si el resto de la division es cero. Se debe divir por dos.

// Ej:

// resto(2) devuelve true

// Tu código:

#######################################################################

n1=float(input("Intro número uno: "))

n2=float(input("Intro numero dos: "))

division=n1/n2

if division <= 0:

print("error es menor o igual a cero: ",division)

else:

print("la division es mallor a cerro ",division)

#######################################################################

}

function menosde100(num1,num2) {

// La funcion llamada 'menosde100' recibe como argumento dos numeros enteros 'num1' y 'num2'

// Esta devuelve un arreglo con todos los números de existe entre ambos con un incremental de 1.

// Ninguno de los dos numeros debe superar el 100.

// Si se ingresa un numero mayor en 'num1' que en 'num2' debe devolver el correcto error.

// Ej:

// resto(11,15) devuelve [12, 13, 14, 15]

// Tu código:

#######################################################################

n1=float(input("Intro número uno: "))

n2=float(input("Intro numero dos: "))

suma=n1+n2

if suma >= 100:

print("La suma es menor a 100: ",suma)

else:

print("La suma es menor a 100: ",suma)

#######################################################################

}

function crearClaseCliente() {

// Crear una clase para construir objeto de tipo Cliente.

// el constructor debe recibir:

// nombre (string) , edad (integer) , carrito\_de\_compras (array de objetos) , direcciones (array de strings)

// Esta funcion debe retonar la clase Cliente.

// Tu código:

######################################################################

@app.route('/user', methods=['POST'])

def crearClaseCliente():

print(request.json)

id = db.insert({

'nombre': request.json['nom'],

'edad': request.json['cantidad'],

'carrito\_de\_compras': request.json[''],

'direcciones': request.json['direcciones']

})

return jsonify(str(ObjectId(id)))

#######################################################################

}

function matriz(num1,num2) {

// La función llamada 'matriz' recibe como argumento dos numeros enteros 'num1' y 'num2'.

// Esta función se carga con 'num1' filas y 'num2' columnas con números al azar.

// La función debe devolver la matriz ordenada de forma descendente

// Tu código:

######################################################################

import numpy as np

matris = np.random.random((4,4))

matris

######################################################################

}