Hacer en Python los siguientes ejercicios

EJERCICIOS DE DICCIONARIOS

https://www.freecodecamp.org/espanol/news/guia-de-funciones-de-python-con-ejemplos/

Ejercicio 1: Elabore un algoritmo que permita ingresar un número entero (1 a 10), y muestre su equivalente en romano.

```
def entero_a_romano (numero):
    romanos = {1: 'I', 2: 'II', 3: 'III', 4: 'IV', 5: 'V', 6: 'VI', 7: 'VII', 8: 'VIII', 9: 'IX', 10: 'X'}
    if numero in romanos:
        return romanos[numero]
    else:
        return "Número fuera de rango"

numero_entero = int (input ("Ingrese un número entero del 1 al 10: "))
print ("El equivalente en romano es:", entero_a_romano (numero_entero))
```

Ejercicio 2: Elabore un algoritmo que solicite un número entero y muestre un mensaje indicando la vocal correspondiente, considerando que la vocal A = 1.

```
def asignar_vocal (numero):
    vocales = {1: 'A', 2: 'E', 3: 'I', 4: 'O', 5: 'U'}
    if numero in vocales:
        return vocales[numero]
    else:
        return "Valor Incorrecto"
```

```
numero_entero = int (input ("Ingrese un número entero del 1 al 5: "))
print ("La vocal correspondiente es:", asignar_vocal(numero_entero))
```

Ejercicio 3: Crear un Diccionario de Estudiantes

Crea un diccionario donde las claves sean los nombres de los estudiantes y los valores sean sus calificaciones.

```
def crear_diccionario_estudiantes():
    estudiantes = {
        "Ana": 85,
        "Luis": 92,
        "Carlos": 78,
        "María": 95,
        "Pedro": 88
    }
    return estudiantes

if __name__ == '__main__':
    diccionario_estudiantes = crear_diccionario_estudiantes()
    print(diccionario_estudiantes)
```

Ejercicio 4: Calcular el Promedio de Calificaciones

Escribe una función que calcule el promedio de las calificaciones en un diccionario de estudiantes.

```
python
def promedio_calificaciones(estudiantes):
    return sum(estudiantes.values()) / len(estudiantes)

if __name__ == '__main__':
    estudiantes = {
        "Ana": 85,
        "Luis": 92,
        "Carlos": 78,
        "María": 95,
        "Pedro": 88
    }
    promedio = promedio_calificaciones(estudiantes)
    print(f"El promedio de las calificaciones es: {promedio}")
```

Ejercicio 5: Obtener Estudiantes con Calificaciones Mayores a un Valor

Escribe una función que devuelva los nombres de los estudiantes con calificaciones mayores a un valor dado.

```
def estudiantes_mayores_a(estudiantes, valor):
    return [nombre for nombre, calificacion in estudiantes.items() if calificacion > valor]

if __name__ == '__main__':
    estudiantes = {
        "Ana": 85,
        "Luis": 92,
```

```
    "Carlos": 78,
    "María": 95,
    "Pedro": 88
  }
  valor = 80
  estudiantes_mayores = estudiantes_mayores_a(estudiantes, valor)
  print(f"Estudiantes con calificaciones mayores a {valor}: {estudiantes_mayores}")
```

Ejercicio 6: Añadir un Estudiante al Diccionario

Escribe una función que añada un estudiante y su calificación al diccionario.

```
def añadir_estudiante(estudiantes, nombre, calificacion):
  estudiantes[nombre] = calificacion

if __name__ == '__main__':
  estudiantes = {
    "Ana": 85,
    "Luis": 92,
    "Carlos": 78,
    "María": 95,
    "Pedro": 88
  }
  añadir_estudiante(estudiantes, "Juan", 90)
  print(estudiantes)
```

Ejercicio 7: Eliminar un Estudiante del Diccionario

Escribe una función que elimine a un estudiante del diccionario dado su nombre.

```
def eliminar_estudiante(estudiantes, nombre):
    if nombre in estudiantes:
        del estudiantes[nombre]

if __name__ == '__main__':
    estudiantes = {
        "Ana": 85,
        "Luis": 92,
        "Carlos": 78,
        "María": 95,
        "Pedro": 88
    }
    eliminar_estudiante(estudiantes, "Carlos")
    print(estudiantes)
```

Ejercicio 8: Crear un Diccionario de Inventario

Crea un diccionario que represente el inventario de una tienda, donde las claves son los nombres de los productos y los valores son las cantidades disponibles.

```
python
def crear_inventario():
    inventario = {
```

```
        "Manzanas": 50,
        "Naranjas": 30,
        "Peras": 20,
        "Bananas": 40,
        "Kiwis": 10
    }
    return inventario

if __name__ == '__main__':
    inventario_tienda = crear_inventario()
    print(inventario_tienda)
```

Ejercicio 9: Actualizar la Cantidad de un Producto

Escribe una función que actualice la cantidad de un producto en el inventario.

```
def actualizar_inventario(inventario, producto, cantidad):
    if producto in inventario:
        inventario[producto] = cantidad

if __name__ == '__main__':
    inventario = {
        "Manzanas": 50,
        "Naranjas": 30,
        "Peras": 20,
        "Bananas": 40,
        "Kiwis": 10
```

```
    }
    actualizar_inventario(inventario, "Naranjas", 35)
    print(inventario)
```

Ejercicio 8: Añadir un Nuevo Producto al Inventario

Escribe una función que añada un nuevo producto al inventario.

```
def añadir_producto(inventario, producto, cantidad):
    inventario[producto] = cantidad

if __name__ == '__main__':
    inventario = {
        "Manzanas": 50,
        "Naranjas": 30,
        "Peras": 20,
        "Bananas": 40,
        "Kiwis": 10
    }
    añadir_producto(inventario, "Mangos", 25)
    print(inventario)
```

Ejercicio 11: Eliminar un Producto del Inventario

Escribe una función que elimine un producto del inventario.

```
def eliminar_producto(inventario, producto):
```

```
    if producto in inventario:
        del inventario[producto]

if __name__ == '__main__':
    inventario = {
        "Manzanas": 50,
        "Naranjas": 30,
        "Peras": 20,
        "Bananas": 40,
        "Kiwis": 10
    }
    eliminar_producto(inventario, "Peras")
    print(inventario)
```

Ejercicio 12: Encontrar Productos con Cantidades Menores a un Valor

Escribe una función que devuelva los nombres de los productos con cantidades menores a un valor dado.

```
python
def productos_menores_a(inventario, valor):
    return [producto for producto, cantidad in inventario.items() if cantidad < valor]

if __name__ == '__main__':
    inventario = {
        "Manzanas": 50,
        "Naranjas": 30,
```

```
    "Peras": 20,
    "Bananas": 40,
    "Kiwis": 10
  }
  valor = 25
  productos_menores = productos_menores_a(inventario, valor)
  print(f"Productos con cantidades menores a {valor}: {productos_menores}")
```

Ejercicio 13: Crear un Diccionario de Contactos

Crea un diccionario donde las claves sean nombres de contactos y los valores sean sus números de teléfono.

```
python
def crear_diccionario_contactos():
  contactos = {
    "Ana": "555-1234",
    "Luis": "555-5678",
    "Carlos": "555-8765",
    "María": "555-4321",
    "Pedro": "555-6789"
  }
  return contactos

if __name__ == '__main__':
  diccionario_contactos = crear_diccionario_contactos()
  print(diccionario_contactos)
```

Ejercicio 14: Buscar un Número de Teléfono por Nombre

Escribe una función que busque el número de teléfono de un contacto dado su nombre.

```
def buscar_telefono(contactos, nombre):
    return contactos.get(nombre, "No encontrado")

if __name__ == '__main__':
    contactos = {
        "Ana": "555-1234",
        "Luis": "555-5678",
        "Carlos": "555-8765",
        "María": "555-4321",
        "Pedro": "555-6789"
    }
    nombre = "María"
    telefono = buscar_telefono(contactos, nombre)
    print(f"El número de teléfono de {nombre} es: {telefono}")
```

Ejercicio 15: Añadir un Nuevo Contacto

Escribe una función que añada un nuevo contacto al diccionario de contactos.

```
def añadir_contacto(contactos, nombre, telefono):
    contactos[nombre] = telefono
```

```
if __name__ == '__main__':
    contactos = {
        "Ana": "555-1234",
        "Luis": "555-5678",
        "Carlos": "555-8765",
        "María": "555-4321",
        "Pedro": "555-6789"
    }
    añadir_contacto(contactos, "Juan", "555-0000")
    print(contactos)
```

Ejercicio 16: Eliminar un Contacto del Diccionario

Escribe una función que elimine un contacto del diccionario dado su nombre.

```
def eliminar_contacto(contactos, nombre):
    if nombre in contactos:
        del contactos[nombre]

if __name__ == '__main__':
    contactos = {
        "Ana": "555-1234",
        "Luis": "555-5678",
        "Carlos": "555-8765",
```

```
    "María": "555-4321",
    "Pedro": "555-6789"
  }
  eliminar_contacto(contactos, "Carlos")
  print(contactos)
```

Ejercicio 17: Crear un Diccionario de Notas por Asignatura

Crea un diccionario donde las claves sean nombres de asignaturas y los valores sean listas de notas.

```
python
def crear_diccionario_notas():
  notas = {
    "Matemáticas": [90, 85, 92],
    "Ciencias": [88, 79, 85],
    "Historia": [95, 80, 82],
    "Literatura": [85, 87, 90]
  }
  return notas

if __name__ == '__main__':
  diccionario_notas = crear_diccionario_notas()
  print(diccionario_notas)
```

Ejercicio 18: Calcular el Promedio de Notas por Asignatura

Escribe una función que calcule el promedio de notas para cada asignatura.

```
def promedio_por_asignatura(notas):
    return {asignatura: sum(notas_asignatura) / len(notas_asignatura) for asignatura, notas_asignatura in notas.items()}


if __name__ == '__main__':
    notas = {
        "Matemáticas": [90, 85, 92],
        "Ciencias": [88, 79, 85],
        "Historia": [95, 80, 82],
        "Literatura": [85, 87, 90]
    }
    promedio_asignaturas = promedio_por_asignatura(notas)
    print(promedio_asignaturas)
```

Ejercicio 19: Añadir una Nota a una Asignatura

Escribe una función que añada una nota a la lista de notas de una asignatura dada.

```
def añadir_nota(notas, asignatura, nota):
    if asignatura in notas:
        notas[asignatura].append(nota)


if __name__ == '__main__':
```

```
    notas = {
        "Matemáticas": [90, 85, 92],
        "Ciencias": [88, 79, 85],
        "Historia": [95, 80, 82],
        "Literatura": [85, 87, 90]
    }
    añadir_nota(notas, "Matemáticas", 89)
    print(notas)
```

Ejercicio 20: Crear un Diccionario con Listas de Empleados por Departamento

Crea un diccionario donde las claves sean los nombres de los departamentos y los valores sean listas de nombres de empleados.

```
def crear_diccionario_empleados():
    empleados = {
        "Ventas": ["Ana", "Luis", "Carlos"],
        "Marketing": ["María", "Pedro", "Juan"],
        "Finanzas": ["Sara", "Daniel", "Claudia"]
    }
    return empleados

if __name__ == '__main__':
    diccionario_empleados = crear_diccionario_empleados()
    print(diccionario_empleados)
```

Ejercicio 21: Añadir un Empleado a un Departamento

Escribe una función que añada un empleado a la lista de un departamento dado.

```
def añadir_empleado(empleados, departamento, nombre):
    if departamento in empleados:
        empleados[departamento].append(nombre)

if __name__ == '__main__':
    empleados = {
        "Ventas": ["Ana", "Luis", "Carlos"],
        "Marketing": ["María", "Pedro", "Juan"],
        "Finanzas": ["Sara", "Daniel", "Claudia"]
    }
    añadir_empleado(empleados, "Ventas", "Miguel")
    print(empleados)
```

Ejercicio 22: Eliminar un Empleado de un Departamento

Escribe una función que elimine un empleado de la lista de un departamento dado su nombre.

```
def eliminar_empleado(empleados, departamento, nombre):
    if departamento in empleados and nombre in empleados[departamento]:
        empleados[departamento].remove(nombre)

if __name__ == '__main__':
```

```
empleados = {
    "Ventas": ["Ana", "Luis", "Carlos"],
    "Marketing": ["María", "Pedro", "Juan"],
    "Finanzas": ["Sara", "Daniel", "Claudia"]
}
eliminar_empleado(empleados, "Marketing", "Pedro")
print(empleados)
```

7 EJERCICIOS QUE SE HACEN EN LISTAS Y NO SE PUEDEN EN DICCIONARIO (VER LA EXPLICACIÓN EN CADA UNO):

1. Producto de Elementos Impares:

Explicación: En un diccionario, los valores no están indexados y no tienen un orden secuencial. Además, necesitarías iterar sobre los valores específicos, no las claves.

```
lista = [1, 2, 3, 4, 5, 6]

producto_impares = 1

for num in lista:

   if num % 2 != 0:

      producto_impares *= num

print("El producto de los elementos impares es:", producto_impares)
```

2. Eliminar los Primeros N Elementos:

Explicación: Los diccionarios no tienen un orden intrínseco de elementos, por lo que no puedes eliminar los "primeros N elementos" de la misma manera que en una lista.

```
lista = [1, 2, 3, 4, 5, 6, 7, 8]

N = 3

lista = lista[N:]

print ("Lista después de eliminar los primeros", N, "elementos:", lista)
```

3. Dividir una Lista en dos Partes:

Explicación: Los diccionarios no tienen un orden secuencial, por lo que dividir un diccionario en dos partes no tiene sentido.

```
lista = [1, 2, 3, 4, 5]
mitad = len(lista) // 2
if len(lista) % 2 != 0:
    mitad += 1
lista1 = lista[:mitad]
lista2 = lista[mitad:]
print("Primera parte:", lista1)
print("Segunda parte:", lista2)
```

4. Remover Elementos por Índice:

Explicación: Los diccionarios no usan índices, solo claves, por lo que no puedes eliminar elementos por índice.

```
lista = [1, 2, 3, 4, 5, 6, 7, 8]
lista_pares = [num for i, num in enumerate(lista) if i % 2 == 0]
print("Lista sin elementos en posiciones impares:", lista_pares)
```

5. Invertir el Orden de los Elementos en una Lista:

Explicación: Los diccionarios no tienen un orden intrínseco que pueda invertirse.

```
lista = [1, 2, 3, 4, 5]

lista = lista[::-1]

print("Lista invertida:", lista)
```

6. Eliminar un Elemento por Índice:

Explicación: Los diccionarios no usan índices, solo claves, por lo que no puedes eliminar un elemento basado en su posición.

```
lista = [1, 2, 3, 4, 5]

i = 0

del lista[i]

print(lista)
```

7. Último Elemento con `pop()`:

Explicación: En un diccionario, `pop()` requiere una clave específica, no puedes simplemente eliminar el "último" elemento porque los diccionarios no mantienen un orden de los elementos.

```
lista = [1, 2, 3, 4, 5]

ultimo = lista.pop()

print(ultimo)

print(lista)
```