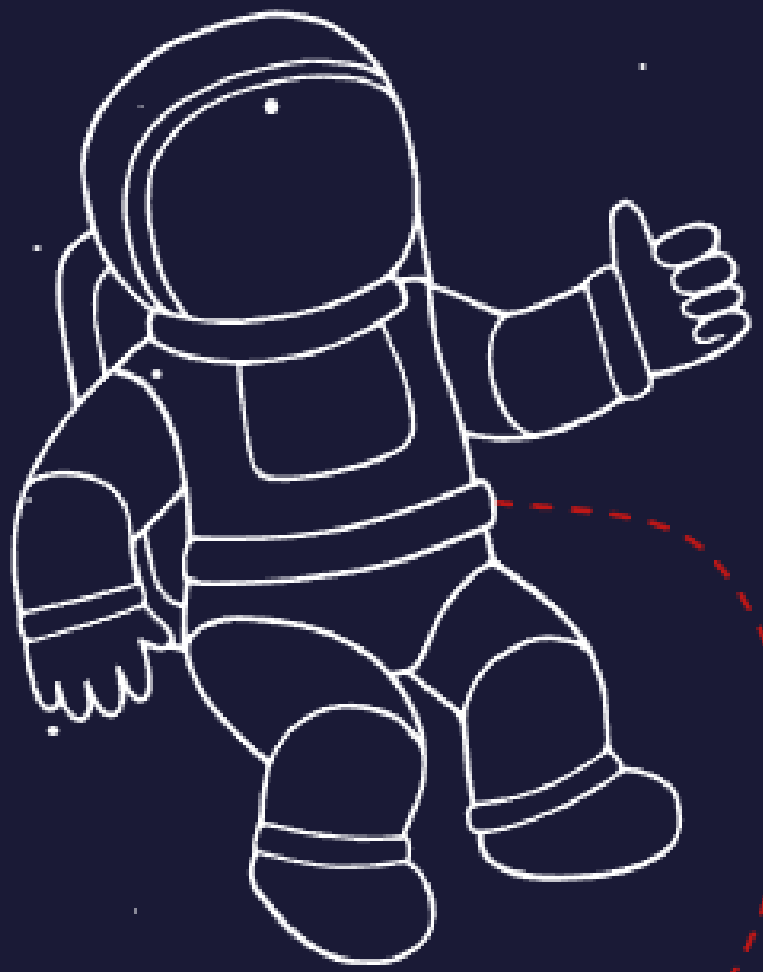


Programa académico CAMPUS

Trainer

Ing. Carlos H. Rueda C.

JS

JavaScript

Set

SET JAVASCRIPT

Es una colección de tipo especial: “conjunto de valores” donde cada valor puede aparecer solo una vez.

Cada valor **solo** puede ocurrir **una vez** en un conjunto.

METODOS DE SET JS

Método	Descripción
new Set()	Crea un nuevo conjunto
add()	Agrega un nuevo elemento
delete()	Elimina un elemento
has()	Devuelve verdadero si existe un valor
forEach()	Invoca una devolución de llamada para cada elemento
values()	Devuelve un iterador con todos los valores
clear()	Elimina todo el contenido

PROPIEDADES DE SET JS

Propiedad	Descripción
size	Devuelve el número de elementos en un Conjunto

METODO NEW SET()

```
// Crea un nuevo conjunto
const letras = new Set(["a","b","c"]);
```

METODO ADD()

```
// Crea un nuevo Set
const letras = new Set();

// Agrega valores al conjunto
letras.add("a");
letras.add("b");
letras.add("c");
```

METODO DELETE()

```
// Borra valores del conjunto
console.log(letras.delete("b")) //true
console.log(letras.delete("b")) //false
```

METODO HAS()

```
// Evalua si existe un elemento en el conjunto
console.log(letras.has("b")) // false
console.log(letras.has("a")) // true
```

METODO FOREACH()

```
// Recorre todos los elementos
let text = "";
letras.forEach (function(value) {
  text += value;
});
console.log(text); // "ac"
```

METODO VALUES()

Devuelve un nuevo objeto iterador que contiene todos los valores en un Conjunto.

Se puede usar el objeto Iterator para acceder a los elementos:

```
// recorre todos los valores
let text = "";
for (const x of letras.values()) {
  text += x;
}
console.log(text); // "ac"
```

METODO VALUES()

Forma más simplificada:

```
// recorre todos los valores
let text = "";
for (const x of letras) {
  text += x;
}
console.log(text); // "ac"
```

PROPIEDAD SIZE

```
// Cantidad elementos del conjunto
console.log(letras.size); // 2
```

METODO CLEAR()

```
// recorre todos los valores
console.log(letras); // Set(2) {'a', 'c'}
console.log(letras); // Set(0) {size: 0}
```

EJEMPLO SETs JS

```
function distaciaLetras(s, distance) {
    const setCad = new Set(s);
    for (const letra of setCad) {
        const pos1 = s.indexOf(letra);
        const pos2 = s.indexOf(letra, pos1 + 1)
        const d = pos2 - pos1 - 1;
        if (d >= 0) {
            const posLet = letra.charCodeAt(0) - 97;
            if(distance[posLet] !== d) return false;
        }
    }
    return true;
}
```

JS

JavaScript

Map

DICCIONARIOS (MAPs) JAVASCRIPT

Los `Map` (*diccionarios en Python*) en Javascript son estructuras de datos nativas que permiten implementar una estructura de tipo **mapa**, es decir, una estructuras donde tiene **valores** guardados a través de una **clave** para identificarlos. Comúnmente, esto se denomina **pares clave-valor**.

METODOS DE MAP JS

Método	Descripción
new Map()	Crea un nuevo diccionario
set(key, value)	Establece o modifica la clave `key` con el valor `value`.
has(key)	Comprueba si `key` ya existe o no.
get(key)	Obtiene el valor de la clave `key` del mapa.
delete(key)	Elimina el elemento con la clave `key` del mapa. Devuelve si lo eliminó correctamente.
clear()	Vacía el mapa completamente.

METODOS DE MAP JS

Método	Descripción
forEach()	Llama a una función para cada par clave/valor de un mapa
keys()	Devuelve un iterable con las claves
values()	Devuelve un iterable con los valores
entries()	Devuelve un iterador con los pares [clave, valor] de un mapa

PROPIEDAD DE MAP JS

Propiedad	Descripción
size	Devuelve el número de elementos en un Mapa

METODO NEW MAP()

```
// Crea un nuevo conjunto
const frutas = new Map([
  ["apples", 500],
  ["bananas", 300],
  ["oranges", 200]
]);
```

METODO SET()

```
const frutas = new Map();

// Agrega valores al diccinario
frutas.set("apples", 500);
frutas.set("bananas", 300);
frutas.set("oranges", 200);
```

METODO GET()

```
console.log(frutas.get("apples")); // 500
```

PROPIEDAD SIZE

```
console.log(frutas.size); // 3
```

METODO DELETE()

```
console.log(frutas.delete("apples")); // true
console.log(frutas.delete("apples")); // false
```

METODO HAS()

```
console.log(frutas.has("apples")); // false
```

METODO FOREACH()

```
let text = "";
frutas.forEach (function(value, key) {
  text += key + ' = ' + value;
})
console.log(text); //bananas = 300oranges = 200
```

METODO ENTRIES()

```
let text = "";
for (const x of frutas.entries()) {
  text += x;
}
console.log(text); //bananas,300oranges,200
```

METODO ENTRIES()

```
let text = "";
for (const [k, v] of frutas) {
  text += k + "=" + v;
}
console.log(text); //bananas,300oranges,200
```

METODO KEYS()

```
let text = "";
for (const k of frutas.keys()) {
  text += k +",";
}
console.log(text); //banana, soranges,
```

METODO VALUES()

```
let text = "";
for (const v of frutas.values()) {
  text += v +",";
}
console.log(text); //300,200,
```

EJEMPLO MAPs JS

```
function distaciaLetras_set(s, distance) {
    let d, posLet, letra;
    const mapLetras = new Map();
    const codeA = 97;
    for (let i=0; i < s.length; ++i) {
        letra = s.charAt(i);
        if (mapLetras.has(letra)) {
            d = i - mapLetras.get(letra, i) - 1;
            posLet = letra.charCodeAt(0) - codeA;
            if (distance[posLet] !== d) return false;
        }
        else
            mapLetras.set(letra, i);
    }
    return true;
}
```

EJERCICIO 1 PROPUESTO MAP

Contar el número de veces que una palabra aparece dentro de un texto.

EJERCICIO 2 PROPUESTO MAP

Escriba un programa WEB que gestione las facturas pendientes de cobro de una empresa.
Las facturas se alamacenarán en un diccionario donde la clave de cada factura será el número de factura y el valor el coste de factura.

- El programa debe permitir ingresar facturas. A medida que se ingresan se muestran en un listado.
- El programa debe permitir pagar una factura existente y se elmina del diccionario.
- Despues de cada operación se debe mostrar un resumen con cantidad de facturas ingresdas, cantidad cobrada y cantidad pendiente de cobro.

NOTA: Ajuste la solución a un desarrollo WEB (entrada por formulario, listado con tablas, etc). El diseño UX se deja a elección.

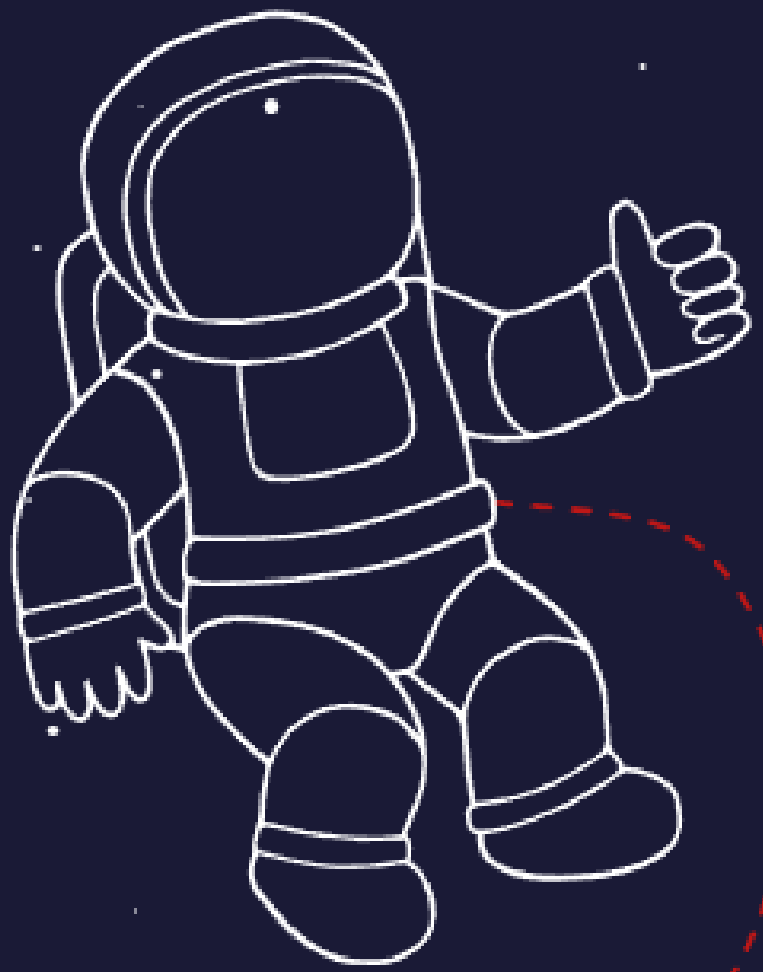
EJERCICIO 3 PROPUESTO MAP

Se realiza la compra de N artículos, en donde se ingresa el código del artículo y la cantidad y mediante el uso de diccionarios para los nombres y valores unitarios de los artículos, el programa debe obtener el nombre de cada artículo, cantidad comprada, valor unitario, valor total de acuerdo a la cantidad comprada y finalmente calcular el valor total de la compra.

Se suministra los diccionarios preestablecidos de nombres de artículo y otro con los valores unitarios de donde tomar la información.

articulos = *{1:"Lapiz",2:"Cuadernos",3:"Borrador",4:"Calculadora",5:"Escuadra"}*
valores = *{1:2500,2:3800,3:1200,4:35000,5:3700}*

NOTA: Ajuste la solución a un desarrollo WEB (entrada por formulario, listado con tablas, etc). El diseño UX se deja a elección.



Programa académico CAMPUS

Trainer
Ing. Carlos H. Rueda C.