Estructuras de datos y funciones

Introducción a los diccionarios (Parte II)

{desafío} latam_

Utilizar estructuras de datos apropiadas para la elaboración de un algoritmo que resuelve un problema acorde al lenguaje Python.

Codificar un programa utilizando funciones para la reutilización de código acorde al lenguaje Python.

- Unidad 1:
  Introducción a Python
- Unidad 2:
  Sentencias condicionales e iterativas
- Unidad 3:
  Estructuras de datos y funciones

Te encuentras aquí

{desafío} latam_

¿Qué aprenderás en esta sesión?

- *Utiliza operaciones de creación y acceso a los elementos de una estructura de datos acorde al lenguaje Python para resolver un problema.*
- *Utiliza operaciones para la agregación, modificación y eliminación de elementos de una estructura de datos acorde al lenguaje Python para resolver un problema.*

{desafío} latam_

¿Para qué sirven los diccionarios?

{desafío} latam_

/* Otros métodos para diccionarios */

Método keys()

Entrega una lista con todas las claves de un diccionario

```
computador = {'notebook': 489990, 'tablet': 120000, 'cargador': 12400}
```

```
computador.keys()
```

```
dict_keys(['notebook', 'tablet', 'cargador'])
```

Método values()

Entrega una lista con todos los valores de un diccionario

```
computador = {'notebook': 489990, 'tablet': 120000, 'cargador': 12400}
```

```
computador.values()
```

```
dict_values([489990, 120000, 12400])
```

Método items()

Entrega una lista con los pares clave-valor de un diccionario

```
computador = {'notebook': 489990, 'tablet': 120000, 'cargador': 12400}
```

```
computador.items()
```

```
dict_items([('notebook', 489990), ('tablet', 120000), ('cargador', 12400)])
```

Método get()

Entrega un mensaje alternativo en caso de no encontrar alguna clave

```
computador = {'notebook': 489990, 'tablet': 120000, 'cargador': 12400}
```

```
computador.get('iphone', 'No se encuentra el elemento solicitado'))
```

```
'No se encuentra el elemento solicitado'
```

/* Otras estructuras de datos */

Tuplas

Par ordenado inmutable

(no se pueden modificar partes de ella, en caso de querer actualizarlo se debe modificar la tupla completa)

```
# definición una tupla
tupla_ej = ("Abril", 2021)
type(tupla_ej)
```

```
tuple
```

Una propiedad útil es lo que se llama **unpacking** o **desempaquetamiento**:

```
# cada valor de la tupla se asigna a month y year respectivamente
month, year = tupla_ej
```

```
# este es una manera alternativa de hacerlo
month, year = "Abril", 2021
```

Sets

Permite trabajar similar a lo que es la teoría de conjuntos.

No permite valores duplicados, por lo que si se necesita conocer sólo valores únicos, esta es la estructura de datos a utilizar.

Este tipo de estructura suele ser bastante útil en **análisis de texto**, para conocer las palabras únicas que existen en él.

```
# definición de un set
# se pueden ver que existen valores repetidos
muchos_animales = {'Gato', 'Perro', 'Tortuga',
                   'Gato', 'Perro', 'Tortuga',
                   'Gato', 'Perro', 'Tortuga',
                   'Gato', 'Perro', 'Tortuga',
                    'Hurón', 'Hamster', 'Erizo de Tierra'}
```

```
# no hay duplicados, sólo valores únicos
print(muchos_animales)
```

```
{'Erizo de Tierra', 'Gato', 'Hamster', 'Hurón', 'Perro', 'Tortuga'}
```

Los sets utilizan llaves {}
al igual que los diccionarios.

La principal diferencia entre los
diccionarios y los sets,
es que los sets solo tienen valores,
no tienen claves.

/* Convertir estructuras */

Convertir un diccionario en una lista

Para lograr esto, se debe utilizar la función items(). Cada par (clave, valor) será una tupla:

```
list({"k1": 5, "k2": 7}.items()) # [('k1', 5), ('k2', 7)]
```

Convertir una lista en un diccionario

Para invertir la transformación se utiliza la función dict.

```
dict([('k1', 5), ('k2', 7)]) # {"k1": 5, "k2": 7}
```

Análogamente existen las funciones **`tuple()`** y **`set()`** que permitirán transformar a tuplas y/o sets respectivamente.

/* Funciones */

dir()

Permite determinar qué métodos están disponibles en cada tipo de dato y/o estructura.

```
var = 'string'
dir(var)
```

```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
```

etc...

Otras funciones en Python

que pueden resultar útiles

sum()

```
var = [1,2,3]
print(sum(var))
```

```
6
```

max()

```
var = [1,2,3]
print(max(var))
```

```
3
```

min()

```
var = [1,2,3]
print(min(var))
```

```
1
```

¿Cuál es la utilidad de la función `dir()`?

Próxima sesión...

- *Guía de ejercicios.*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales