Esta clase va a ser

- grabada

Clase 03. PYTHON

Conjuntos y diccionarios

Temario

02

Listas y tuplas

- ✓ Listas
- ✓ Funciones
- ✓ Tuplas

03

Conjuntos y diccionarios

- ✓ Conjuntos
- ✓ Diccionarios

04

Métodos de colecciones

- ✓ Cadenas
- ✓ Listas
- ✓ Conjuntos
- ✓ Diccionarios

Objetivos de la clase

- **Identificar** un Conjunto
- **Reconocer** similitudes y diferencias entre list y set
- **Agregar y borrar** valores al set
- **Identificar** un Diccionario, agregar y borrar valores al dict

Repositorio Github

Te dejamos el acceso al Repositorio de Github donde encontrarás todo el material complementario y scripts de la clase.

✓ Repositorio Python

CODERHOUSE

Conjuntos

¿Qué son?

Un conjunto o **set** es una colección no ordenada de objetos únicos, es decir, no tiene elementos duplicados. Python provee este tipo de datos por defecto al igual que otras colecciones más convencionales como las **listas, tuplas y diccionarios**.

PARA RECORDAR

Conjuntos

Los conjuntos son ampliamente utilizados en lógica y matemática, y desde el lenguaje podemos sacar provecho de sus propiedades para crear código más eficiente y legible en menos tiempo.

Conjuntos en Python

El conjunto se describe como una lista de ítems separados por coma y contenido entre dos llaves.

Para crear un conjunto vacío debemos decirle set() de lo contrario si quisiéramos hacer como las listas y crearlo con **{}** Python crea un diccionario, el cual veremos más adelante

```
>>> conjunto =  {1, 2, 3, 4}
>>> otro_conjunto = {“Hola”, “como”, “estas”, “?”}
>>> conjunto_vacio = set()   #{ } [ (  ] )
```

Heterogéneos

Heterogéneos

En otros lenguajes, las colecciones tienen una restricción la cual solo permite tener un tipo de dato. Pero en Python, no tenemos esa restricción. Podemos tener un **conjunto heterogéneo** que contenga números, variables, strings, o tuplas.

Ejemplo:

```
mi_var = 'Una variable'
datos = {1, -5, 123.1,34.32, 'Una cadena', 'Otra cadena', mi_var}
print(type(datos))
```

Heterogéneos

Sin embargo, un conjunto no puede incluir objetos mutables como listas, diccionarios, e incluso otros conjuntos o set.

Ejemplo:

```
>>> s = {{1,2}, [1,2,3,4],2}
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'set'
```

Heterogéneos

De la misma forma podemos obtener un conjunto a partir de cualquier objeto **iterable**:

```
set1 = set([1, 2, 3, 4])
print(set1)
set2 = set(range(10))
print(set2)
```

Set

Set

Un set puede ser **convertido** a una **lista y viceversa**. En este último caso, los elementos duplicados son unificados.

```
mi_lista=list({1, 2, 3, 4})
mi_set = set(mi_lista)
print(type(mi_set))

 <class 'set'>
```

List vs. Set

Como hablamos, **las listas son mutables,** sin embargo, el set también es mutable, pero no podemos hacer slicing, ni manejar un set por índice.

Ejemplo:

```
>>> conjunto = {‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’}
>>> conjunto[:3] = [‘A’, ‘B’, ‘C’]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'set' object is not subscriptable
```

Funciones de conjunto

Funciones integradas

En los conjuntos, hay funciones que son muy interesantes e importantes, las funciones integradas.

Los conjuntos en Python tienen muchas funciones para utilizar, entre todas ellas vamos a nombrar las más importantes.

ADD

Add

La primera función de los conjuntos de la que estaremos hablando es **ADD**. Esta función **permite agregar un nuevo ítem al set**. La misma se escribe mi_conjunto.add(ítem_a_agregar)

mi_conjunto sería el set al que se le desee agregar el ítem, e **ítem_a_agregar** sería el ítem que deseemos agregar al set.

Ejemplo:

```
>>> numeros =  {1,2,3,4}
>>> numeros.add(5)
{1,2,3,4,5}
```

Add

No solo acaba ahí. **En la función add también podemos realizar operaciones aritméticas en nuestro ítem**.

Ejemplo:

```
>>> numeros =  {1,2,3,4}
>>> numeros.add(3*2)
{1,2,3,4,6}
>>> numeros.add(3**2+1-12+5*3)
{1,2,3,4,6,13}
```

UPDATE

Update

Para añadir múltiples elementos a un set se usa la función **update()**, que puede tomar como argumento una lista, tupla, string, conjunto o cualquier objeto de tipo iterable. La misma se escribe:
mi_conjunto.update(ítem_a_agregar)

```
numeros =  {1,2,3,4}

numeros.update([5,6,7,8])

numeros.update(range(9,12))

print(numeros)
```

LEN

Longitud del set

¿Se acuerdan cuando hablamos de len en listas? . En set, se puede usar exactamente la misma función para poder saber la longitud de un set, es decir, la cantidad de ítems dentro del mismo.

Ejemplo:

```
>>> numeros =  {1,2,3,4}
>>> len(numeros)
4
>>> datos = {1, -5, 123.34, ‘Una cadena’, ‘Otra cadena’}
>>> len(datos)
5
```

DISCARD

Discard

Si add te deja agregar un ítem al set, **discard** hace todo lo contrario, elimina el ítem del set, sin modificar el resto del set, si el elemento pasado como argumento a **discard()** no está dentro del conjunto es simplemente ignorado.

Se escribe como **mi_conjunto.discard(item_a_descartar)**.

```
>>> numeros =  {1, 2, 3, 4}
>>> numeros.discard(2)
{1, 3, 4}
>>> datos = {1, -5, 123,34, 'Una cadena', 'Otra cadena'}
>>> datos.discard('Otra cadena')
{1, -5, 123,34, 'Una cadena'}
```

REMOVE

Remove

La función **remove** funciona igual al discard, pero con una diferencia, en discard **si el ítem a remover no existe**, simplemente se ignora. En **remove** en este caso **nos indica un error**.

Se escribe como
mi_conjunto.remove(item _a_ remove)

```
>>> numeros =  {1, 2, 3, 4}
>>> numeros.remove(2)
{1, 3, 4}
>>> numeros.remove(5)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 5
```

IN

In

Para determinar si un elemento pertenece a un set, utilizamos la palabra reservada in.

Se escribe como
tem_a_validar in mi_conjunto

```
>>> numeros =  {1, 2, 3, 4}
>>> 2 in numeros
True
>>> 2 not in numeros
False
>>> 4 in numeros
False
```

CLEAR

Clear

Igual que en las listas, podremos borrar todos los valores de un set simplemente usando la función clear.

Se escribe como
mi_conjunto.clear()

¡No se puede asignar un set vacío por que lo toma como diccionario!

```
>>> numeros =  {1, 2, 3, 4}
>>> numeros.clear()
set()
```

POP

Pop

La función pop retorna un elemento en forma aleatoria (no podría ser de otra manera, ya que los elementos no están ordenados). Así, el siguiente bucle imprime y remueve uno por uno los miembros de un conjunto.

```
numeros =  {1,2,3,4}
while numeros:
    print("Se está borrando: ",
numeros.pop())

Se está borrando:  1
Se está borrando:  2
Se está borrando:  3
Se está borrando:  4
```

Sets

Crear un conjunto en Python de Colores

Duración: **15 minutos**

CODERHOUSE

ACTIVIDAD EN CLASE

Sets

Descripción de la actividad.

Trabajaremos con el notebook de la sesión, específicamente sobre la temática de Sets.
Crear un conjunto en Python que posea los siguientes elementos:

- ✔ Colores: Rojo, Blanco, Azul.
- ✔ Posteriormente, agrega nuestro set de colores, los valores de: Violeta y Dorado
- ✔ Elimina a los colores: Celeste, Blanco y Dorado

Pregunta: ¿Qué pasa si queremos eliminar el color Celeste utilizando el método discard?

Break

¡10 minutos y volvemos!

Diccionarios

¿Qué son?

Un diccionario **dict** es una colección no ordenada de objetos. Es por eso que para identificar un valor cualquiera dentro de él, especificamos una **clave** (a diferencia de las listas y tuplas, cuyos elementos se identifican por su posición).

Las **claves** suelen ser **int** o **string**, aunque cualquier otro objeto inmutable puede actuar como una clave. Los valores, por el contrario, pueden ser de cualquier tipo, incluso otros diccionarios.

¿Cómo se crean?

Para crear un diccionario se emplean llaves **{}**, y sus pares clave-valor se separan por comas. A su vez, intercalamos la clave del valor con dos puntos **(:)**

Para crear un diccionario vacío se puede hacer diccionario = {}

```
>>> colores = {"amarillo": "yellow", "azul": "blue", "rojo": "red"}
{"amarillo": "yellow", "azul": "blue", "rojo": "red“}
>>> type(colores)
<class 'dict'>
```

¿Cómo traer valor de Diccionarios?

Para traer el valor de un diccionario se utiliza su clave

```
>>> colores = {"amarillo": “yellow”, "azul": “blue”, "rojo": “red”}
>>> colores[“amarillo”]
“yellow”
>>> colores[“azul”]
“blue”
>>> numeros = {10:”diez”, 20:”veinte”}
>>> numeros[10]
“diez”
```

Mutabilidad

Mutabilidad

Los diccionarios al igual que las listas son **mutables**, es decir, que podemos reasignar sus ítems haciendo referencia con el índice.

```
>>> colores = {"amarillo": “yellow”, "azul": “blue”, "rojo": “red”}
>>> colores[“ amarillo”] = “ white “
>>> colores[“ amarillo”]
“white”
```

Asignación

También permite operaciones en asignación

```
>>> edades = {"Juan": 26, "Esteban": 35, "Maria": 29}
>>> edades[“Juan”] += 5
>>> edades[“Juan”]
31
>>> edades[“Maria”] *= 2
>>> edades[“Maria”]
58
```

Funciones de diccionarios

Funciones de Diccionarios

Al igual que en conjuntos, en los diccionarios encontramos funciones integradas.

Los diccionarios en Python tienen muchas funciones para utilizar. Si bien las desarrollaremos más adelante, a continuación vamos a nombrar las más importantes.

ADD

Add

No hay una función de add, pero para agregar una nueva clave-valor se puede realizar de la siguiente manera:

```
>>> numeros = {'uno': 1, 'dos': 2, 'tres': 3, 'cuatro': 4}
>>> numeros['cinco'] = 5
{“uno”: 1, ”dos”: 2, “tres”: 3, “cuatro”: 4, “cinco”: 5}
```

En este caso, **creamos una nueva clave que no existe “cinco” y asignamos el valor 5.**

UPDATE

Update

Este método actualiza un diccionario agregando los pares clave-valores.

```
>>> numeros =  {'uno': 1, 'dos': 2, 'tres': 3, 'cuatro': 4}
>>> numeros.update({'cinco': 5, 'seis': 6})
{'uno': 1, 'dos': 2, 'tres': 3, 'cuatro': 4, 'cinco': 5, 'seis': 6}
>>> otro_dict = dict(siete=7)
>>> numeros.update(otro_dict)
{'uno': 1, 'dos': 2, 'tres': 3, 'cuatro': 4, 'cinco': 5, 'seis': 6,
'siete': 7}
```

LEN

Longitud del diccionario

¿Se acuerdan cuando hablamos de len en listas? En dict, se puede usar exactamente la misma función para poder saber la longitud de un dict, es decir, **la cantidad de ítems dentro del mismo**.

```
>>> numeros =  {'uno': 1, 'dos': 2, 'tres': 3, 'cuatro': 4}
>>> len(numeros)
4
```

DEL

Del

Del elimina el ítem del dict, sin modificar el resto del dict, si el elemento pasado como argumento a del() no está dentro del dict es simplemente ignorado.

Se escribe como del **mi_dict[“clave”]**.

```
>>> numeros =  {'uno': 1, 'dos': 2, 'tres': 3, 'cuatro': 4}
>>> del numeros['dos']
{'uno': 1, 'tres': 3, 'cuatro': 4}
```

IN

In

Para determinar si un elemento **pertenece** a un dict, utilizamos la palabra reservada **in**.

Se escribe como **clave_a_validar in mi_dict**

```
>>> numeros =  {'uno': 1, 'dos': 2, 'tres': 3, 'cuatro': 4}
>>> 'dos'in numeros
True
>>> 2 not in numeros
True
>>> 4 in numeros
False
```

CLEAR

Clear

Igual que en las listas, podremos borrar todos los valores de un dict simplemente usando la función clear.

Se escribe como **dict.clear()**. Otra forma más cómoda es hacer **mi_dict = {}**

```
>>> numeros =  {'uno': 1, 'dos': 2, 'tres': 3, 'cuatro': 4}
>>> numeros.clear()
{}
```

Repositorio Github

Te dejamos el acceso al Repositorio de Github donde encontrarás todo el material complementario y scripts de la clase.

✓ Repositorio Python

CODERHOUSE

Dicts

Copa Mundial FIFA

Duración: **15 minutos**

CODERHOUSE

ACTIVIDAD EN CLASE

Dicts

Trabajaremos con el notebook de la sesión, específicamente sobre la temática de Diccionarios.

Deberás crear un diccionario que almacene a los ganadores de la Copa Mundial de la FIFA desde el año 1990 al 2018. Y mostrarlo por pantalla.

ACTIVIDAD EN CLASE

Dicts

Datos para la resolución:

- ✔ 1990: 'Alemania',
- ✔ 1994: 'Brasil',
- ✔ 1998: 'Francia',
- ✔ 2002: 'Brasil',
- ✔ 2006: 'Italia',
- ✔ 2010: 'España',
- ✔ 2014: 'Alemania'
- ✔ 2018: 'Francia'

#Codertraining

¡No dejes para mañana lo que puedes practicar hoy! Te invitamos a revisar la Guía de Ejercicios Complementarios, donde encontrarás un ejercicio para poner en práctica lo visto en la clase de hoy.

Actividad extra N° 4

Sets - Dicts

Consigna Sets

Crear un conjunto en Python que posea los siguientes elementos:

- ✓ Países: Inglaterra, USA, México.
- ✓ Posteriormente agrega nuestro set de países, los elementos de: Islandia, Italia, Argentina y Portugal, USA
- ✓ Elimina a los países: Chile e Italia

Pregunta: ¿Qué pasa si queremos eliminar al país Chile utilizando el método **remove**?, ¿Qué pasó con el **element** de USA?

Actividad extra Nº 4

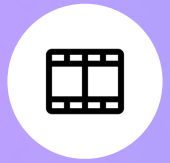
Sets - Dicts

Consigna Dicts

Escribir un programa que le solicite al usuario su nombre, edad, dirección y que, posteriormente, lo muestre por pantalla:

Ejemplo del output solicitado:

- ✓ Juan tiene 25 años, y vive en Carrera 7 - Bogotá



¿Quieres saber más?
Te dejamos material ampliado de la clase

CODERHOUSE

MATERIAL AMPLIADO

Recursos multimedia

Set y funciones

✓ Set y funciones | **Covantec**

Diccionarios y funciones

✓ Diccionarios y recursos| **Covantec**

Disponible en nuestro repositorio.

CODERHOUSE

¿Preguntas?

Resumen
de la clase hoy

- ✓ Listas
- ✓ Tuplas
- ✓ Anidación
- ✓ Transformación de colecciones

Opina y valora esta clase

Muchas gracias.

#DemocratizandoLaEducación