ESTRUCTURAS III
Python

UT 4 Introducción a Python

Baldomero Sánchez Pérez

Frozenset

Los frozenset en Python son una estructura de datos muy similar a los set, con la salvedad de que son inmutables, es decir, no pueden ser modificados una vez declarados.

1. Crear frozenset Python

Los frozensets en Python son un tipo muy parecido a los sets con la salvedad de que son inmutables, es decir, están congelados y no pueden ser modificados una vez inicializados.

```
fs = frozenset([1, 2, 3])
print(fs)       #frozenset({1, 2, 3})
print(type(fs)) #<class 'frozenset'>
```

2. Ejemplos Frozenset

Dado que son inmutables, cualquier intento de modificación con los métodos que ya hemos visto en otros capítulos como add() o clear() dará un error, ya que los frozenset no disponen de esos métodos.

```
fs = frozenset([1, 2, 3])

#fs.add(4)  #Error! AttributeError

#fs.clear() #Error! AttributeError
```

3. Utilidad Frozenset

Los frozenset pueden ser útiles cuando queremos usar un set pero se requiere que el tipo sea inmutable.

Algo no muy común, pero que podría darse, es crear un set de sets. Es decir, un ser que contiene como elementos a otros sets.

El siguiente código daría un TypeError ya que los elementos de un set deben ser por definición inmutables.

```
s1 = {1, 2}
s2 = {3, 4}
#s3 = {s1, s2} # Error! TypeError
```

3. Utilidad Frozenset

```
s1 = {1, 2}
s2 = {3, 4}
#s3 = {s1, s2} # Error! TypeError
```

Para resolver este problema, podemos crear un set de frozensets. Esto si es posible ya que el frozenset es un tipo inmutable.

```
s1 = frozenset([1, 2])
s2 = frozenset([3, 4])
s3 = {s1, s2}
print(s3)
#{frozenset({3, 4}), frozenset({1, 2})}
```

4. Frozenset con Diccionarios

Lo mismo aplica a los diccionarios, ya que su key debe ser un tipo inmutable. Si intentamos crear un diccionario con set como key, obtendremos un TypeError.

```
s1 = set([1, 2])

s2 = set([3, 4])

#d = {s1: "Texto1", s2: "Texto2"} # Error! TypeError
```

Pero si podemos crear un diccionario donde sus key son frozenset, ya que son un tipo inmutable.

```
s1 = frozenset([1, 2])

s2 = frozenset([3, 4])

d = {s1: "Texto1", s2: "Texto2"}

print(d) #{frozenset({1, 2}): 'Texto1', frozenset({3, 4}): 'Texto2'}
```

Cast en Python

Hacer un cast o casting significa convertir un tipo de dato a otro. Anteriormente hemos visto tipos como los int, string o float. Pues bien, es posible convertir de un tipo a otro.

Pero antes de nada, veamos los diferentes tipos de cast o conversión de tipos que se pueden hacer. Existen dos:

- **Conversión implícita:** Es realizada automáticamente por Python. Sucede cuando realizamos ciertas operaciones con dos tipos distintos.
- **Conversión explícita:** Es realizada expresamente por nosotros, como por ejemplo convertir de str a int con str().

Conversión implícita

Esta conversión de tipos es realizada automáticamente por Python, prácticamente sin que nos demos cuenta. Aún así, es importante saber lo que pasa por debajo para evitar problemas futuros.

El ejemplo más sencillo donde podemos ver este comportamiento es el siguiente:

```
a = 1   # <class 'int'>
b = 2.3 # <class 'float'>
a = a + b
print(a)       # 3.3
print(type(a)) # <class 'float'>
a es un int
b es un float
```

Conversión implícita

Pero si sumamos a y b y almacenamos el resultado en a, podemos ver como internamente Python ha convertido el int en float para poder realizar la operación, y la variable resultante es float

Sin embargo hay otros casos donde Python no es tan listo y no es capaz de realizar la conversión. Si intentamos sumar un int a un string, tendremos un error TypeError.

```
a = 1
b = "2.3"
c = a + b
```

 # TypeError: unsupported operand type(s) for +: 'int' and 'str'

Conversión explicita

Por otro lado, podemos hacer conversiones entre tipos o cast de manera explícita haciendo uso de diferentes funciones que nos proporciona Python.

Las más usadas son las siguientes:

float(), str(), int(), list(), set()

Y algunas otras como **hex(), oct() y bin()**

Conversión explicita. Convertir float a int

Para convertir de float a int debemos usar float().

Pero mucho cuidado, ya que el tipo entero no puede almacena decimales, por lo que perderemos lo que haya después de la coma.

```
a = 3.5

a = int(a)

print(a)

# Salida: 3
```

Conversión explicita. Convertir float a string

Podemos convertir un float a string con str(). Podemos ver en el siguiente código como cambia el tipo de a después de hacer el cast.

```
a = 3.5

print(type(a)) # <class 'float'>

a = str(a)

print(type(a)) # <class 'str'>
```

Conversión explicita. Convertir string a float

Podemos convertir un string a float usando float(). Es importante notar que se usa el . como separador.

```
a = "35.5"
print(float(a))
# Salida: 35.5
```

El siguiente código daría un error, dado que , no se reconoce por defecto como separador decimal.

```
a = "35,5"
print(float(a))
# Salida: ValueError: could not convert string to float: '35,5'
```

Conversión explicita. Convertir string a float

Y por último, resulta obvio pensar que el siguiente código dará un error también.

```
a = "Python"

print(float(a))

# Salida: ValueError: could not convert string to float: 'Python'
```

Conversión explicita. Convertir string a int

Al igual que la conversión a float del caso anterior, podemos convertir de string a int usando int().

```
a = "3"
print(type(a)) # <class 'str'>
a = int(a)
print(type(a)) # <class 'int'>
```

Cuidado ya que no es posible convertir a int cualquier valor.

```
a = "Python"
a = int(a)
# ValueError: invalid literal for int() with base 10: 'Python'
```

Conversión explicita. La conversión de int a string se puede realizar con str().

A diferencia de otras conversiones, esta puede hacerse siempre, ya que cualquier valor entero que se nos ocurra poner en a, podrá ser convertido a string.

```
a = 10

print(type(a)) # <class 'int'>

a = str(a)

print(type(a)) # <class 'str'>
```

Conversión explicita. Convertir a list

Es también posible hacer un cast a lista, desde por ejemplo un set. Para ello podemos usar list().

```
a = {1, 2, 3}

b = list(a)

print(type(a)) # <class 'set'>

print(type(b)) # <class 'list'>
```

Conversión explicita. Convertir a set

Y de manera completamente análoga, podemos convertir de lista a set haciendo uso de set().

```
a = ["Python", "Mola"]

b = set(a)

print(type(a)) # <class 'list'>

print(type(b)) # <class 'set'>
```