Colecciones

Fundamentos de Programación

Agenda

- Características de las colecciones
- Tipos de colecciones
- Operaciones con colecciones
- Tuplas, Conjuntos y Diccionarios

Colecciones

Terminología

- **Una colección** es un tipo de dato que agrupa varios elementos en una misma unidad. Se utilizan para almacenar, recuperar, manipular y comunicar una agregación de datos.

- Fuente: https://www.iconfinder.com

Tipos de colecciones

- Python tiene los siguientes tipos de colecciones:
  - Listas
  - Tuplas
  - Conjuntos
  - Diccionarios

Listas

Operar sobre Listas

- Las listas soportan lo siguiente:
  - len(x): retorna el tamaño de la lista
  - 1 in x: retorna True si 1 está dentro de la lista o False caso contrario
  - Métodos propios:
    - sort(), insert(index, valor), pop(), pop(index), append(), reverse(), sort(reverse=True)
    - count(ele),index(ele), clear()

Tuplas

Tuplas

- Las tuplas son muy similares a las listas pero tiene algunas diferencias:
  - Su definición se hace por medio de paréntesis.

```
T = (1, 2, True, 'python')
t1 = tuple(T)

inventario=('amor', 'ira', 'alegria', 'tristeza')

for emo in inventario:
    print(emo)

print('Done')
```

  - Las tuplas no poseen funciones de modificación, es decir, son inmutables.

Tuplas

- Se puede hacer uso de la función len().
- El operador **in** se utiliza igual
- Para acceder a un determinado elemento en una tupla se especifica el número del indice de la posición
- Se puede utilizar la técnica de slicing para acortar una tupla [:]
- Se puede convertir tuplas a listas con la función list()

Tuplas

- Las tuplas no pueden ser modificadas (inmutables).

```
>>>T= (1, 2, True, 'python')
>>>T[0] = 5
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment

>>T = T + ('dir', 40.5)
>>T
(1, 2, True, 'python', 'dir', 40.5)
```

- Sin embargo, si se puede crear nuevas tuplas a partir de unas existentes.

Tuplas: desempaquetar

- Se pueden asignar a multiples variables valores de una tupla:

```
>>julia = ("Julia", "Roberts", 1967, "Duplicity", 2009)
>>(name, surname, b_year, movie, m_year) = julia

>>> b = ("Bob", 19, "CS")
>>> (name, age, studies) = b    # tuple unpacking
>>> name
'Bob'
>>> age
19
>>> studies
'CS'
```

Tuplas: intercambio, retorno

- Se pueden asignar para intercambio de valores entre dos variables

```
temp = a
a = b
b = temp
```

```
(a, b) = (b, a)
```

- Para retornar varios resultados en funciones

```
def f(r):
    # Retorna (circunferencia, area) de un círculo r
    c = 2 * math.pi * r
    a = math.pi * r * r
    return (c, a)
```

Listas vs. Tuplas

- Las listas pueden hacer todo lo que las tuplas y más. Sin embargo, es recomendable usar tuplas en los siguientes escenarios:
  - Las tuplas son más rápidas que las listas.
  - La inmutabilidad de las tuplas las hace apropiadas para la creación de constantes en vista que estas no pueden cambiar.
  - En ocasiones, Python requiere valores inmutables. Por ejemplo, los diccionarios requieren tipos inmutables, por lo tanto las tuplas son esenciales en la creación de este tipo de escenarios.

Conjuntos

Conjuntos

- Los conjuntos se construyen como una lista de valores, no ordenados ni repetidos, encerrados entre llaves {}.

- También se pueden definir conjuntos con la instrucción set(c), en donde c representa cualquier objeto que se pueda indexar, como listas, cadenas o tuplas.

Conjuntos

- Los conjuntos son útiles para realizar operaciones como unión, intersección y diferencia de conjuntos.

```
>>> c1 = {1, 2, 3, 4, 5, 6}
>>> c2 = {2, 4, 6, 8, 10}
>>> c3 = {1, 2, 3}
>>> c4 = set([4, 5, 6])

>>> c1 | c2
{1, 2, 3, 4, 5, 6, 8, 10}

>>> c1 & c2 & c3
{2}
```

```
>>> c1.union(c2)
{1, 2, 3, 4, 5, 6, 8, 10}

>>> c1.intersection(c2,c3)
{2}

>>> c1 - c2
{1, 3, 5}
```

Conjuntos

- No son indexables

```
>>>C = {10, 4, 'python'}
>>>C[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'set' object does not support indexing
```

- Pero se pueden iterar al igual que las listas

```
>>>for ele in C:
       print(ele)
```

Conjuntos

- Se puede hacer uso de la función len().
- El operador **in** se utiliza igual
- NO se puede acceder a un determinado elemento segun la posición
- NO se puede utilizar la técnica de slicing
- Se puede convertir conjunto a listas con la función list()
- Se pueden agregar elementos con .add() o quitar elementos con .remove().
- Se pueden remover elementos con.pop() pero removerá un elemento de manera aleatoria