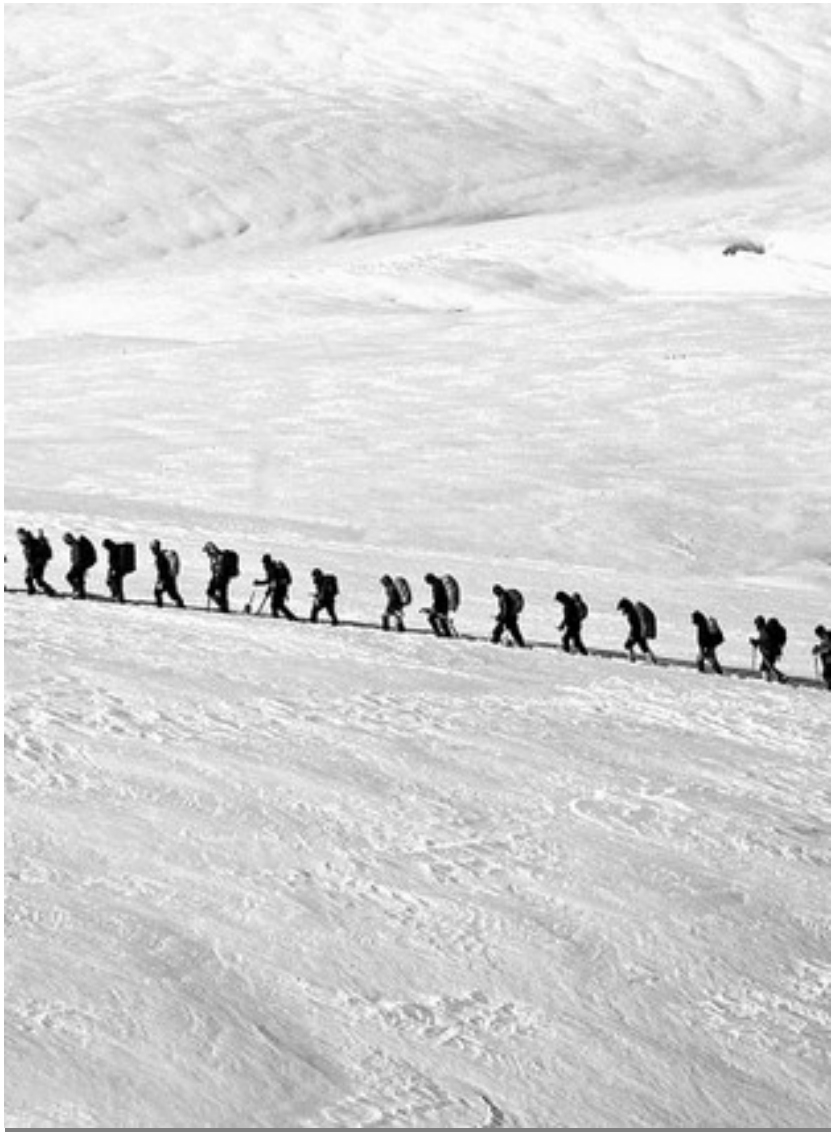
Fundamentos de Programación. Primer Curso de ASIR.

UT04.01 - Estructuras de datos. Tuplas.

UT04.01 - Estructuras de datos. Listas.
1.- Introducción.
1.1.- Estructuras de datos de datos.

- En muchos lenguajes de programación existen estructuras denominadas **arrays**.
- En Python hay estructuras más versátiles que permite manejar **grupos de elementos**, realizando multitud de operaciones: añadir, eliminar, buscar, etc.
- Estas estructuras son las siguientes: l**istas, tuplas y diccionarios**.
- Todas ellas admiten operaciones similares, aunque la diferencia principal está en que se pueda cambiar sus elementos (**mutables**) o no (**inmutables**).
- En este documento trataremos la primera de ellas: **las listas**.



UT04.01 - Estructuras de datos. Listas.
1.- Introducción.
1.2.- Definición de tupla. Características (I).

Definición.

- Los valores guardados en una tupla pueden ser de cualquier tipo, y **son indexados por números enteros**.
- **Las tuplas son inmutables**. Eso implica que una vez creada una tupla no se puede modificar.
- Las tuplas son comparables y dispersables.

Estructura de una tupla.

- Una tupla es una lista de valores separados por comas:

  tupla = ('a', 'b', 'c', 'd', 'e')

  También es posible crear una tupla sin incluir los paréntesis.
- Para crear una tupla con un solo elemento, es necesario incluir una coma al final. En caso contrario la considera una cadena o un número.

  tupla = ('a',)

```
>>> tupla = ("a", "b", "c", "d")
>>> tupla
('a', 'b', 'c', 'd')
>>> tupla = ("a")
>>> tupla
'a'
>>> tupla = ("a",)
>>> tupla
('a',)
```

UT04.01 - Estructuras de datos. Listas.
1.- Introducción.
1.2.- Definición de tupla. Características (II).

- Otra forma de construir una tupla es utilizando la función interna **tuple()**.
  - Sin argumentos crea una tupla vacía.
  - Si el argumento es una cadena, lista, o tupla el resultado de la llamada es una tupla con los elementos que componen el argumento.
- La mayoría de los operadores de listas también funcionan en tuplas.
  - El operador corchete indexa un elemento.
  - El operador de rebanado (slice) selecciona un rango de elementos.
- Si se intenta modificar uno de los elementos de la tupla, se produce un error.
- No se puede modificar los elementos de una tupla, pero sí se puede reemplazar una tupla por otra.

```
>>> tupla = tuple()
>>> tupla
()

>>> tupla = tuple('Prueba')
>>> tupla
('P','r','u','e','b','a')

>>> tupla[2]
'u'

>>> tupla[1:4]
('r', 'u', 'e')

>>> tupla = ("ZZZ",) + tupla
>>> tupla
('ZZZ', 'P', 'r', 'u', 'e', 'b', 'a')
```

UT04.01 - Estructuras de datos. Listas.
2.- Asignación multivariable y multivalor.

- Una utilidad a la hora de programar en Python es la **asignación de múltiples valores a múltiples variables**.
- Esto es debido a que las múltiples variables y los múltiples valores son consideradas como tuplas. De esta forma se realiza una asignación uno a uno.

```
>>> x, y = 1, 2
>>> x
1
>>> y
2

>>> m = [ 'pásalo', 'bien' ]
>>> (x, y) = m
>>> x
'pásalo'
>>> y
'bien'
```

UT04.01 - Estructuras de datos. Listas.
3.- Comparación de tuplas.

- Las tuplas son comparables entre si.
- Para comparar tuplas se van comparando cada uno de los elementos del mismo orden. Es decir, **se comparan los primeros elementos, si son iguales se procede a comparar los segundos elementos y así sucesivamente**.

```
>>> (0,1,3) < (0,1,4)
True
>>> (2,1,4) > (3,1,4)
False
```

UT04.01 - Estructuras de datos. Listas.
4.- Elemento está en una tupla. Recorrido de tuplas.

Comprobar si un elemento está en una tupla.

- Al igual que en las listas podemos comprobar si un elemento está en una tupla con el operador **in**.
- Analogamente podemos saber si un elemento está en una tupla con **not in**.

Recorrido de una tupla.

- Podemos recorrer una tupla mediante el bucle **for..in**.

```
>>> (2,1,4) > (3,1,4)
False
>>> 4 in (1,3,6,4,6)
True
>>> 7 not in (1,3,6,4,6)
True

>>> (2,1,4) > (3,1,4)
False
>>> 4 in (1,3,6,4,6)
True
>>> 7 not in (1,3,6,4,6)
True

tupla = (3,2,5,7)
suma = 0
for elemento in tupla:
    suma = suma + elemento
print(f"La suma es {suma}.")

La suma es 16.
```

UT04.01 - Estructuras de datos. Listas.
5.- Métodos de tuplas.

- Se pueden usar los mismos métodos de las listas que no cambien el contenido de la tupla.
- Por tanto podemos realizar operaciones de busqueda y de localización de elementos dentro de la tupla.
- En ellos están los siguientes:

tupla = (1,4,2,4,5)			
Método	**Función**	**Ejemplo**	**Resultado**
tupla.count(elemento)	Cuenta las ocurrencias de elemento en la tupla.	tupla.count(4)	2
tupla.index(elem, ind)	Busca el elemento y devuelve el índice en el que se ha encontrado. Si se añade un índice se empieza a buscar a partir de ese índice. En el caso de no encontrarse, se devuelve un ValueError.	tupla.index(5)	4