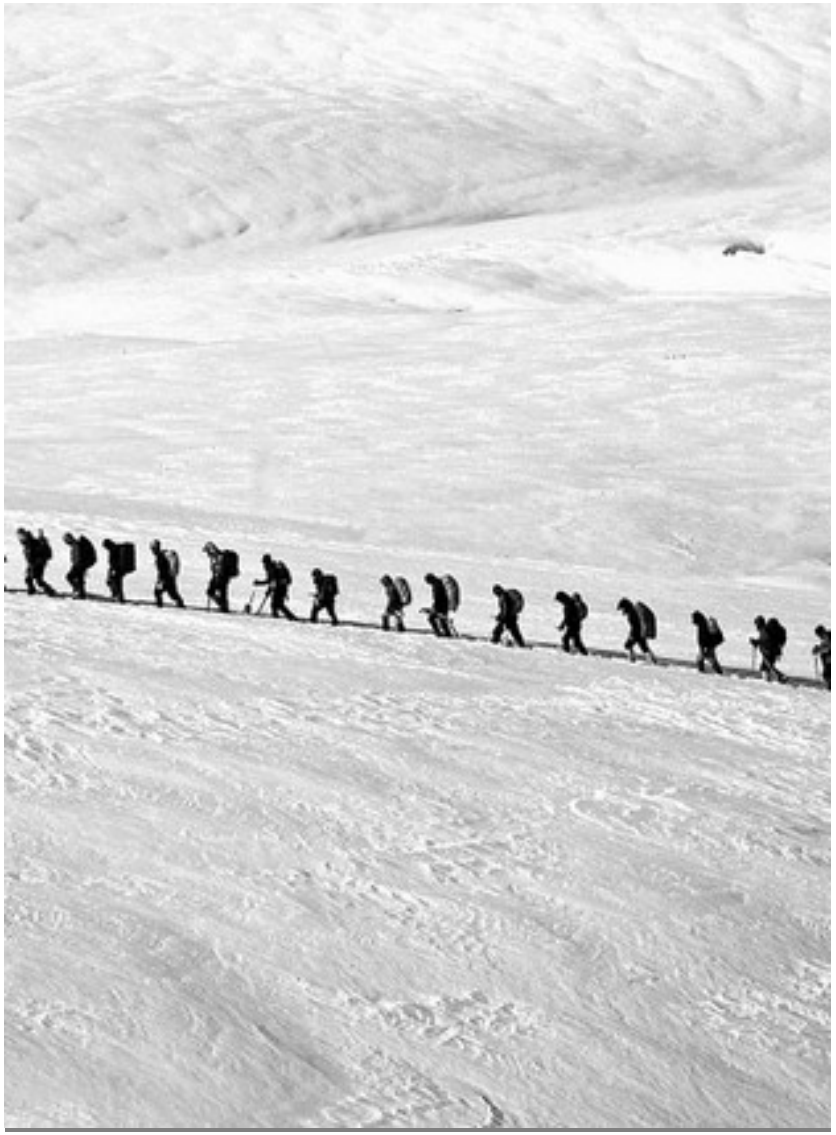
Fundamentos de Programación. Primer Curso de ASIR.

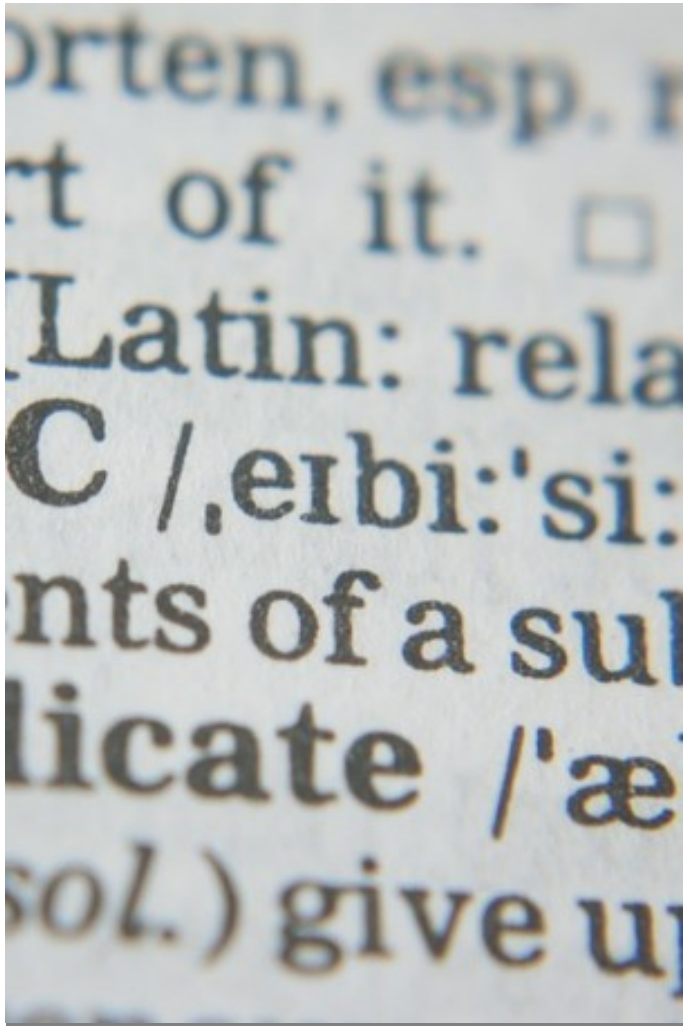
UT04.03 - Estructuras de datos. Diccionarios.

UT04.01 - Estructuras de datos. Diccionarios.
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: **listas, 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. Diccionarios.
2.- Introducción.
1.2.- Definición de diccionario. Características.



Definición.

- Un diccionario es como una lista, pero más general. **En un diccionario los índices pueden ser (casi) de cualquier tipo**.
- Se puede pensar en un diccionario como una asociación entre un conjunto de índices (claves) y valores. **Cada clave apunta a un valor**.
- La asociación de una clave y un valor es llamada par **clave-valor** o a veces elemento.

Caracteristicas.

- No tienen orden.
- Pueden contener elementos de distintos tipos.
- **Son mutables**. Pueden alterarse durante la ejecución de un programa.
- **Las claves son únicas**. No pueden repetirse en un mismo diccionario, y **pueden ser de cualquier tipo de datos inmutable**.

UT04.01 - Estructuras de datos. Diccionarios.
2.- Definición y acceso a los componentes de un diccionario.
2.1.- Definición de una variable diccionario.

- **Mediante la función dict().**

  diccionario = dict()

- **Mediante la definición de grupos clave:valor** separados por comas y encerrados entre llaves.

  dic = {clave1:valor1, ...}

- **Mediante la función dict() y tuplas** de valores separadas por comas.

  dic = {(clave1,valor1), …}

- **Mediante dict() y asignaciones de valores a claves separadas por comas**.

  dic = {clave1=valor1, …}

```
>>> diccionario
{}

>>> diccionario = {"H":1, "He":2, "Li":3}
>>> diccionario
{'H': 1, 'He': 2, 'Li': 3}

>>> diccionario = {("H",1), ("He",2), ("Li",3)}
>>> diccionario
{('Li', 3), ('He', 2), ('H', 1)}

>>> diccionario = dict(H=1, He=2, Li=3)
>>> diccionario
{'H': 1, 'He': 2, 'Li': 3}
```

UT04.01 - Estructuras de datos. Diccionarios.
2.- Definición y acceso a los componentes de un diccionario.
2.2.- Acceso a los elementos de un diccionario.

- Un elemento puede ser referenciado mediante su clave. La forma de hacerlo es indicando el nombre del diccionario y la clave entre corchetes.

  diccionario[clave]

- Los diccionarios tienen un método llamado **get**. Este toma una clave y un valor por defecto.

  Si la clave aparece en el diccionario, get regresa el valor correspondiente; en caso contrario, regresa el valor por defecto.

- En el ejemplo siguiente se usa un diccionario para contar el número de veces que aparecen las letras de una cadena.
- Otra forma es **usar el operador in**. **Con el método in podemos saber si una clave está presente en el diccionario**.

```
>>> diccionario = {"H":1, "He":2, "Li":3}
>>> diccionario["He"]
2

# Ejemplo de uso del método get().
cadena = "odontologo"
dic = dict()
for letra in cadena:
   dic[letra] = dic.get(letra,0) + 1
print(dic)

{'o': 5,'d': 1,'n': 1,'t': 1,'l': 1,'g': 1}

# Otra forma del método get().
cadena = "odontologo"
dic = dict()
for letra in cadena:
    if letra not in dic:
        dic[letra] = 1
    else:
        dic[letra] = dic[letra] + 1
print(dic)
```

UT04.01 - Estructuras de datos. Diccionarios.
2.- Definición y acceso a los componentes de un diccionario.
2.3.- Acceso a las claves y los valores de un diccionario (I).

- Se puede acceder a las claves de un diccionario mediante el método

  diccionario.keys()

  Este método devuelve un conjunto con las claves de los distintos elementos.
- Análogamente, se puede acceder a los valores de un diccionario con el método

  diccionario.values()

  Este método devuelve un conjunto con los valores de los distintos elementos.

```
>>> diccionario = {"H":1, "He":2, "Li":3}
>>> diccionario.keys()
dict_keys(['H', 'He', 'Li'])

>>> diccionario.values()
dict_values([1, 2, 3])
```

UT04.01 - Estructuras de datos. Diccionarios.
2.- Definición y acceso a los componentes de un diccionario.
2.3.- Acceso a las claves y los valores de un diccionario (II).

- Existe la posibilidad de obtener ambas cosas mediante el método

  diccionario.items()

  Este método devuelve un conjunto de tuplas (clave, valor) el cual puede ser iterable.
- Veamos un ejemplo de esto. Con el script de ejemplo vamos a mostrar por pantalla una tabla con el contenido del diccionario.

```
>>> elementos = {"H":1, "He":2, "Li":3}
>>> elementos.items()
dict_items([('H', 1), ('He', 2), ('Li', 3)])

# Tabla con el contenido del diccionario.
elementos = dict(H=1, He=2, Li=3)
for (elemento,numAtomico) in elementos.items():
    print(f"{elemento}\t{numAtomico:>3}")

H           1
He          2
Li          3
```

UT04.01 - Estructuras de datos. Diccionarios.
3.- Eliminar elementos de un diccionario.

- Para eliminar los elementos de un diccionario o el diccionario en si mismo se usa la función **del**.
- Si se desea eliminar una pareja clave:valor usaremos la expresión

  del diccionario[clave]
- Por el contrario si se desea eliminar el diccionario entero se usa la expresión

  del diccionario

  de esta forma **se elimina también la variable**.

```
>>> colores = {"Verde": 32, "Azul": 21}

>>> del colores["Azul"]
>>> colores
{'Verde': 32}

>>> del colores
>>> colores
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'colores' is not defined
```

UT04.01 - Estructuras de datos. Diccionarios.
4.- Número de elementos de un diccionario.

- Se puede obtener el número de elementos que tiene un diccionario mediante la función **len**.

  len(diccionario)

```
>>> colores = {"Verde": 32, "Azul": 21}

>>> len(diccionario)
>>> 2
```

UT04.01 - Estructuras de datos. Listas.
5.- Métodos de diccionarios (I).

- En ellos están los siguientes:

colores = {“Verde”: 32, “Azul”: 21 }		
Método	**Función**	**Ejemplo**
colores.clear()	Elimina todos los elementos del diccionario. Este método no elimina la variable.	colores.clear() {}
colores.pop(key, default)	Elimina el elemento dado por la clave key, devolviendo su valor. Si no se encuentra devuelve default.	colores.pop(“Azul”,0) 21 colores.pop(“Rojo”,0) 0
colores.popitem()	Elimina el elemento al azar.	
colores.update(dic)	Actualiza el contenido del diccionario. Se actualizan los valores y si una clave: valor no aparece se añade.	masColores {“Rojo”:25, “Verde”:10} colores.update(masColores) colores {“Verde”: 10, “Azul”: 21, “Rojo”: 25 }

- **Diccionarios en Python: https://docs.python.org/es/3/library/stdtypes.html#typesmapping**

UT04.01 - Estructuras de datos. Listas.
5.- Métodos de diccionarios (II).

- En ellos están los siguientes:

colores = {“Verde”: 32, “Azul”: 21 }		
Método	**Función**	**Ejemplo**
colores.copy()	Obtiene una copia del diccionario en otra variable.	otroDic = colores.copy() otroDic {“Verde”: 32, “Azul”: 21 }
colores.setdefault(key,v)	Si se omite el segundo parámetro funciona como el método get. En caso contrario, si no existe la clave inserta la clave y el valor.	colores.setdefaultkey(“Rojo”,52) colores {“Verde”: 32, “Azul”: 21, “Rojo”: 52 }

- **Diccionarios en Python: https://docs.python.org/es/3/library/stdtypes.html#typesmapping**