Fundamentos Python 3 – Estructuras de datos

Las herramientas del científico de datos

Juan Manuel Moreno – jmmoreno@profesorescol.imf.com

IMF
Smart Education

ÍNDICE

IMF
Smart Education

01 Objetivos unidad 1

IMF
Smart Education

1.- Objetivos Unidad 1

- **Conocer los principales fundamentos de Python.**
- **Saber instalar Jupyter Notebook, la herramienta que vamos a utilizar para trabajar con Python.**
- **Realizar desarrollos básicos en Python a través de Jupyter Notebook**
- **Saber cómo declarar, procesar y distinguir distintos tipos de variables.**
- **Conocer y manejar las sentencias condicionales If – Else.**
- **Conocer el funcionamiento de los bucles en Python, for y while.**
- **Trabajar con las principales estructuras de datos en Python: Tuplas, listas y diccionarios de datos**.
- Comprender cómo modularizar los programas a través de funciones.
- Resolver problemas de diferente dificultad.

02

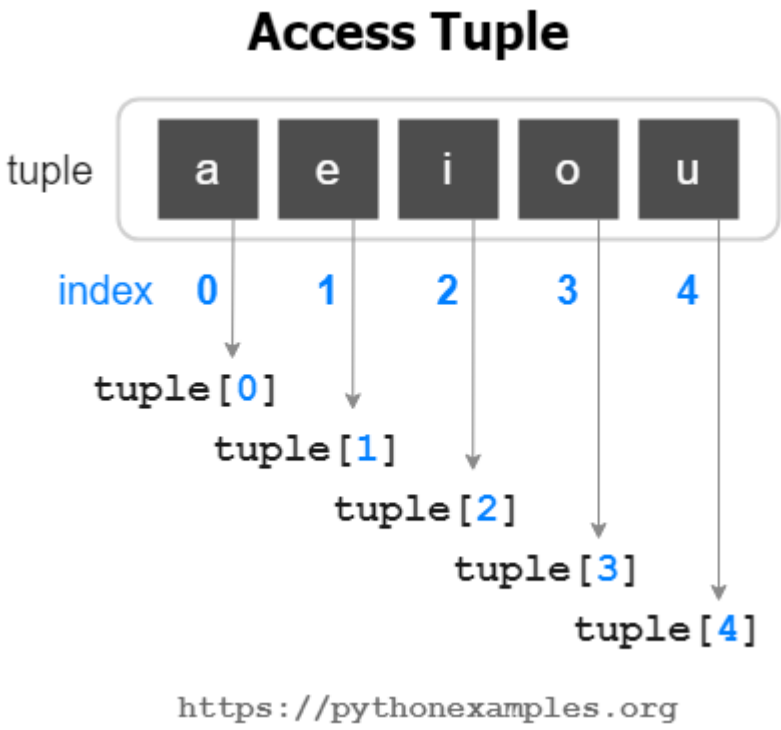
Tuplas

IMF
Smart Education

2.- Tuplas

2.1.- Tuplas

- Estructura de datos más básica.
- La información es almacenada entre paréntesis y separados por comas
- Su representación es la siguiente:
  - (‘a’, ‘e’, ‘i’, ‘o’, ‘u’)
- Puede haber tuplas anidadas
  - (‘a’, ‘e’, ‘i’, ‘o’, ‘u’, (1, 2, 3, 4))
- No es posible modificar los valores de una tupla ya que **no son mutables**.
- Para acceder a sus valores realizaremos **indexación**.
- Para indexar siempre utilizaremos esta nomenclatura **t[n_indice]**
- La clase de una tupla se designa como `tuple`



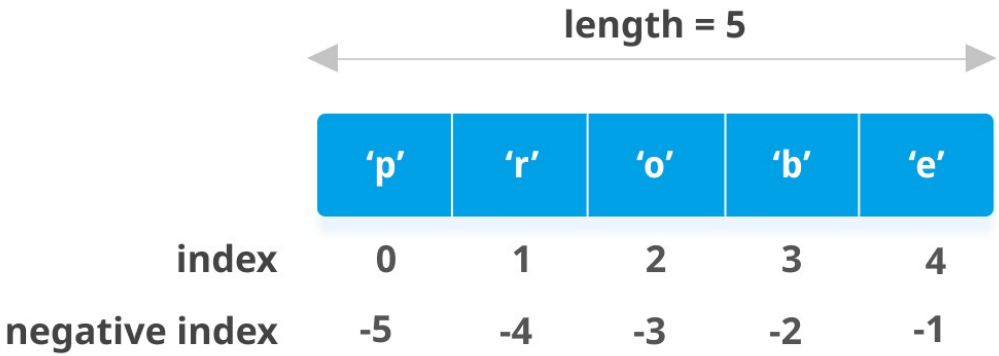
03 Listas

IMF Smart Education

3.- Listas

3.1.- Listas - Características

- Similar a una tupla, pero sus valores son mutables.
- También acepta valores de cualquier tipo.
- Los datos de una tupla van entre paréntesis, los de una l entre corchetes [ ]
- Los valores se modifican mediante posición del índice.
- Una de las estructuras de datos más empleadas en ciencia de datos.
- Tienen la clase list



3.2.- Listas - Funciones

- Algunas funciones de las listas
  - append: Añadir un elemento en una lista.
  - clear: Borra por completo los elementos de una lista.
  - copy: Realiza una copia de una lista (una asignación de una lista sin copy provoca que ambas compartan posiciones de memoria RAM, por lo que una alteración en la lista nueva, provocará que también ocurra en la lista original si no empleamos copy)
  - extend: Añade elementos de una lista al final de otra.
  - index: Devuelve la posición del índice de un valor.
  - insert: Añade un elemento en una posición específica.
  - pop: Borra un elemento en una posición específica.
  - remove: Borra el elemento a través de un valor específico.
  - reverse: Invierte el orden de la lista.
  - sort: Ordena los valores de una lista.

3.3.- Listas – Recorrer listas

- La forma más habitual de recorrer listas es a través de un bucle for:
  - Directamente sobre los **ítems** de la lista: `for ítem in lista:`
  - Recorriendo los **índices** de la lista: `for ítem in range(len(lista)):`

```
1 for item in lista:
2     print(item)
```

```
0
1
3
4
9
85
5000
```

```
1 for item in range(len(lista)):
2     print('Índice --> ', item , ' valor: ', lista[item])
```

```
Índice -->  0  valor:  0
Índice -->  1  valor:  1
Índice -->  2  valor:  3
Índice -->  3  valor:  4
Índice -->  4  valor:  9
Índice -->  5  valor:  85
Índice -->  6  valor:  5000
```

04

Diccionarios de datos

IMF
Smart Education

4.- Diccionarios de datos

4.1 – Diccionarios de datos – Recorrer listas

- Estructura de datos de rápido procesamiento.
- A diferencia de las listas y tuplas, los diccionarios de datos no tienen índice.
- Su estructura está compuesta por pares de elementos
  - `{ clave : valor}`
- En este caso, el campo clave sería similar a un índice ya que es único para cada valor de un diccionario de datos.
- La clase de un diccionario es `dict`

Seguimiento práctico del contenido

A partir de aquí, vamos a ver cómo funcionan las estructuras de datos en Python

1_4_Estructuras_datos.ipynb

IMF

Smart Education