PYTHON PARA LINGÜISTAS

STRINGS, LISTAS, DICCIONARIOS

ALEJANDRO ARIZA

CENTRE DE LLENGUATGE I COMPUTACIÓ

UNIVERSITAT DE BARCELONA

TIPO DE DATO: STRING

- Un string es una secuencia de caracteres delimitada por comillas (“” o ‘’).
- Un string puede representar un carácter, un morfema, una palabra, una frase, un párrafo o un libro entero.
- Un string no tiene una estructura compleja ni una “unidad” de texto en concreto.
- Las dos operaciones más básicas que podemos hacer con Strings son concatenación y segmentación.
- Los Strings son muy útiles, pero a menudo no son suficientes para crear un programa complejo.

TIPO DE DATO: LISTA

- Una lista es una secuencia de datos de longitud arbitraria.
- A diferencia de los strings, los elementos de una lista pueden tener un valor “significativo”:

  Comparemos: “Esto es una frase” → Traducido a lista, sus elementos serían → “E”, “s”, “t”, “o”...

  De igual forma, podemos encontrarnos la siguiente lista: [“Esto”, ”es”, “una”, “frase”] → elementos “Esto”, “es”, “una”, “frase”
- Las listas son mucho más útiles a la hora de PROCESAR/ANALIZAR texto.
- Nosotros, como programadores, decidimos la unidad de texto que almacenaremos en cada posición de la lista (caracteres, palabras, frases, etc).
- Las listas pueden contener datos de distinto tipo [“Esto”, “es”, 1, “frase”] – integer y string.
- Las listas tienen un orden – normalmente los nuevos elementos se añaden al final de la lista.

TRABAJANDO CON LISTAS

- Las listas son más complejas que los strings, pero pueden ser herramientas más potentes y útiles también.

`mi_lista = []` – Creamos una lista vacía

`mi_lista = list()` – Otra forma de crear una lista vacía es usando el constructor

`mi_lista = [“un”, “texto”]` – Creamos una lista con dos elementos

`mi_lista_1 + mi_lista_2` – Concatenamos dos listas

`mi_lista[1]` – Accedemos al segundo elemento de la lista (“texto”)

`mi_lista[1:3]` – Accedemos a los elementos que van de la segunda posición a la cuarta, excluyendo al cuarto

TRABAJANDO CON LISTAS (2)

- La lista es un objeto que tiene una serie de métodos implementados:

`mi_lista.append("aquí")`	– Añade un nuevo elemento al final de la lista
`mi_lista.extend(["aquí", "ahora"])`	– Concatena la nueva lista al final de mi_lista
`mi_lista.remove("aquí")`	– Borra el primer elemento "aquí" de la lista
`mi_lista.index("aquí")`	– Devuelve la posición del primer elemento con valor "aquí"

EL OPERADOR “IN”

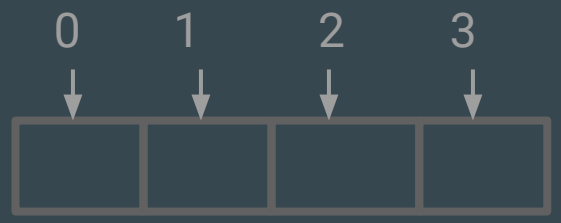
- El operador “in” se utiliza en la mayoría de los programas Python.
- El operador “in” comprueba la existencia de ciertos valores dentro de strings, listas o diccionarios.

  “Algo” in mi_lista – True if mi_lista tiene algún elemento “Algo”
- Recordad, el operador “in” es también parte del bucle “for”

  for mi_elemento in mi_lista: – hace algo por cada elemento en mi_lista

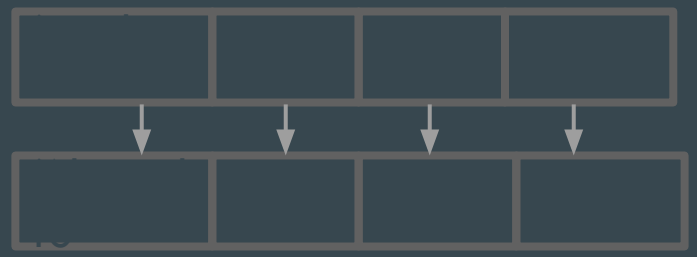
LIMITACIONES DE LAS LISTAS

- El tipo de dato lista, a pesar de ser más potente que el string, tiene ciertas limitaciones.
- El tipo de dato string contiene únicamente caracteres como sus “elementos”
- El tipo de dato lista puede contener elementos de varios tipos
- Una importante limitación de la lista es que sus elementos sólo pueden ser accedidos por su posición en la lista:



TIPO DE DATO: DICCIONARIO

- El tipo de dato diccionario es similar a la lista, a diferencia de que podemos acceder a cada elemento utilizando una etiqueta (no necesariamente numérica) en vez de por la posición que ocupa en la lista:



- La etiqueta que le damos a cada valor recibe el nombre de clave. En el diccionario cada elemento pasa a ser un par de clave-valor:

  mi_perfil[“nombre”] – “Alejandro”)

WORKING WITH DICTIONARIES

- Trabajar con diccionarios es bastante directo

  `mi_diccionario = {}` – Creates new empty dictionary

  `mi_diccionario[“nombre”] = “Alejandro”` – Asigna el valor “Alejandro” para la clave “nombre”

  `print(mi_diccionario[“edad”])` – Si el diccionario contiene la clave “edad”, la imprimirá

  Si la clave no existe, devolverá un error KeyNotFound

  `mi_diccionario_1 + mi_diccionario_2` – Combinar dos diccionarios en uno

- Las claves dentro de un diccionarios tienen que ser únicas. No pueden existir dos entradas con la misma clave.

STRINGS, LISTAS, DICCIONARIOS

- Strings, listas y diccionarios serán a menudo usados a lo largo del curso
- Es importante que os sintáis cómodos con estos tipos de datos y entendáis cuando conviene utilizar un tipo u otro.
- Principio de economía: “en igualdad de condiciones, la explicación más sencilla suele ser la más probable”
- Conoced vuestros datos y vuestros objetivos antes de elegir el tipo de datos o construir vuestro algoritmo.

CHULETA DE PYTHON

- La documentación oficial de Python la podéis encontrar en:

https://docs.python.org/3/

- Para pistas o recordatorios de conceptos en Python, podéis utilizar:

https://www.pythoncheatsheet.org/

¡GRACIAS!
¿PREGUNTAS?