think python

cadenas

- `<string>.capitalize()`
- `<string>.isupper()`
- `list(cadena)` : separa en caracteres
- `<string>.split()` : separa en palabras
    - se puede pasar por parámetro el delimitador
- `<delimitador>.join(<lista>)`: junta los elementos de la lista, poniendo en medio el delimitador (!!)

list

- por referencia (aliasing, más de una referencia al mismo objeto)
- OJO al usar **funciones** o **métodos** al trabajar con listas, (referencia VS new lista) : página 96
- `empty = []`, `cadenas = ['uno','dos','tres','cuatro','cinco']`, `numeros = [1,2,3]`, `mixta = ['uno',2,['tres',3],'cuatro']`
- mutables : `cadenas[1] = 'DOS'`
- búquedas: `'tres' in cadenas` → true
- recorridos:
    - `for i in numeros:`
    - `for i in range(len(mixta)):`
- concatenación: `otra_mixta = cadenas + numeros` → ['uno','dos','tres','cuatro','cinco',1,2,3]
- multiplicativo: `numeros * 2` → [1,2,3,1,2,3]
- slices:
    - de los índices pasados, incluye el primero, excluye el segundo.
    - `cadenas[1:]` → ['dos','tres','cuatro','cinco']
    - `cadenas[:1]` → ['uno','dos']
    - `cadenas[2:4]` → ['tres','cuatro']
    - asignación:
        - `cadenas[2:3]=['aaa','bbb']` → ['uno','aaa','bbb','cuatro','cinco']
- métodos:
    - `<list>.append()` : añade elemento al final
    - `<list>.extend(<list>)` : añade al final de la lista otra lista, cambia la primera.
    - `<list>.sort()` : ordena
    - `<list>.pop(#elemento)` : extrae elemento de la lista (lo devuelve) en función del índice del mismo
        - `del <list>[#elemento]` : borra sin devolverlo en función del índice del mismo. permite **slices**
        - `<list>.remove(elemento)` : borra el elemento si sabemos cual es
    - `print list('cadena')` : ['c','a','d','e','n','a']

diccionarios

- pareja clave-valor
- no mantiene el orden de entrada dentro del diccionario
- `empty = dict()`
- `diccio = {}`
- asignación: `diccio[<clave>] = 'valor'`
- **len**: `len(<diccionario>)`

- **get** : recupera un valor de una clave y permite establecer un valor por defecto (si no existe la clave)
  - `<diccionario>.get('<clave>',<default>)`
- **keys** : devuelve como lista las keys de un diccionario
- **in** :
  - busca en las claves, devuelve cierto/falso → `'<clave>' in diccio`
  - con el uso de `<diccionario>.values()` se pueden hacer búsquedas en los valores
  - búsqueda por hashtable
  - **not in**
- INVERSE pagína 127 → 'inverse[val] = [key] ????

  == Tuplas

  == otros * raise'' : provoca/lanza una excepción

- variables globales:
  - para ser manipuladas en una función, se han de declarar previamente **global <var>** si son inmutables
  - si son mutables (listas, diccionarios), se puede añadir, borrar, modificar sin problema. Solo se tendrían que declarar en caso de reasignación