Programación Avanzada
IIC2233 2024-2

Hernán Valdivieso - Daniela Concha - Francisca Ibarra - Lucas Van Sint Jan - Francisca Cattan

Repaso

Estructuras de datos

- Forma especializada de agrupar datos.
- Almacenamiento, acceso y utilización eficiente.
- ¿Qué estructura es mejor para cada caso?

Lo básico

Estructuras secuenciales

- Orden secuencial de elementos.
- Garantizan un recorrido ordenado y eficiente de los elementos.
- Algunas de estas estructuras son:
    - Tuplas
    - *Named tuples*
    - Listas (Arreglos)
    - *Stacks*
    - Colas

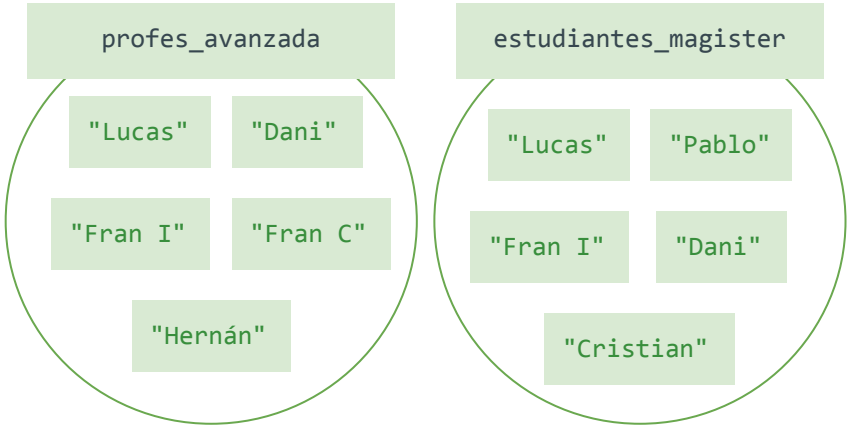
Estructuras no secuenciales

- No hay orden entre elementos.
- Búsqueda de elementos muy eficiente.
- Algunas de estas estructuras son:
    - *Sets* (Conjuntos)
    - Diccionarios
    - *Defaultdict*

Sets (Conjuntos)

- No hay orden entre elementos.
- Búsqueda de elemento específico muy eficiente.
- No permite duplicados.
- Solo permite elementos inmutables.

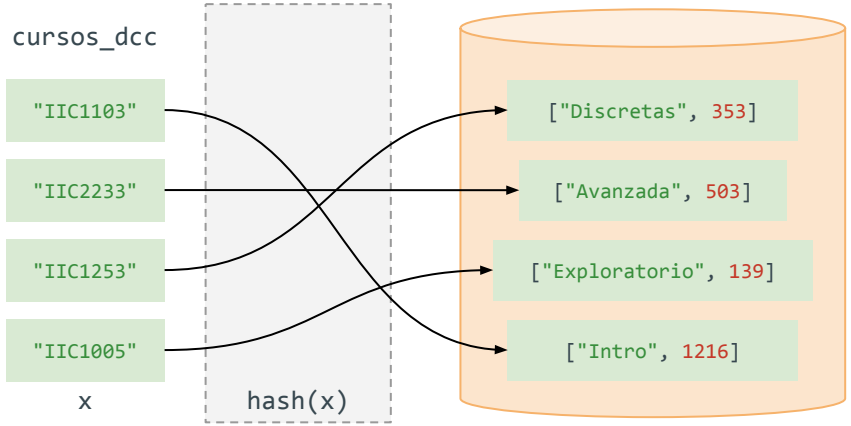
```
estudiantes_magister = {"Cristian", "Lucas",
                        "Fran I", "Pablo", "Dani"}

profes_avanzada = set(profesores)

print("Cristian" in profes_avanzada)
print(profes_avanzada & estudiantes_magister)
```



Diccionarios

- Almacena pares: llave-valor.
- Búsqueda de llave específica muy eficiente.
- Llaves no pueden estar duplicadas.
- Solo permite elementos inmutables como llaves.
- Valor puede ser cualquier elemento.

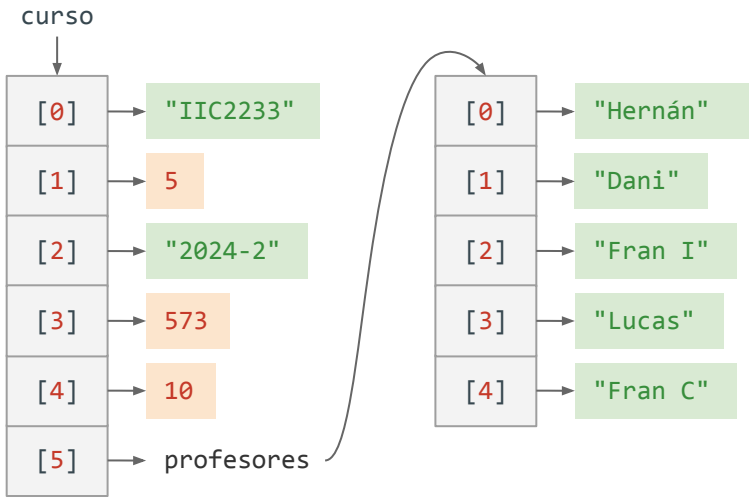
```
cursos_dcc = { "IIC1103": ["Intro", 1216],
               "IIC2233": ["Avanzada", 528],
               "IIC1253": ["Discretas", 353],
               "IIC1005": ["Exploratorio", 139]}

print(cursos_dcc["IIC2233"])
print(cursos_dcc["IIC1103"][1])
```



Tuplas

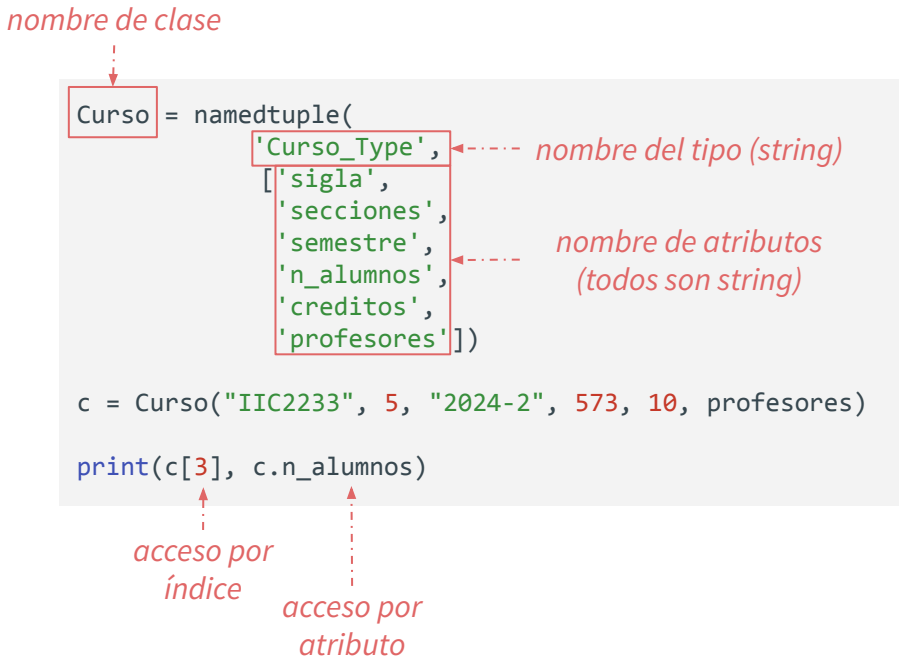
- Orden secuencial de elementos.
- Búsqueda de i-ésimo elemento muy eficiente.
- Immutable.
- Suelen utilizarse para agrupar elementos heterogéneos.

```
curso = ("IIC2233", 5, "2024-2", 573, 10, profesores)
print(curso[3])
```



Named tuples

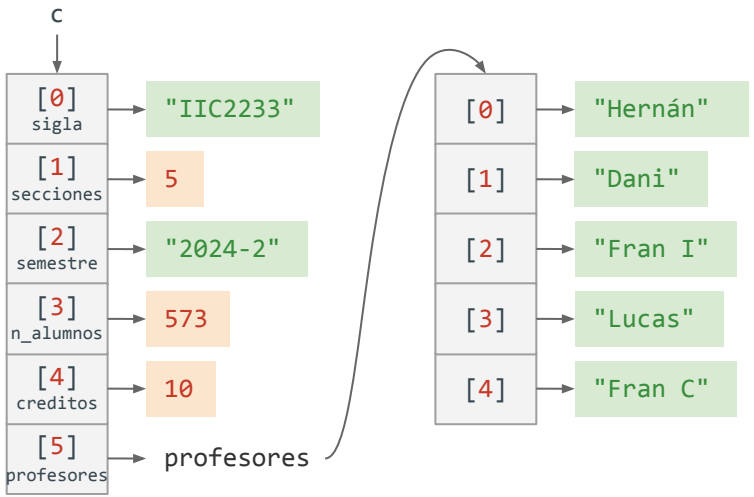
- Orden secuencial de elementos.
- Búsqueda de i-ésimo elemento muy eficiente.
- Immutable.
- **Cada posición tiene un nombre (atributo).**



Named tuples

- Orden secuencial de elementos.
- Búsqueda de i-ésimo elemento muy eficiente.
- Immutable.
- **Cada posición tiene un nombre (atributo).**

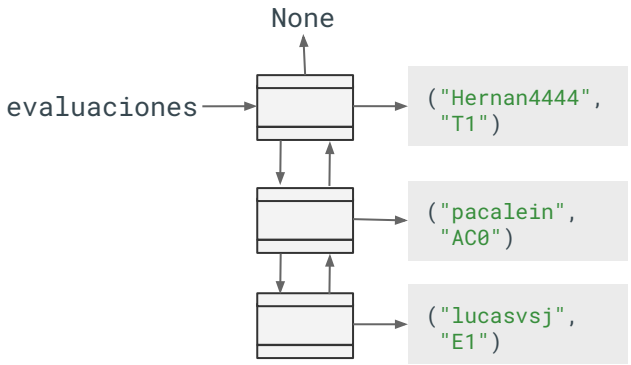
```
c = Curso("IIC2233", 5, "2024-2", 573, 10, profesores)
print(c[3], c.n_alumnos)
```



Colas

- Orden secuencial de elementos.
- Inserción/eliminación eficiente en el extremo de la cola.
- Mutable
- Las *deque* de Python son eficientes en ambos extremos.

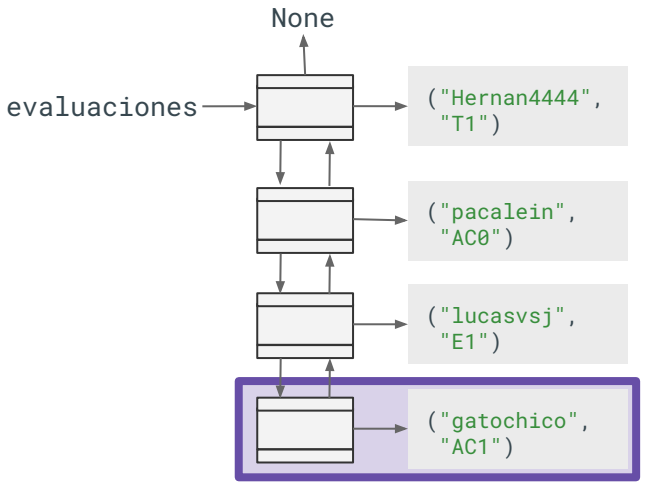
```
from collections import deque

evaluaciones = deque([("Hernan4444", "T1"),
                      ("pacalein", "AC0"),
                      ("lucasvsj", "E1")])
```



Colas

- Orden secuencial de elementos.
- Inserción/eliminación eficiente en el extremo de la cola.
- Mutable
- Las *deque* de Python son eficientes en ambos extremos.

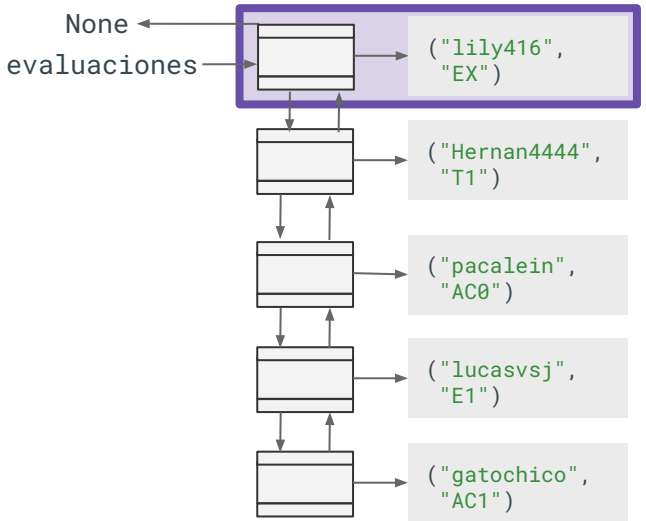
```
from collections import deque

evaluaciones = deque([("Hernan4444", "T1"),
                      ("pacalein", "AC0"),
                      ("lucasvsj", "E1")])
evaluaciones.append(("gatochico", "AC1"))
```



Colas

- Orden secuencial de elementos.
- Inserción/eliminación eficiente en el extremo de la cola.
- Mutable
- Las *deque* de Python son eficientes en ambos extremos.

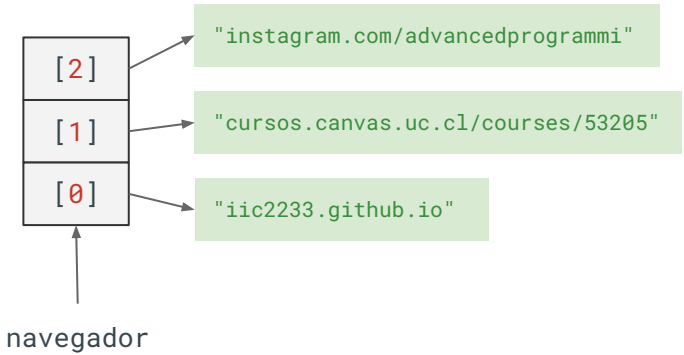
```
from collections import deque

evaluaciones = deque([("Hernan4444", "T1"),
                      ("pacalein", "AC0"),
                      ("lucasvsj", "E1")])
evaluaciones.append(("gatochico", "AC1"))
evaluaciones.appendleft(("lily416", "EX"))
```



Stack

- Orden secuencial de elementos.
- Búsqueda de i-ésimo elemento muy eficiente.
- Mutable.
- Inserción y eliminación al final.

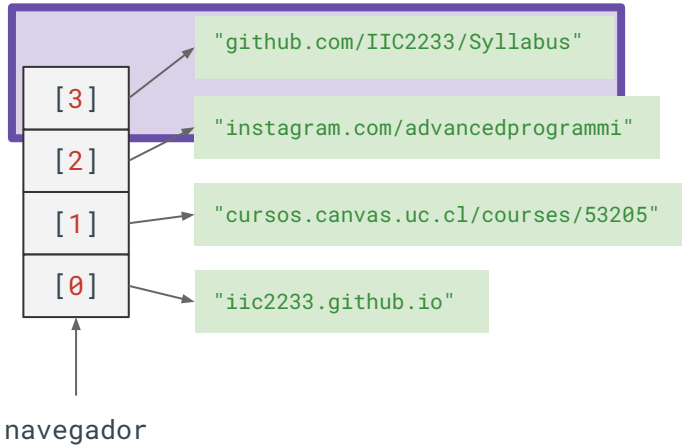
```
navegador = [
    "iic2233.github.io",
    "cursos.canvas.uc.cl/courses/53205",
    "instagram.com/advancedprogrammi"
]
```



Stack

- Orden secuencial de elementos.
- Búsqueda de i-ésimo elemento muy eficiente.
- Mutable.
- Inserción y eliminación al final.

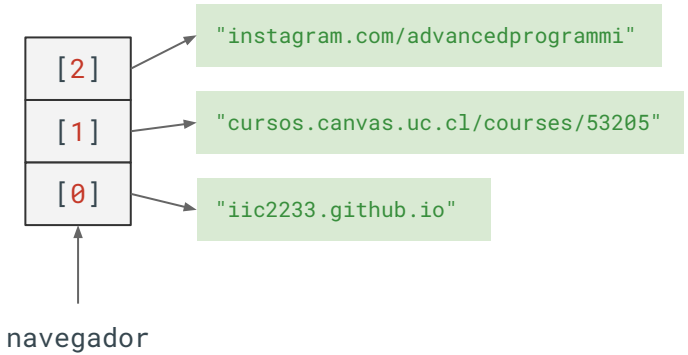
```
navegador = [
    "iic2233.github.io",
    "cursos.canvas.uc.cl/courses/53205",
    "instagram.com/advancedprogrammi"
]
navegador.append("github.com/IIC2233/Syllabus")
```



Stack

- Orden secuencial de elementos.
- Búsqueda de i-ésimo elemento muy eficiente.
- Mutable.
- Inserción y eliminación al final.

```
navegador = [
    "iic2233.github.io",
    "cursos.canvas.uc.cl/courses/53205",
    "instagram.com/advancedprogrammi"
]
navegador.append("github.com/IIC2233/Syllabus")
navegador.pop() # Elimina el último
```



Estructuras secuenciales

Estructura	Mutable	*Hasheable*	
Lista	✔	✘	Permiten agregar, eliminar, modificar elementos.
Tupla	✘	✔*	Útiles para retornar múltiples valores y contener valores que no cambian.
Named Tuple	✘	✔*	Tuplas donde se puede acceder a cada posición mediante un nombre.
Colas	✔	✘	Eficientes para insertar o retirar desde sus extremos.
Stacks	✔	✘	Inserción y eliminación son siempre en el tope del *stack.*

Resumen

Estructura	Insertar	Búsqueda por índice	Búsqueda por llave	Búsqueda por valor
Lista	✔	✔✔✔	✘	✔
Tupla	✘	✔✔✔	✘	✔
Diccionario	✔✔✔	✘	✔✔✔	✔
Set (Conjunto)	✔✔✔	✘	✔✔✔	✘

✔✔✔ Se puede y es eficiente.

✔ Se puede pero no siempre es eficiente.

✘ No se puede.

Operador * y **

- Desempaquetamiento
- *args
- **kwargs

Desempaquetamiento

- Nos permite extraer ciertos elementos de una secuencia (iterable) y almacenarlos en una variable.
- Realizable con el operador asterisco (*****).
- Es posible desempaquetar tanto una estructura secuencial (*****) como una de llave valor (******).

```
puntos = [i for i in range(10)]
x, ys, z =  puntos[0], puntos[1:-1], puntos[-1]
x, *ys, z = puntos
[x, *ys, z] = puntos

# Resultados
# 0 [1, 2, 3, 4, 5, 6, 7, 8] 9
# 0 [1, 2, 3, 4, 5, 6, 7, 8] 9
# 0 [1, 2, 3, 4, 5, 6, 7, 8] 9
```

Todas esas expresiones son equivalentes

*args

- Argumentos **posicionales.**
- Sirve para establecer que existe una cantidad **variable** de argumentos en una función (o método) sin poder tener referencia a cada uno.

```
def funcion_por_partes(*args):
    if len(args) < 3:
        return sum(args)
    else:
        mod_args = [pow(arg, 2) for arg in args]
        return max(mod_args)

funcion_por_partes(1, 5, 7, 9, 4, 3)
```

**kwargs

- Argumentos **por palabra clave.**
- Sirve para establecer que existe una cantidad **variable** de argumentos en una función (o método) manteniendo la referencia a cada uno.

```
def printer_simple(**kwargs):
    if 'parametro_0' in kwargs.keys():
        print('Parámetro inávalido')
    elif 'name' in kwargs.keys() and 'age' in kwargs.keys():
        print(f'Nombre: {kwargs["name"]}. '
              f'Edad: {kwargs["age"]}')
    else:
        print('No has entregado los argumentos esperados')

printer_simple(parametro_0='Persona', name='Pedro', age=26)
```

*args VS **kwargs

- Estos nombres son solo una **sugerencia** (convención), no es estrictamente necesario que se llamen así, es decir, las 3 siguientes formas son equivalentes:

```
def(*args, **kwargs):
    pass
```

```
def(*a, **b):
    pass
```

```
def(*el, **barto):
    pass
```

- Debe existir un orden secuencial en la definición de ellos, **primero** los **posicionales** (*args) **luego** los de **palabra clave** (**kwargs).

Veamos una pregunta de Evaluación Escrita

Tema: Estructuras de datos (Midterm 2023-2)

7. ¿Cuál(es) de la(s) siguiente(s) afirmación(es) es/son correctas respecto a las tuplas?
   - I. Son estructuras de datos mutables.
   - II. Pueden desempaquetarse en variables independientes.
   - III. Se puede hacer slicing sobre una tupla.
   - IV. Se puede acceder a sus elementos a través de índices.

   A) Solo I
   B) Solo III
   C) II y III
   D) II, III, IV
   E) I, II, III y IV

Veamos una pregunta de Evaluación Escrita

Tema: Estructuras de datos (Midterm 2023-2)

7. ¿Cuál(es) de la(s) siguiente(s) afirmación(es) es/son correctas respecto a las tuplas?
   - I. Son estructuras de datos mutables.
   - II. Pueden desempaquetarse en variables independientes.
   - III. Se puede hacer slicing sobre una tupla.
   - IV. Se puede acceder a sus elementos a través de índices.

   A) Solo I
   B) Solo III
   C) II y III
   D) II, III, IV
   E) I, II, III y IV

Actividad

Comentarios AC1

Programación Avanzada
IIC2233 2024-2

Hernán Valdivieso - Daniela Concha - Francisca Ibarra - Lucas Van Sint Jan - Francisca Cattan