Anuncios

Jueves 29 de agosto 2019

1. Encuesta de Carga Académica. ¡Respóndanla!

2. Hoy tenemos actividad evaluada. Deben tener acceso a su repositorio y saber hacer push.

Estructuras de datos 101

Semana 03 - Jueves 29 de agosto 2019

Estructuras de datos

- Forma especializada de agrupar datos
- Almacenamiento, acceso y utilización eficiente

Estructuras secuenciales

- Listas
- Tuplas
- *Named Tuples*

Estructuras secuenciales

- Ordenamiento secuencial
- Buenas para acceder al elemento *i*-ésimo

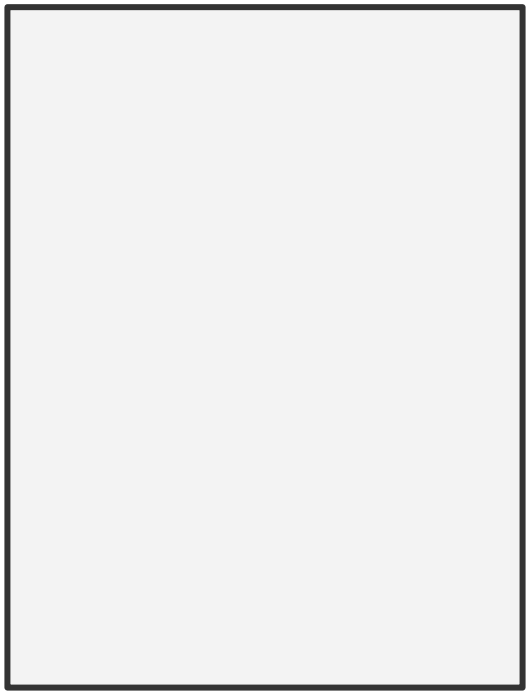
Estructura	Inmutable	*Hasheable*	Comentarios
Lista	✘	✘	Permiten agregar, eliminar, modificar elementos.
Tupla	✔	✔*	Sirven como llaves de diccionarios y para retornar múltiples valores.
Named Tuple	✔	✔*	Se puede acceder a cada posición mediante un nombre.

Tuplas *versus* listas

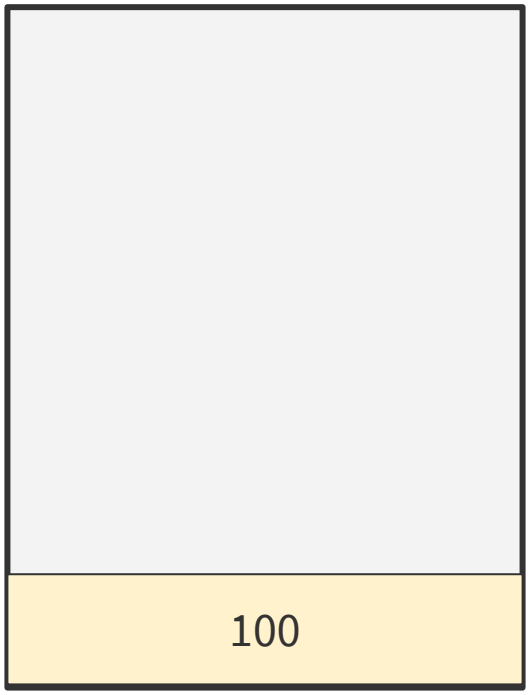
- **Listas:** almacenan una colección ordenada (homogénea)
- **Tuplas:** almacenan estructura (heterogénea)

Stacks

Stacks

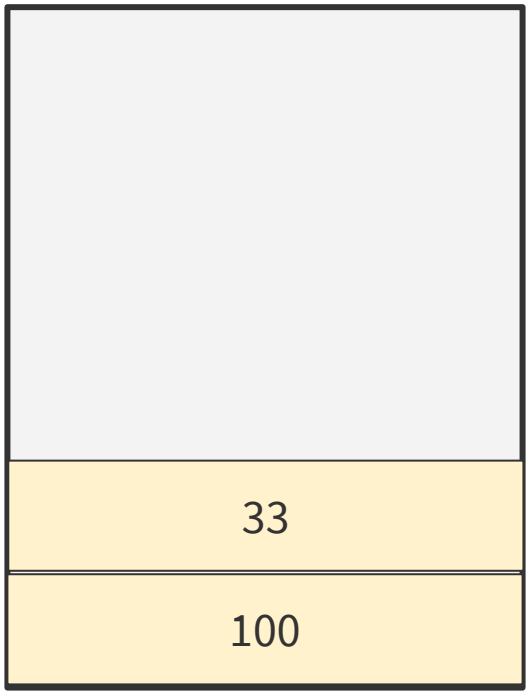


Stacks



¿Qué operación vimos?

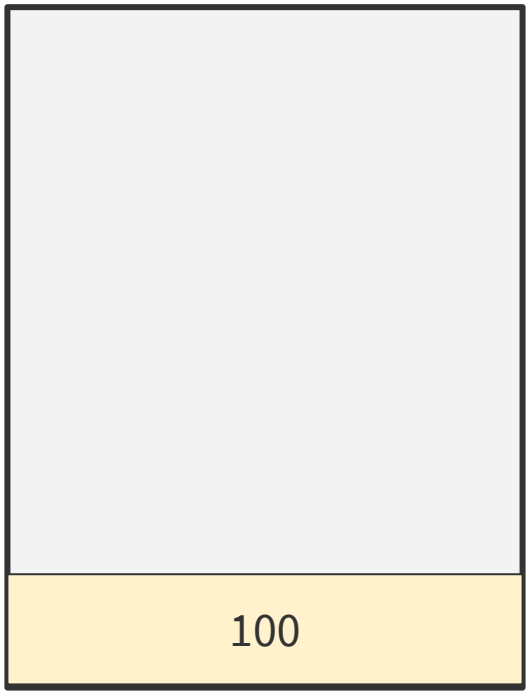
Stacks



push

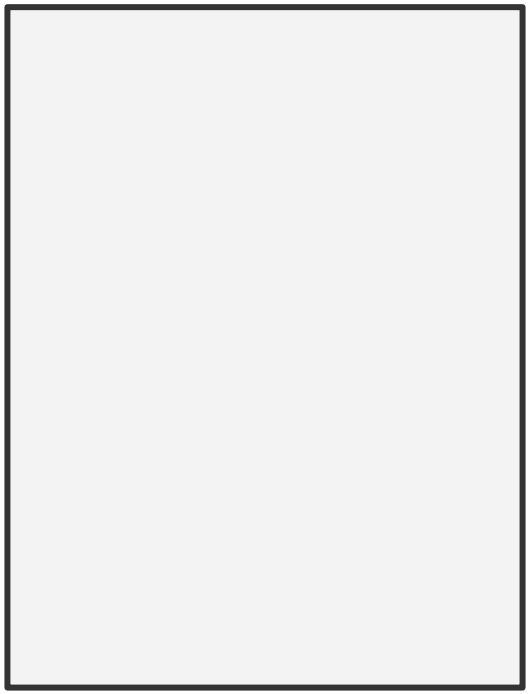
¿Qué es *peek*?

Stacks



¿Qué operación vimos?

Stacks

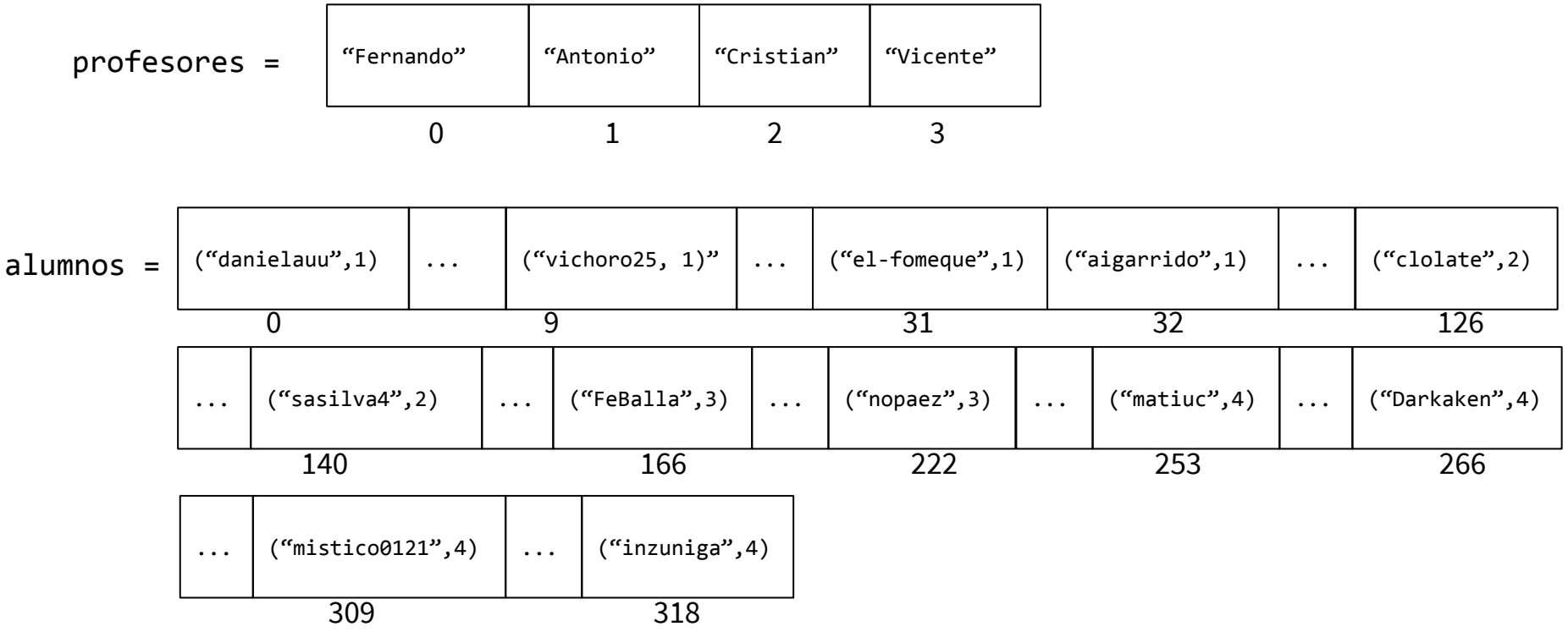


pop

Colas

- Tal y como la cola del **Quick** Deli

No da lo mismo qué estructura usar



¿Hay algún profesor Juan?

¿Hay algún alumno nopaez?

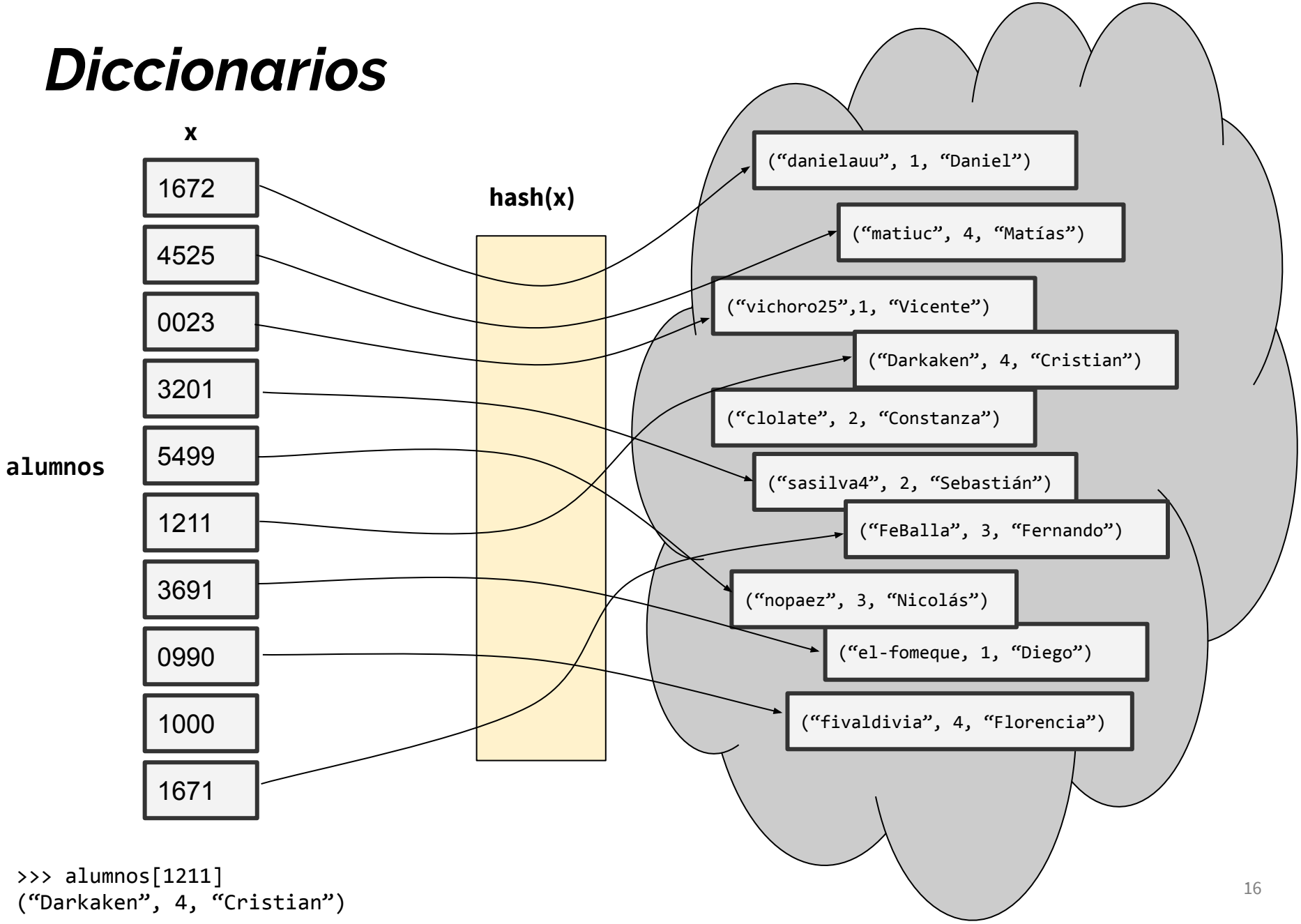
¿Cuántos hay en cada sección?

¿Dónde agrego alguien a la secc 2?

Diccionarios

- Estructura de mapeo. Llave -> valor
- ¿Qué dato puede ser una llave?
- Una llave está asociada a un único valor

Diccionarios

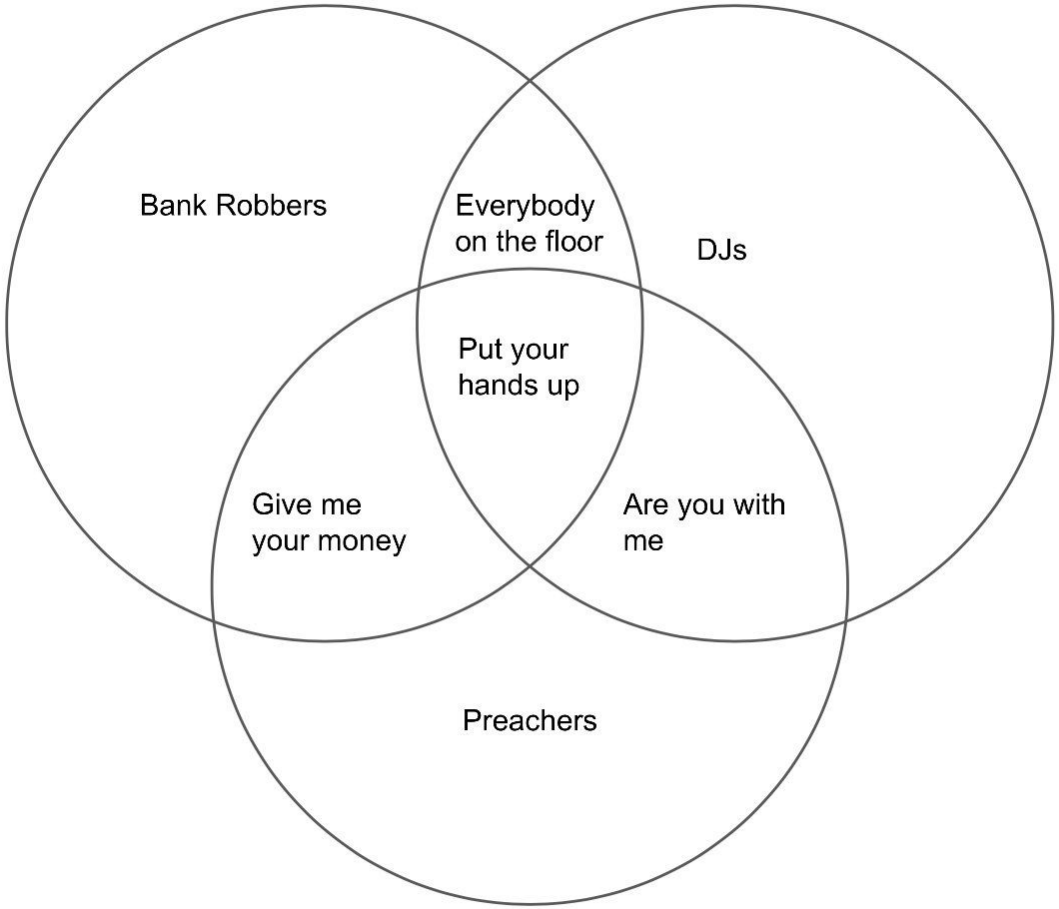



```
>>> alumnos[1211]
("Darkaken", 4, "Cristian")
```

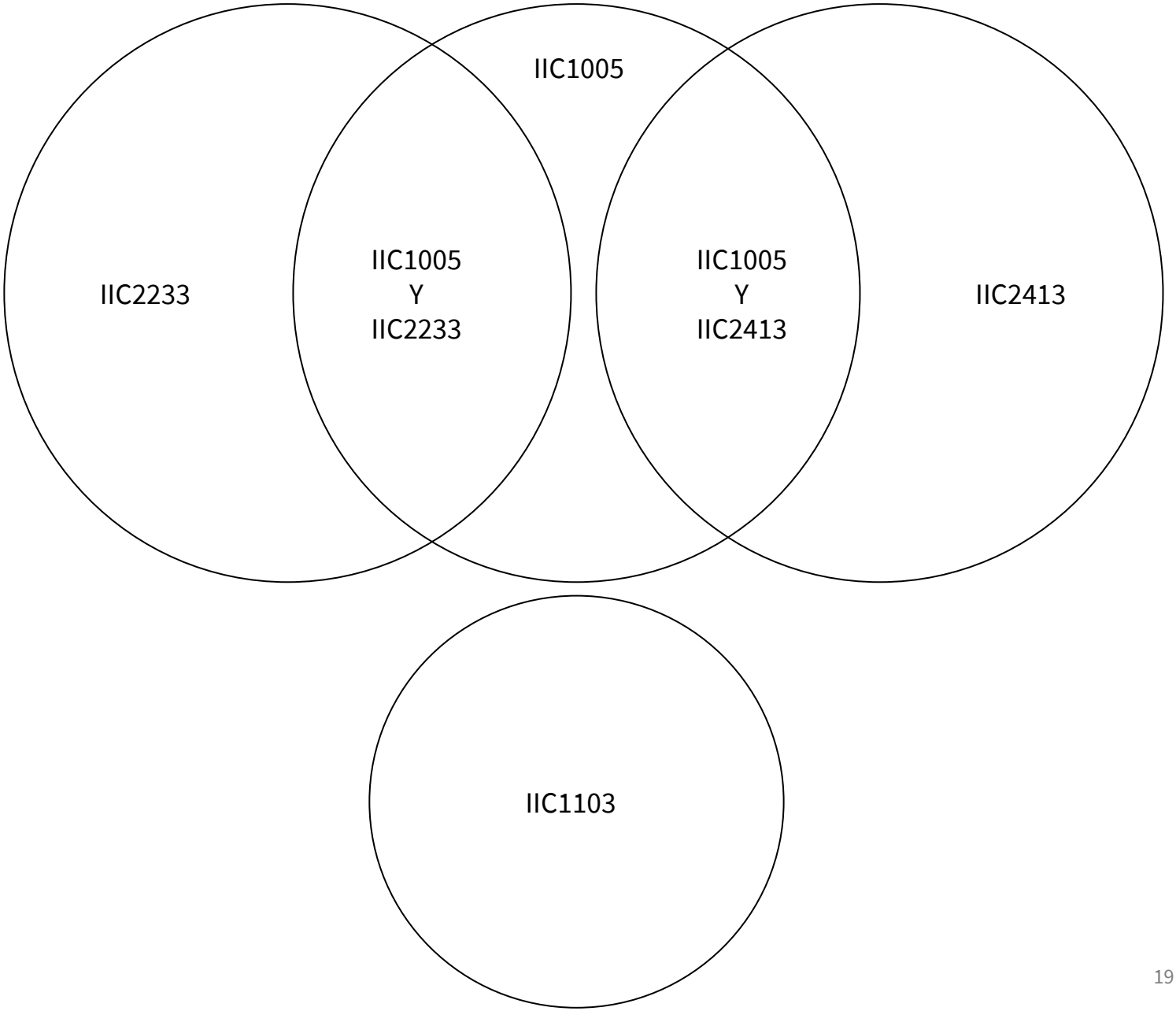
Sets

- ¡Conjuntos!
- Elementos sin ordenamiento
- Elementos únicos

Conjuntos


Bank Robbers
Everybody on the floor
DJs
Put your hands up
Give me your money
Are you with me
Preachers

Alumnos Profesor Vicente



Resumen

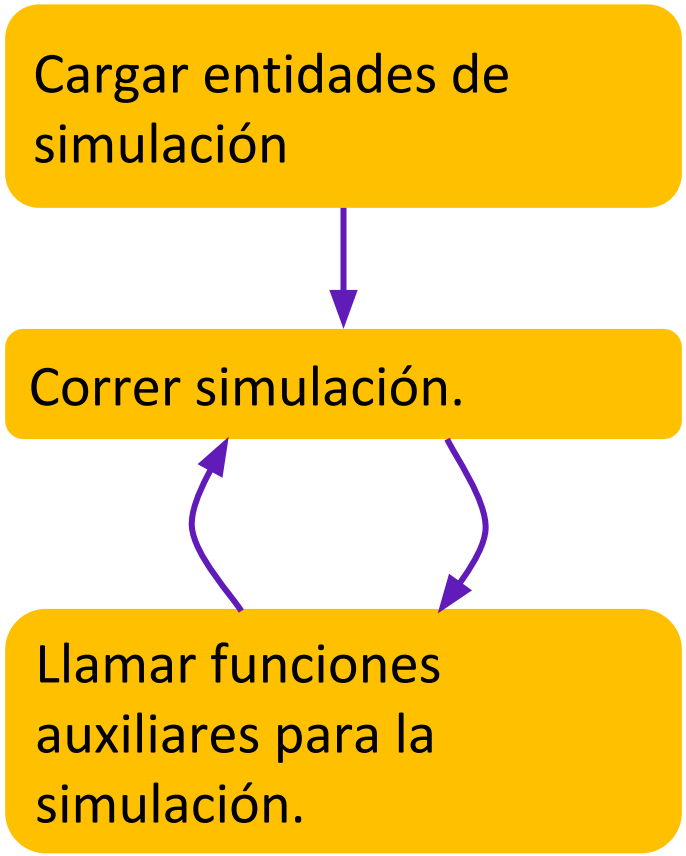
Estructura	Insertar	Búsqueda por índice	Búsqueda por llave	Búsqueda por valor
Lista	✓	✓✓✓	✗	✓
Tupla	✗	✓✓✓	✗	✓
Stack	✓ en un extremo	✗	✗	✗
Cola *(deque)*	✓✓✓ en ambos extremos	✓	✗	✓
diccionario	✓✓✓	✗	✓✓✓	✓
set	✓✓✓	✗	✓✓✓	✓

Actividad

1. En el *syllabus*, vayan a la carpeta “Actividades” y descarguen el enunciado de la actividad 2 (AC02) https://github.com/IIC2233/syllabus
2. Trabajen **individualmente** hasta las 16:30.
3. Recuerden hacer *commit* y *push* a sus repositorios personales cada cierto tiempo.

Cierre

Diagrama de flujo de AC02

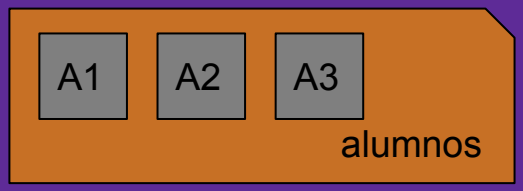



```
def cargar_alumnos(ruta)
def cargar_ayudantes(ruta)
```

```
def simular_batalla(...)
```

```
def distraer(...)
def resumen_actual(...)
def stock_comida(...)
```

Simulación

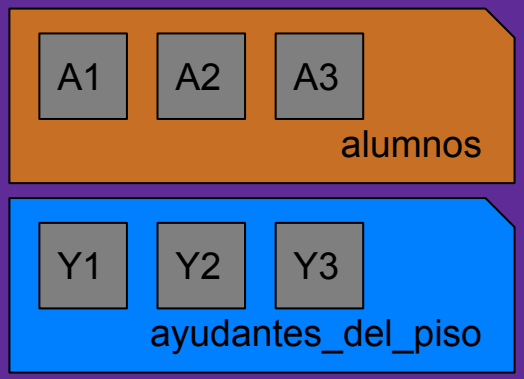


Tenemos los alumnos cargados en una estructura

```
# main.py
while PISOS and alumnos:
    # Mientras queden pisos y alumnos para distraer
    piso_actual = PISOS[0]
    ayudantes_del_piso = None
    while ayudantes_del_piso:
        # Mientras hayan ayudantes en el piso
        ayudante_defensor = None
        alumno_atacante = None
        while not distraer(alumno_atacante, ayudante_defensor):
            # Si no se logró distraer al ayudante
            # se debe mandar a la casa al alumno actual
            # Si quedan alumnos, intentamos con otro alumno,
            # si no, no po
            pass
        # Se acabaron los alumnos o se logró distraer al ayudante
        if not alumnos:
            # Si no quedan alumnos, no podemos distraer ayudantes
            Break
        elif ayudante_defensor.comiendo:
            # Si el ayudante fue distraido,
            # hay que cambiar al siguiente ayudante
            pass
    if not ayudantes_del_piso:
        # Si no quedan ayudantes, avanzamos de piso
        PISOS.popleft()
resumen_actual(ayudantes, alumnos)
```

Simulación

Los ayudantes de este piso también están en una estructura




```
# main.py
while PISOS and alumnos:
    # Mientras queden pisos y alumnos para distraer
    piso_actual = PISOS[0]
    ayudantes_del_piso = None
    while ayudantes_del_piso:
        # Mientras hayan ayudantes en el piso
        ayudante_defensor = None
        alumno_atacante = None
        while not distraer(alumno_atacante, ayudante_defensor):
            # Si no se logró distraer al ayudante
            # se debe mandar a la casa al alumno actual
            # Si quedan alumnos, intentamos con otro alumno,
            # si no, no po
            pass
        # Se acabaron los alumnos o se logró distraer al ayudante
        if not alumnos:
            # Si no quedan alumnos, no podemos distraer ayudantes
            Break
        elif ayudante_defensor.comiendo:
            # Si el ayudante fue distraido,
            # hay que cambiar al siguiente ayudante
            pass
    if not ayudantes_del_piso:
        # Si no quedan ayudantes, avanzamos de piso
        PISOS.popleft()
resumen_actual(ayudantes, alumnos)
```

Simulación

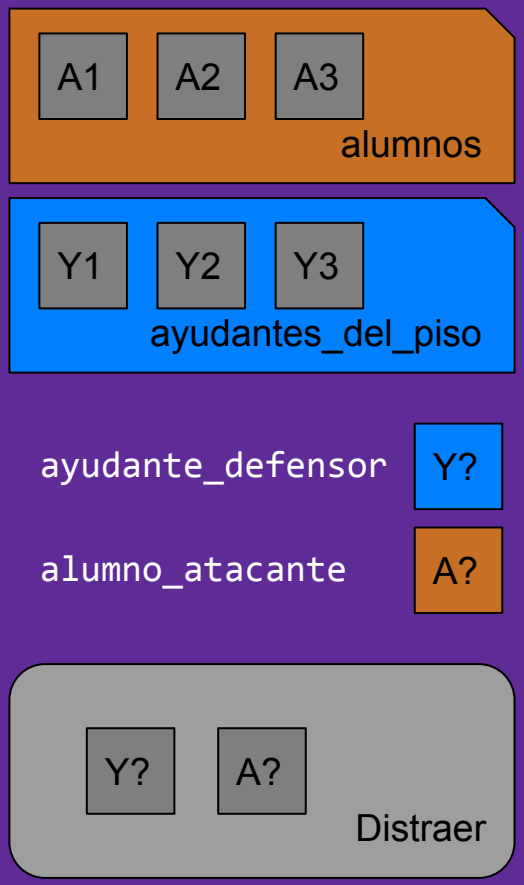
Tomamos un ayudante y un alumno

```
# main.py
while PISOS and alumnos:
    # Mientras queden pisos y alumnos para distraer
    piso_actual = PISOS[0]
    ayudantes_del_piso = None
    while ayudantes_del_piso:
        # Mientras hayan ayudantes en el piso
        ayudante_defensor = None
        alumno_atacante = None
        while not distraer(alumno_atacante, ayudante_defensor):
            # Si no se logró distraer al ayudante
            # se debe mandar a la casa al alumno actual
            # Si quedan alumnos, intentamos con otro alumno,
            # si no, no po
            pass
        # Se acabaron los alumnos o se logró distraer al ayudante
        if not alumnos:
            # Si no quedan alumnos, no podemos distraer ayudantes
            Break
        elif ayudante_defensor.comiendo:
            # Si el ayudante fue distraido,
            # hay que cambiar al siguiente ayudante
            pass
    if not ayudantes_del_piso:
        # Si no quedan ayudantes, avanzamos de piso
        PISOS.popleft()
resumen_actual(ayudantes, alumnos)
```

Simulación

Simulamos la distracción con el método `distraer`

```
# main.py
while PISOS and alumnos:
    # Mientras queden pisos y alumnos para distraer
    piso_actual = PISOS[0]
    ayudantes_del_piso = None
    while ayudantes_del_piso:
        # Mientras hayan ayudantes en el piso
        ayudante_defensor = None
        alumno_atacante = None
        while not distraer(alumno_atacante, ayudante_defensor):
            # Si no se logró distraer al ayudante
            # se debe mandar a la casa al alumno actual
            # Si quedan alumnos, intentamos con otro alumno,
            # si no, no po
            pass
        # Se acabaron los alumnos o se logró distraer al ayudante
        if not alumnos:
            # Si no quedan alumnos, no podemos distraer ayudantes
            Break
        elif ayudante_defensor.comiendo:
            # Si el ayudante fue distraido,
            # hay que cambiar al siguiente ayudante
            pass
    if not ayudantes_del_piso:
        # Si no quedan ayudantes, avanzamos de piso
        PISOS.popleft()
resumen_actual(ayudantes, alumnos)
```



Simulación

Si el alumno distrae exitosamente...

```
# main.py
while PISOS and alumnos:
    # Mientras queden pisos y alumnos para distraer
    piso_actual = PISOS[0]
    ayudantes_del_piso = None
    while ayudantes_del_piso:
        # Mientras hayan ayudantes en el piso
        ayudante_defensor = None
        alumno_atacante = None
        while not distraer(alumno_atacante, ayudante_defensor):
            # Si no se logró distraer al ayudante
            # se debe mandar a la casa al alumno actual
            # Si quedan alumnos, intentamos con otro alumno,
            # si no, no po
            pass
        # Se acabaron los alumnos o se logró distraer al ayudante
        if not alumnos:
            # Si no quedan alumnos, no podemos distraer ayudantes
            Break
        elif ayudante_defensor.comiendo:
            # Si el ayudante fue distraido,
            # hay que cambiar al siguiente ayudante
            pass
    if not ayudantes_del_piso:
        # Si no quedan ayudantes, avanzamos de piso
        PISOS.popleft()
resumen_actual(ayudantes, alumnos)
```

Simulación

Si no podemos distraer esta vez...

```
# main.py
while PISOS and alumnos:
    # Mientras queden pisos y alumnos para distraer
    piso_actual = PISOS[0]
    ayudantes_del_piso = None
    while ayudantes_del_piso:
        # Mientras hayan ayudantes en el piso
        ayudante_defensor = None
        alumno_atacante = None
        while not distraer(alumno_atacante, ayudante_defensor):
            # Si no se logró distraer al ayudante
            # se debe mandar a la casa al alumno actual
            # Si quedan alumnos, intentamos con otro alumno,
            # si no, no po
            pass
        # Se acabaron los alumnos o se logró distraer al ayudante
        if not alumnos:
            # Si no quedan alumnos, no podemos distraer ayudantes
            Break
        elif ayudante_defensor.comiendo:
            # Si el ayudante fue distraido,
            # hay que cambiar al siguiente ayudante
            pass
    if not ayudantes_del_piso:
        # Si no quedan ayudantes, avanzamos de piso
        PISOS.popleft()
resumen_actual(ayudantes, alumnos)
```

Estructuras de datos básicas

¿Se podría haber hecho la AC completa con solo listas?

El conocer una variedad de estructuras de datos nos permite aprovechar ventajas según el contexto en el que nos encontremos.

Las listas pueden ser multipropósito, pero muchas veces debemos trabajar extra para que se adecúe a la situación.

Tuplas

Permiten almacenar una colección heterogénea de datos: una entidad simple.

```
# cargar_datos.py
from collections import namedtuple

Ayudante = namedtuple('Ayudante',
           ['nombre', 'rango', 'debilidades', 'comiendo']
           )

Alumno = namedtuple('Alumno', ['nombre', 'habilidades'])
```

Stacks

Permiten almacenar datos que se necesitarán en orden LIFO.

```
# cargar_datos.py

def cargar_alumnos(ruta_archivo_alumnos):
    stack_alumnos = list() # o [], o incluso deque()
    # Leer archivo y agregar alumnos al stack
    return stack_alumnos
```

Stacks

Permiten almacenar datos que se necesitarán en orden LIFO.

```
# main.py
def simular_batalla(...):
    # ...
        # ...
            # ...
            alumno_atacante = alumnos[-1]
            while not distraer(alumno_atacante, ayudante):
               # Alumno para la casa
                alumnos.pop()
                if alumnos: # Reemplazamos el alumno, si hay más
                   alumno_atacante = alumnos[-1]
                else: # Si no, dejamos de intentar
                    break
    # ...
```

Colas

Permiten almacenar datos que se necesitarán en orden FIFO.

```
# main.py
def simular_batalla(...):
    # ...
        # ...
        piso_actual = PISOS[0]
            # ...
            ayudante_defensor = ayudantes_del_piso[0]
            # ...
            # ...
            elif ayudante_defensor.comiendo:
                # ayudante fue distraido
                ayudantes_del_piso.popleft()
        if not ayudantes_del_piso:
            PISOS.popleft()
    # ...
```

Diccionarios

Permiten almacenar datos organizados por llaves.

```
# cargar_datos.py
from collections import defaultdict, deque

def cargar_ayudantes(ruta_archivo_ayudantes):
    ayudantes = defaultdict(deque)
    # Leer archivo
        # ...
        nuevo_ayudante = Ayudante(nombre, rango, ...)
        piso_ayudante = PISOS[rango]
        ayudantes[piso_ayudante].append(nuevo_ayudante)
    return ayudantes
```

Diccionarios

Permiten almacenar datos organizados por llaves.

```
# main.py
def simular_batalla(...):
    # ...
        # ...
        piso_actual = PISOS[0]
        ayudantes_del_piso = ayudantes[piso_actual]
        # ...
    # ...
```

Conjuntos (*Sets*)

Permiten mantener colecciones de datos donde interesa la pertenencia de elementos, más que el orden. Útiles para quitar repetidos y para revisar contención.

```
# cargar_datos.py
def cargar_alumnos(ruta_archivo_alumnos):
    # ...
        habilidades = set(habilidades.split(‘,’))
    # ...

def cargar_ayudantes(ruta_archivo_ayudantes):
    # ...
        debilidades = set(debilidades.split(‘,’))
    # ...
```

Conjuntos (*Sets*)

Permiten mantener colecciones de datos donde interesa la pertenencia de elementos, más que el orden. Útiles para quitar repetidos y para revisar contención.

```
# main.py

def distraer(alumno, ayudante):
    habilidades = alumno.habilidades
    debilidades = ayudante.debilidades
    comun = debilidades.intersection(habilidades)
    if comun:
        # distraer ayudante y remover habilidad
    else:
        # no distraer
```