

Programación Avanzada

II C2233 2025-2

Cristian Ruz - Pablo Araneda - Francisca Ibarra - Tamara Vidal - Daniela Concha



Anuncios

1. Hoy es la Actividad 3.
2. La ECA se encuentra disponible para responder de domingo a martes.
3. La ETC está abierta.
4. La Tarea 2 ya está publicada.
Revisen las issues ya creadas antes de publicar una nueva.
5. Fonda "Don Yadran"

ITERABLES

Iterables e Iteradores

Tip: Es más fácil acordarse si uno piensa que son recorribles y recorredores, respectivamente.

Iterables e iteradores

- ¿Qué elementos de la vida cotidiana se podrían modelar usando un iterable? ¿Hay algo que pueda ser un iterador para cada caso?
- Al momento de usar `__getitem__` para un iterable, ¿qué se debe tener en cuenta?

¿Cuál es el iterable y cuál el iterador? ¿Cuál funciona de la forma deseada aquí?

Iterables e iteradores: Aplicando en código

```
dias = ["aún no sale", "aún no sale", "sale mañana",  
"sale Silksong", "ya salió", "ya salió", "ya salió"]  
texto = "Jugando Silksong cuando"
```

```
copy_dias = dias.copy()  
  
for dia in copy_dias:  
    if dia == "sale mañana":  
        break  
  
for dia in copy_dias:  
    print(texto, dia)
```

```
iter_dias = iter(dias)  
  
for dia in iter_dias:  
    if dia == "sale mañana":  
        break  
  
for dia in iter_dias:  
    print(texto, dia)
```

¿Cuál es el iterable y cuál el iterador? ¿Cuál funciona de la forma deseada aquí?

Iterables e iteradores: Aplicando en código

```
dias = ["aún no sale", "aún no sale", "sale mañana",  
"sale Silksong", "ya salió", "ya salió", "ya salió"]  
texto = "Jugando Silksong cuando"
```

```
copy_dias = dias.copy()  
  
for dia in copy_dias:  
    if dia == "sale mañana":  
        break  
  
for dia in copy_dias:  
    print(texto, dia)
```

```
Jugando Silksong cuando aún no sale  
Jugando Silksong cuando aún no sale  
Jugando Silksong cuando sale mañana  
    Jugando Silksong cuando sale  
        Silksong  
Jugando Silksong cuando ya salió  
Jugando Silksong cuando ya salió  
Jugando Silksong cuando ya salió
```

¿Cuál es el iterable y cuál el iterador? ¿Cuál funciona de la forma deseada aquí?

Iterables e iteradores: Aplicando en código

```
dias = ["aún no sale", "aún no sale", "sale mañana",  
"sale Silksong", "ya salió", "ya salió", "ya salió"]  
texto = "Jugando Silksong cuando"
```

```
Jugando Silksong cuando sale  
    Silksong  
Jugando Silksong cuando ya salió  
Jugando Silksong cuando ya salió  
Jugando Silksong cuando ya salió
```

```
iter_dias = iter(dias)  
  
for dia in iter_dias:  
    if dia == "sale mañana":  
        break  
  
for dia in iter_dias:  
    print(texto, dia)
```


¿Iterador = Iterable?

NO, pero en Python un Iterador puede funcionar como Iterable. Para entender mejor esto, veamos algunas de sus principales diferencias:

Característica	Iterable	Iterador
Métodos Asociados	Implementa el método <code>__iter__()</code> que devuelve un iterador, o el método <code>__getitem__()</code> que va entregando elementos de forma creciente.	Implementa el método <code>__next__()</code> . Es el objeto retornado de un <code>__iter__()</code> .
Estado Interno	Un iterable no tiene estado interno de iteración.	Un iterador tiene un estado interno que recuerda su posición actual en la iteración.
Reutilización	Los iterables pueden ser reutilizados para obtener múltiples iteradores.	Los iteradores, una vez consumidos, no pueden ser reutilizados .

Generadores

Generadores


- ¿Cuál es la ventaja de usar generadores?

Generadores

```
def procesar_numero(numero, contexto):  
    print(numero, 'en', contexto)  
    return numero ** 2  
  
generador = (procesar_numero(numero, 'generador')  
             for numero in range(2))  
  
lista = [procesar_numero(numero, 'lista')  
         for numero in range(2)]  
  
for numero in generador:  
    print(numero, 'en for')
```

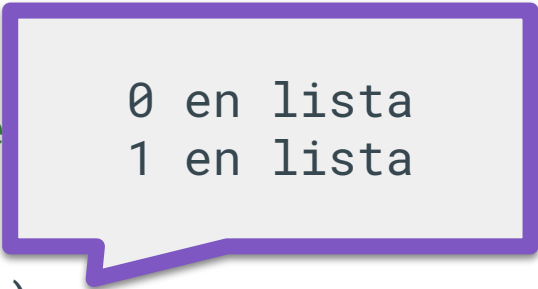
Generadores

```
def procesar_numero(numero, contexto):  
    print(numero, 'en', contexto)  
    return numero ** 2  
  
generador = (procesar_numero(numero, 'generador')  
             for numero in range(2))  
  
lista = [procesar_numero(numero, 'lista')  
         for numero in range(2)]  
  
for numero in generador:  
    print(numero, 'en for')
```



Generadores

```
def procesar_numero(numero, contexto):  
    print(numero, 'en', contexto)  
    return numero ** 2  
  
generador = (procesar_numero(numero, 'ge  
              for numero in range(2))  
  
lista = [procesar_numero(numero, 'lista')  
         for numero in range(2)]  
  
for numero in generador:  
    print(numero, 'en for')
```



0 en lista
1 en lista

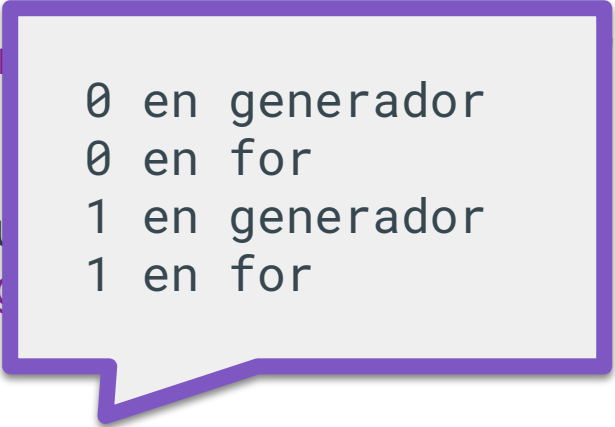
Generadores

```
def procesar_numero(numero, contexto):  
    print(numero, 'en', contexto)  
    return numero ** 2
```

```
generador = (procesar_numero(numero, 'generador')  
             for numero in range(2))
```

```
lista = [procesar_numero(numero, 'lista')  
         for numero in range(2)]
```

```
for numero in generador:  
    print(numero, 'en for')
```

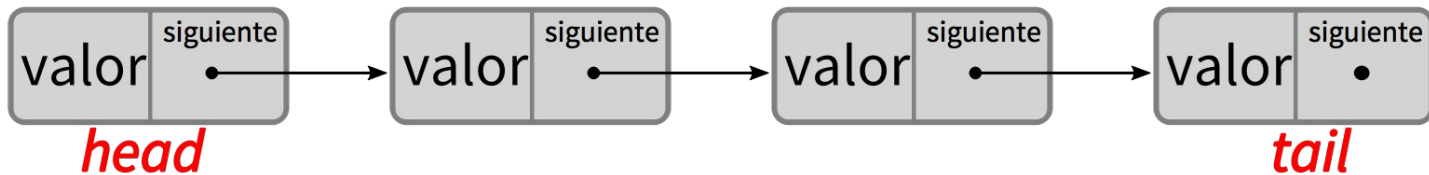


0 en generador
0 en for
1 en generador
1 en for

Listas Ligadas

Listas Ligadas

- ¿Qué función cumple el Nodo?
- ¿A qué ayuda que la Lista Ligada tenga el atributo cola?
 - ¿Cómo se actualiza si se elimina el nodo que corresponde a la cola?
 - ¿Se puede hacer algo que vuelva la operación más fácil?
- ¿Cómo se haría si se desea concatenar una Lista Ligada a otra?



Pregunta de Evaluación Escrita

Tema: Iterables e iteradores (Midterm 2025-1)

5. ¿Cuál de los siguientes tipos de datos **no** presenta un iterador?
- A) *Strings*
 - B) Conjuntos
 - C) Tuplas
 - D) Diccionarios
 - E) Todos los tipos de datos anteriores presentan un iterador.

Pregunta de Evaluación Escrita

Tema: Iterables e iteradores (Midterm 2025-1)

5. ¿Cuál de los siguientes tipos de datos **no** presenta un iterador?

- A) *Strings*
- B) Conjuntos
- C) Tuplas
- D) Diccionarios

E) Todos los tipos de datos anteriores presentan un iterador.

Actividad

Comentarios AC3

NO GIT PUSH NO GAIN!

Programación Avanzada

IIC2233 2025-2

Cristian Ruz - Pablo Araneda - Francisca Ibarra - Tamara Vidal - Daniela Concha

