Programación Avanzada
IIC2233 2025-2

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

Actividad 3

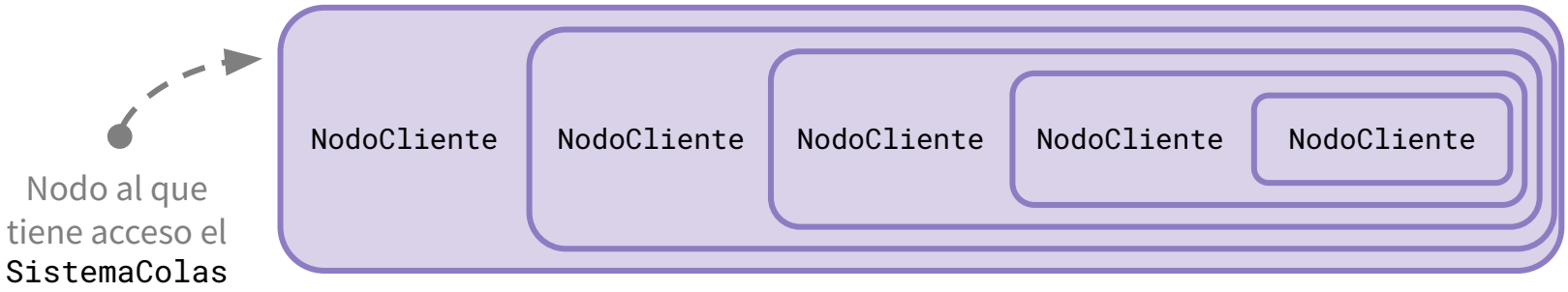
Iterables e Iteradores

Entendiendo NodoCliente

A diferencia de lo visto en el contenido, `NodoCliente` no es solo un `Nodo`, esta estructura es un híbrido entre un `Nodo` y una `ListaLigada` la cual al mismo tiempo:

- Guarda la información del nodo (p. ej. su valor).
- Guarda la cadena de nodos que forma la estructura, es decir, los nodos siguientes.

En resumen, esto significa que **esta estructura hace más que un `Nodo` normal**.



Implementando métodos de `NodoCliente`

Como `NodoCliente` es una estructura que contiene contiene una cadena de nodos que lo anteceden, muchos de sus métodos **dependen de los nodos que lo suceden**, es decir, los `NodoCliente` guardados en el atributo `siguiente`.

Por lo mismo, debemos **revisar los nodos de la cadena**. Esto se puede hacer de:

- **Forma iterativa**, si nos apoyamos de variables auxiliares para guardar estados.
- **Forma recursiva**, si manejamos adecuadamente el caso base y recursivo.

Recuerden, todo algoritmo recursivo se puede hacer iterativo y viceversa

Implementando métodos de `NodoCliente`

En las siguientes diapositivas se mostrará un **pseudo-código de posibles implementaciones** de cada método de `NodoCliente`.

Ten lo siguiente en consideración, el pseudo-código contendrá:

- Comentarios que ayudarán a entender mejor el contexto de ciertas decisiones o partes del código.
- Textos en *`este formato`*, que corresponderán a ciertas instrucciones que no son propias de Python.

NodoCliente.agregar_nodo

Alg. Recursivo

```
def agregar_nodo(self, nuevo_nodo: NodoCliente):
    # Caso base
    if (este nodo no tiene un Siguiente):
        self.siguiente = nuevo_nodo
        # Terminamos la ejecución del algoritmo.
        return

    # Caso Recursivo
    # Como este nodo sí tiene un Siguiente, se
    # pasa el llamado a ese NodoCliente.
    Siguiente.agregar_nodo(nuevo_nodo)
```

Alg. Iterativo

```
def agregar_nodo(self, nuevo_nodo: NodoCliente):
    # Variable auxiliar que guarda el nodo que
    # actualmente estamos revisando.
    nodo_actual = nodo donde empezó el llamado

    # Avanzaremos en la cadena de nodos
    # hasta que no haya un Siguiente
    while (nodo actual tiene un Siguiente):
        nodo_actual = Siguiente

    # Llegaremos a esta parte cuando el nodo
    # nodo actual se encuentre al final de la
    # cadena de nodos, por lo que podemos agregar
    # el nuevo nodo al final de la cadena.
    nodo_actual.siguiente = nuevo_nodo
```

NodoCliente.__str__

Alg. Recursivo

```
def __str__(self) -> str:
    # Caso Recursivo
    # Generamos el texto correspondiente a este
    # nodo y llamamos el __str__ del Siguiente.
    return texto de este nodo + str(Siguiente)
```

El algoritmo recursivo
no necesita de un caso base,
ya que cuando **Siguiente sea None**
terminará el llamado recursivo

Alg. Iterativo

```
def __str__(self) -> str:
    # Variables auxiliares
    nodo_actual = nodo donde empezó el llamado
    texto = ''

    # Avanzaremos en la cadena de nodos
    # hasta que no haya un Siguiente
    while (nodo actual tiene un Siguiente):
        texto += texto de este nodo

    # Llegamos al final de la cadena de nodos.
    texto += texto de un None

    # Retornamos el texto final
    return texto
```

NodoCliente.__len__

Pensando en los dos casos anteriores, ahora trata de **ver los patrones** de los algoritmos recursivos e iterativos y aplicalos para implementar el método `__len__`.

Alg. Recursivo

```
def __len__(self) -> int:
    # Caso base
    # ...
    # ...

    # Caso Recursivo
    # ...
    # ...
```

Alg. Iterativo

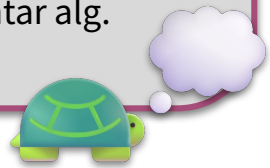
```
def __len__(self) -> int:
    # Variables auxiliares
    # ...
    # ...

    # Avanzar loop
    # ...
    # ...

    # Retorno
    # ...
```

Recuerda los 4 pasos del **Pensamiento Computacional** vistos en Intro a la Progra:

1. Descomponer
2. Reconocer patrones
3. Abstraerse
4. Implementar alg.



Programación Avanzada
IIC2233 2025-2

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