Creative Coding 8

David Lynch

Slide Deck by Ruán Murgatroyd

Last Class

- Intro to OOP
- OOP’s History
- Class inheritance
- Polymorphism
- Art Expo

Today's Class

- Python Dictionaries
- Intro to data structures (linked lists, stacks)
- How to use and reason with these data structures
- Implementing snake in *Processing*
- The future of Creative Coding.

Dictionaries

Dictionaries

Dictionaries are a Python datatype. While similar to lists, it has a few key differences.

Lists index numerically and in order, you can only access a list with an integer.

Dictionaries use *key-value* pairs where each element of a dictionary has a unique key of any data type, instead of just an integer.

Dictionaries

Key	Value
“BestSoc”	“CS++”
-100	“hello”
(1, 2)	True
False	20

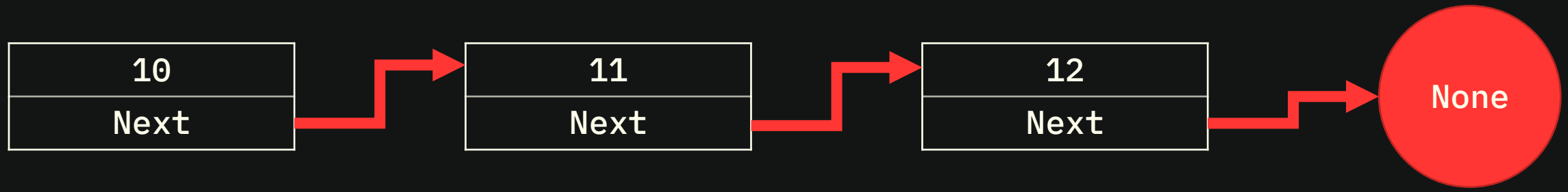
Index	Value
0	“CS++”
1	“hello”
2	True
3	20

Data Structures: Linked Lists

Linked lists are another way of representing lists. Instead of elements being adjacent in memory, they're similar to a chain of nodes.

We can think of a node as storing its data and a reference to the next node in the list.

We must always ensure that the last element is None, this is called **NULL TERMINATION**



Operations on Linked Lists

Linked lists give us more flexibility as we are not bound by the limitations of arrays.

Arrays needs to be adjacent in memory, making it difficult to modify the array as the entire array would need to be re-written.

Linked lists allow us to perform add, remove and duplicate operations easily.

A Node in Python

We will build the operations for this ourselves.

Note: Technically Python's lists are linked lists.

pythonNode.py

```
def __init__(self, value):
    self.value = value
    self.next = None
```

Stacks

Stacks are data structures which follow **LIFO** (Last in, First out)

This means that the only way to add data to the stack is by **PUSH**ing it onto the top and the only way to remove data is by **POP**ping it off the stack.

If we implemented a stack using a linked list, then we can say the element pointing to None is the top of the stack

Traversing Linked List Data Structures

Linked lists allow us to traverse them either with loops or recursion.

These are two possible ways to traverse linked lists.

linkedList.py

```python
def traverseLoop(head):
    t = head
    while t != None:
        print(t.val)
        t = t.next


def traverseRec(head):
    print(head.val)
    if head.next != None:
        traverseRec(head.next)
```

Data Structure Drills!

Write the following methods for the Node Class.

1. A method which allows you to push elements onto a stack.
2. A method which allows you to pop an element off a stack.
3. A method which will give you the length of a linked list.
4. A method to add an element to a linked list at a given index.
5. A method to remove an element of a linked list at a given index.
6. A method to perform math operations on two elements of a linked list together when given an index.

Use looping or recursion!

Processing Time

Processing Time

Snake is a game found mostly from mobile devices of the early ’00s. In it, the player controls a snake which grows whenever it eats food.

If the snake runs into itself, it will die and the game ends.

We will look at the steps needed to design snake and implement it.

Steps to Snake

1. Pick a data structure to represent our snake
2. Implement drawing the snake to the screen
3. Implement the snake’s movement
4. Implement the snake’s dynamic growth
5. Check collisions with food and the snake
6. Check collisions with the snake and itself
7. Implement a scoring system
8. Fun stuff!

The End

Thanks for coming!

Notes uploaded on the Discord.

See you next week!