

BINARY SEARCH TREES

LECTURE 11-1

JIM FIX, REED COLLEGE CSCI 121

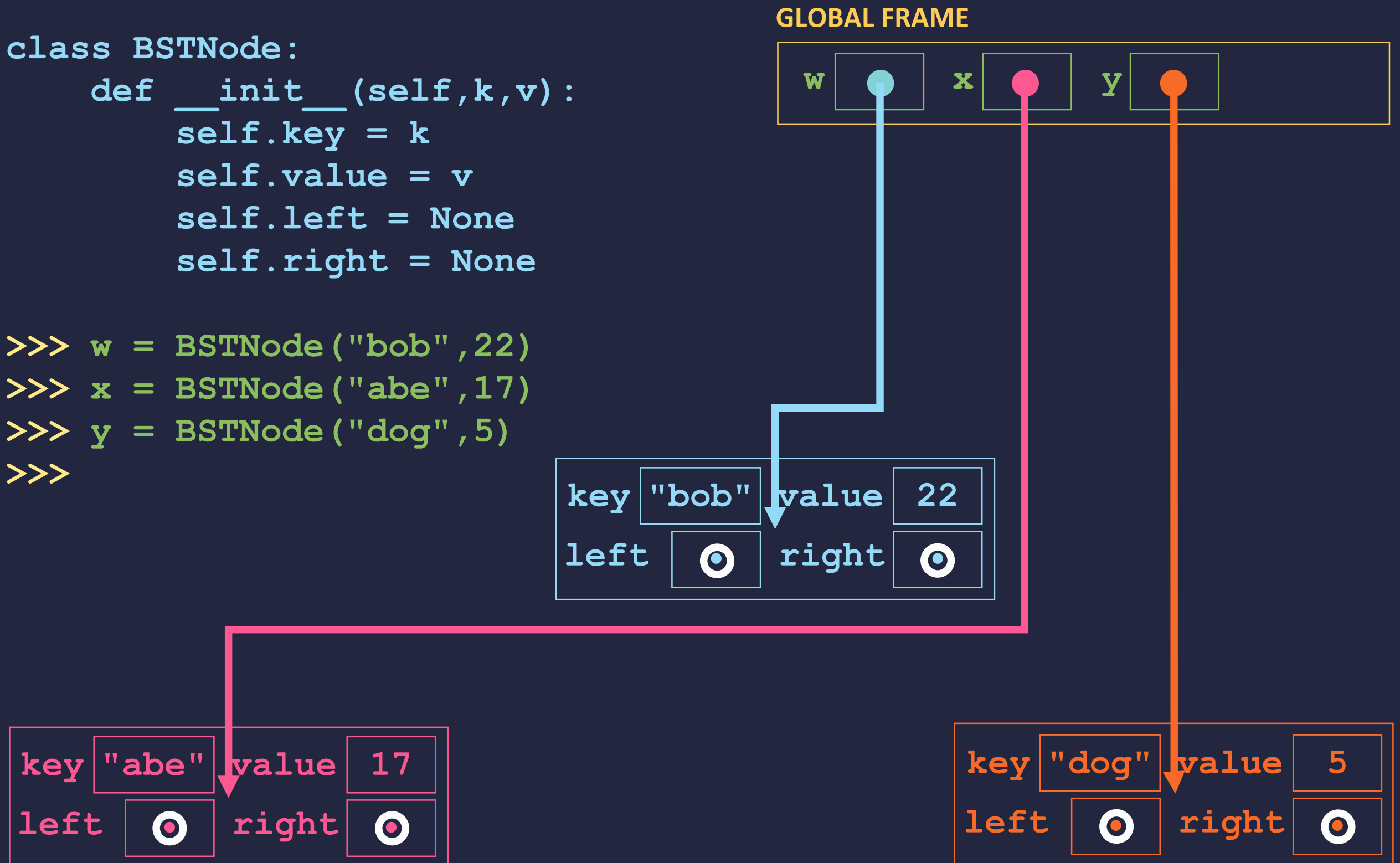
COURSE INFO

- ▶ **Project 4:**
 - due 4/28.
 - **adventure**: a text-based role-playing game
 - You can work with a partner if you like.

A (BINARY) TREE NODE CLASS

```
class BSTNode:
    def __init__(self, k, v):
        self.key = k
        self.value = v
        self.left = None
        self.right = None
```

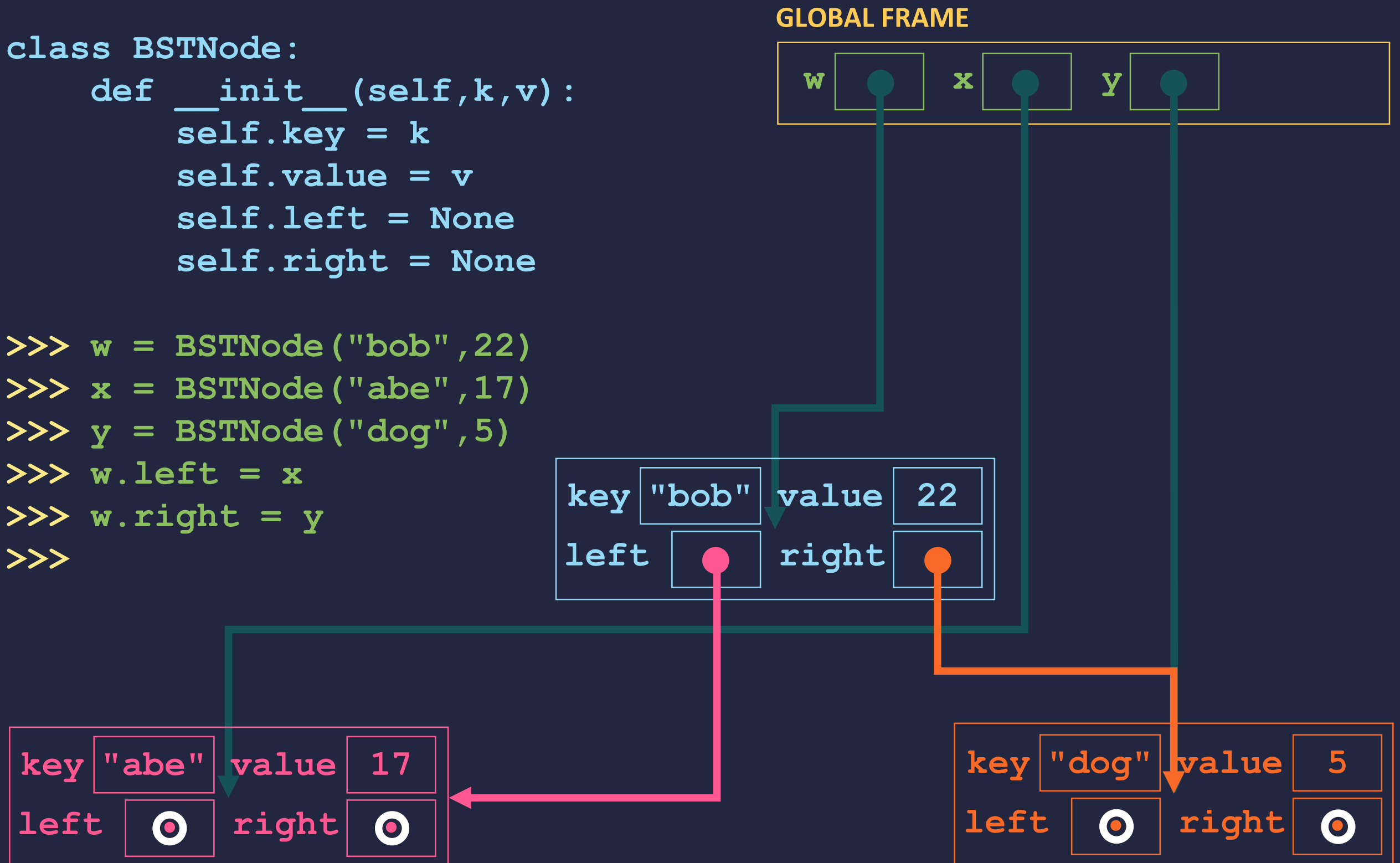
```
>>> w = BSTNode("bob", 22)
>>> x = BSTNode("abe", 17)
>>> y = BSTNode("dog", 5)
>>>
```



A (BINARY) TREE NODE CLASS

```
class BSTNode:
    def __init__(self, k, v):
        self.key = k
        self.value = v
        self.left = None
        self.right = None
```

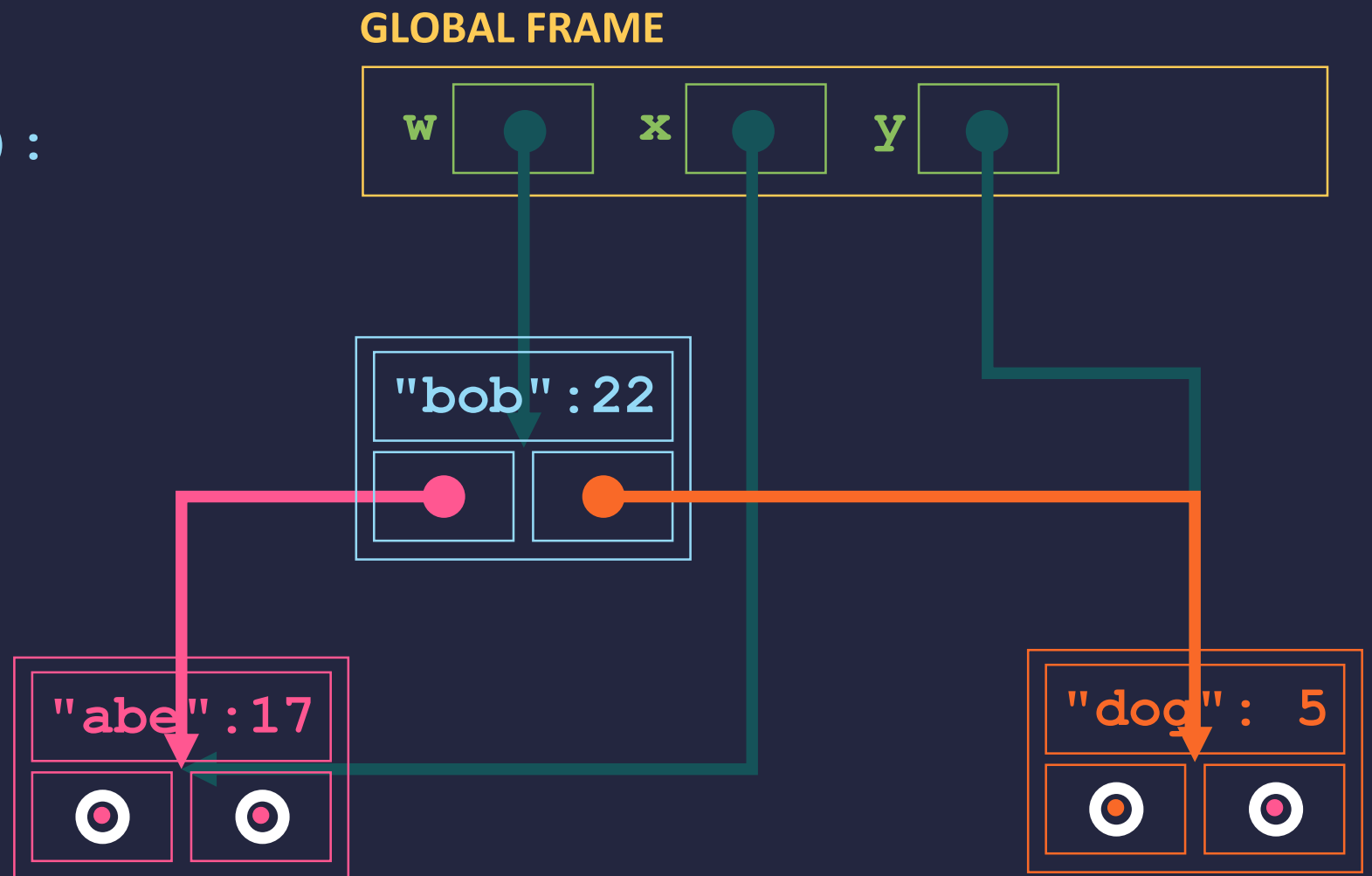
```
>>> w = BSTNode("bob", 22)
>>> x = BSTNode("abe", 17)
>>> y = BSTNode("dog", 5)
>>> w.left = x
>>> w.right = y
>>>
```



A (BINARY) TREE NODE CLASS

```
class BSTNode:
    def __init__(self, k, v):
        self.key = k
        self.value = v
        self.left = None
        self.right = None
```

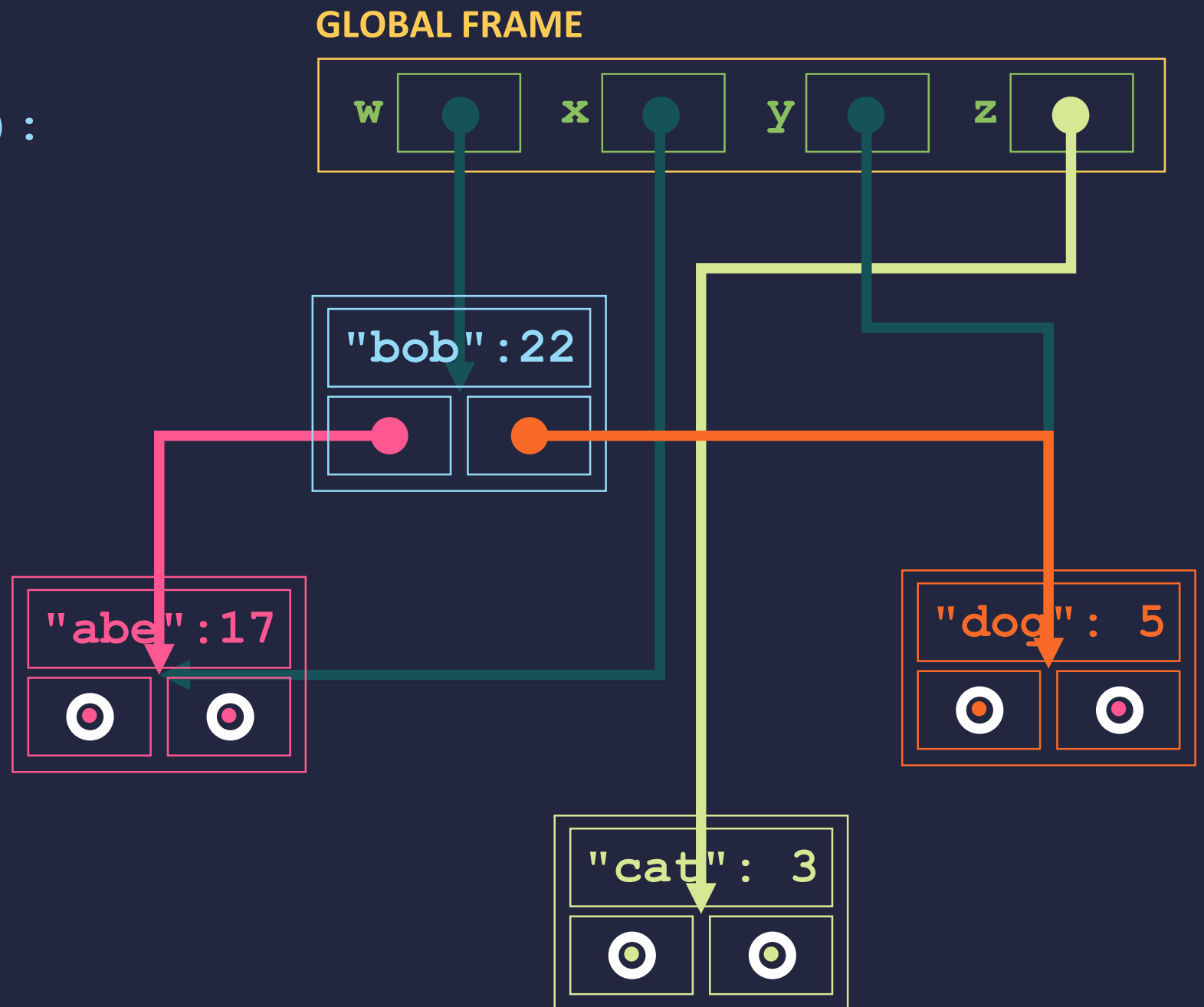
```
>>> w = BSTNode("bob", 22)
>>> x = BSTNode("abe", 17)
>>> y = BSTNode("dog", 5)
>>> w.left = x
>>> w.right = y
>>>
```



A (BINARY) TREE NODE CLASS

```
class BSTNode:
    def __init__(self, k, v):
        self.key = k
        self.value = v
        self.left = None
        self.right = None
```

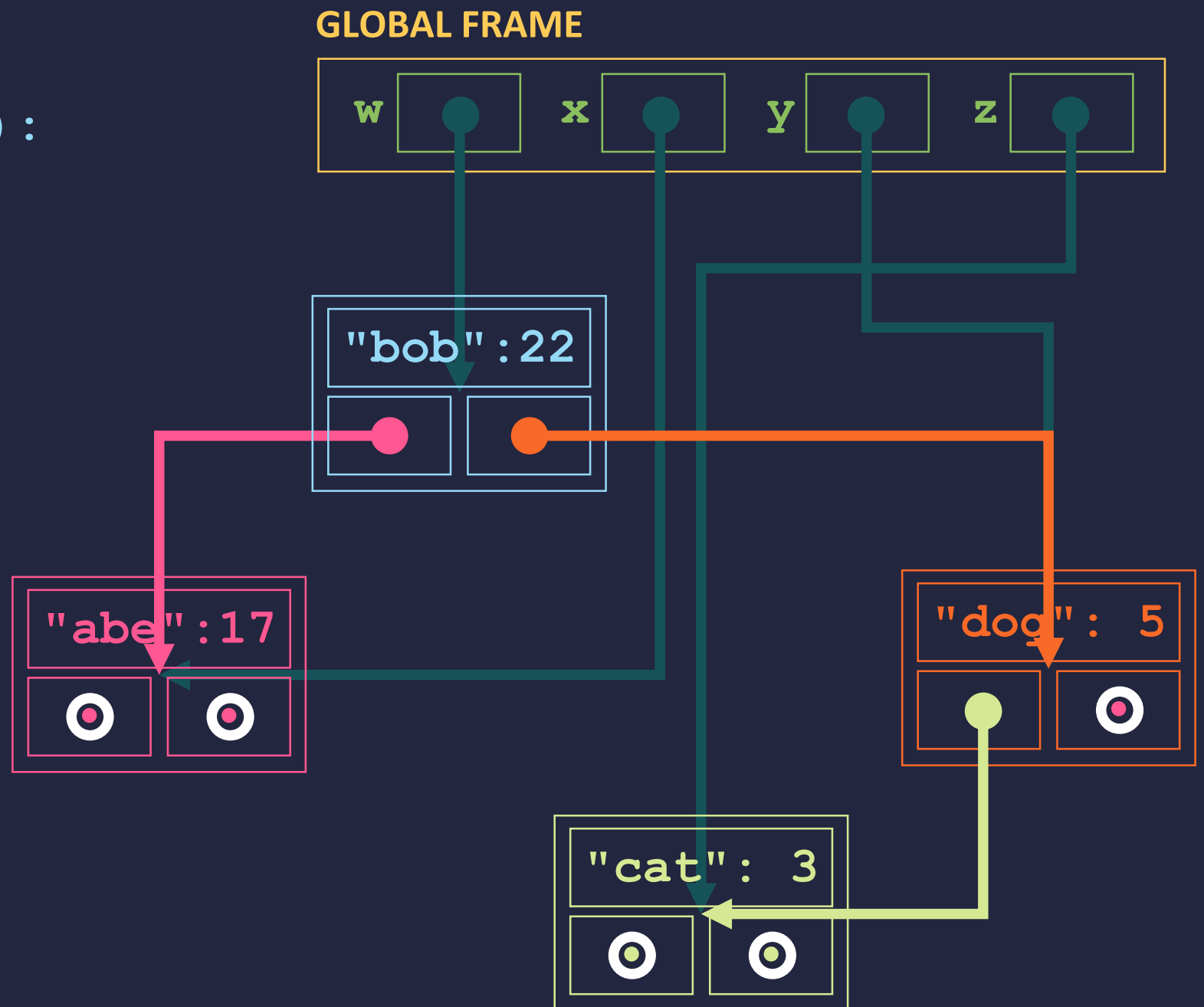
```
>>> w = BSTNode("bob", 22)
>>> x = BSTNode("abe", 17)
>>> y = BSTNode("dog", 5)
>>> w.left = x
>>> w.right = y
>>> z = BSTNode("cat", 3)
>>>
```



A (BINARY) TREE NODE CLASS

```
class BSTNode:
    def __init__(self, k, v):
        self.key = k
        self.value = v
        self.left = None
        self.right = None
```

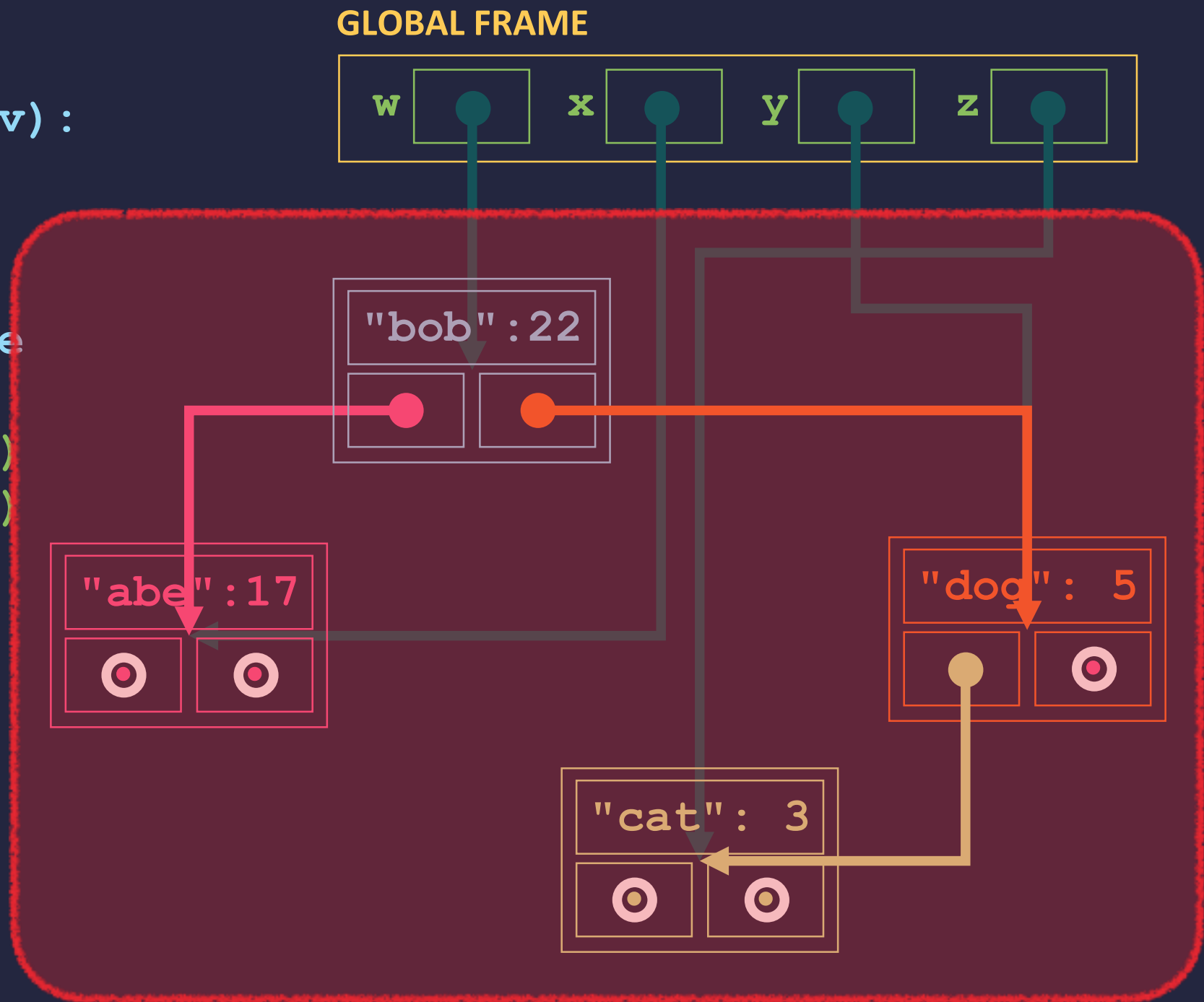
```
>>> w = BSTNode("bob", 22)
>>> x = BSTNode("abe", 17)
>>> y = BSTNode("dog", 5)
>>> w.left = x
>>> w.right = y
>>> z = BSTNode("cat", 3)
>>> y.left = z
>>>
```



A (BINARY) TREE NODE CLASS

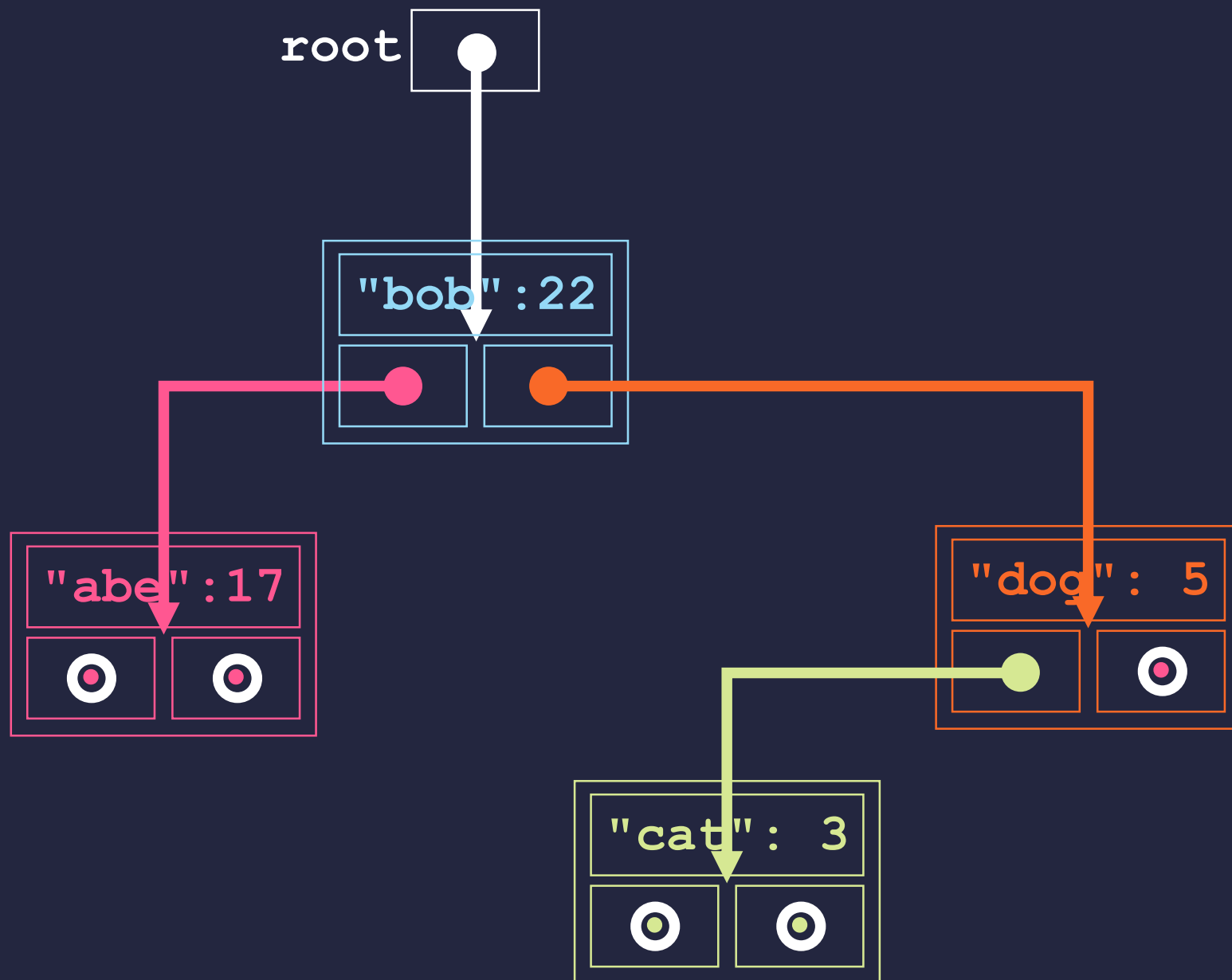
```
class BSTNode:
    def __init__(self, k, v):
        self.key = k
        self.value = v
        self.left = None
        self.right = None
```

```
>>> w = BSTNode("bob", 22)
>>> x = BSTNode("abe", 17)
>>> y = BSTNode("dog", 5)
>>> w.left = x
>>> w.right = y
>>> z = BSTNode("cat", 3)
>>> y.left = z
>>>
```

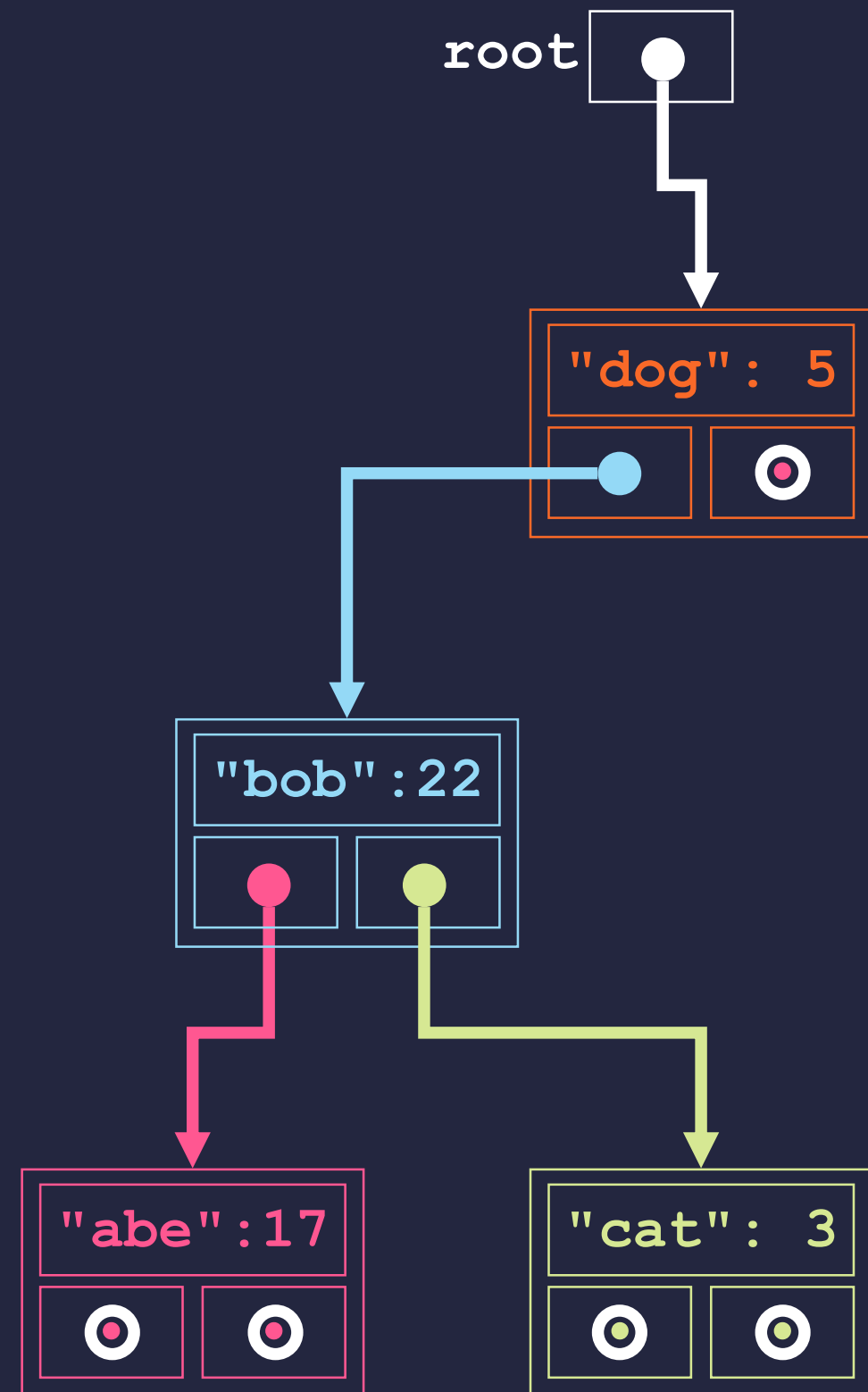


THIS STRUCTURE IS CALLED A BINARY SEARCH TREE

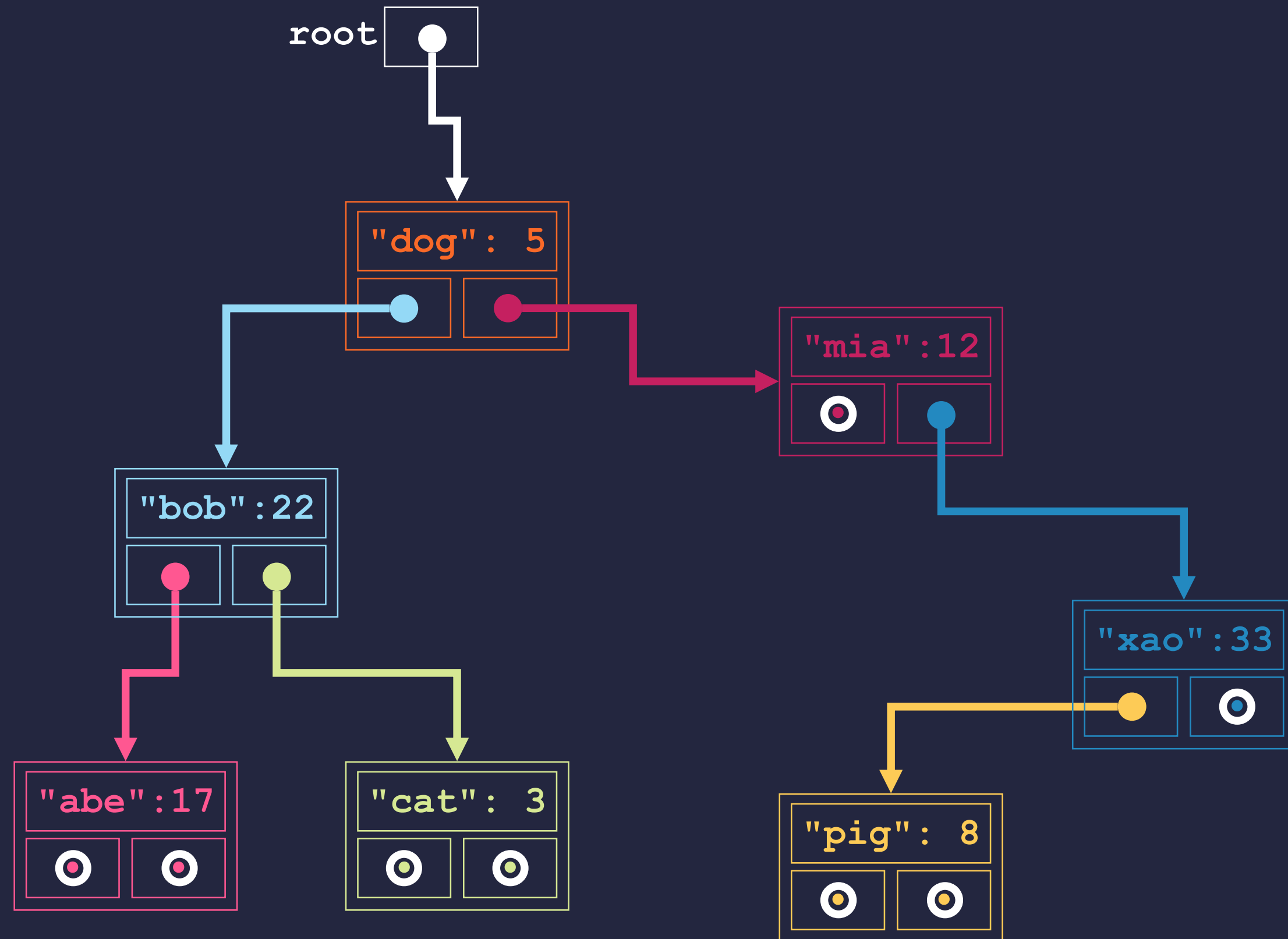
A BINARY SEARCH TREE



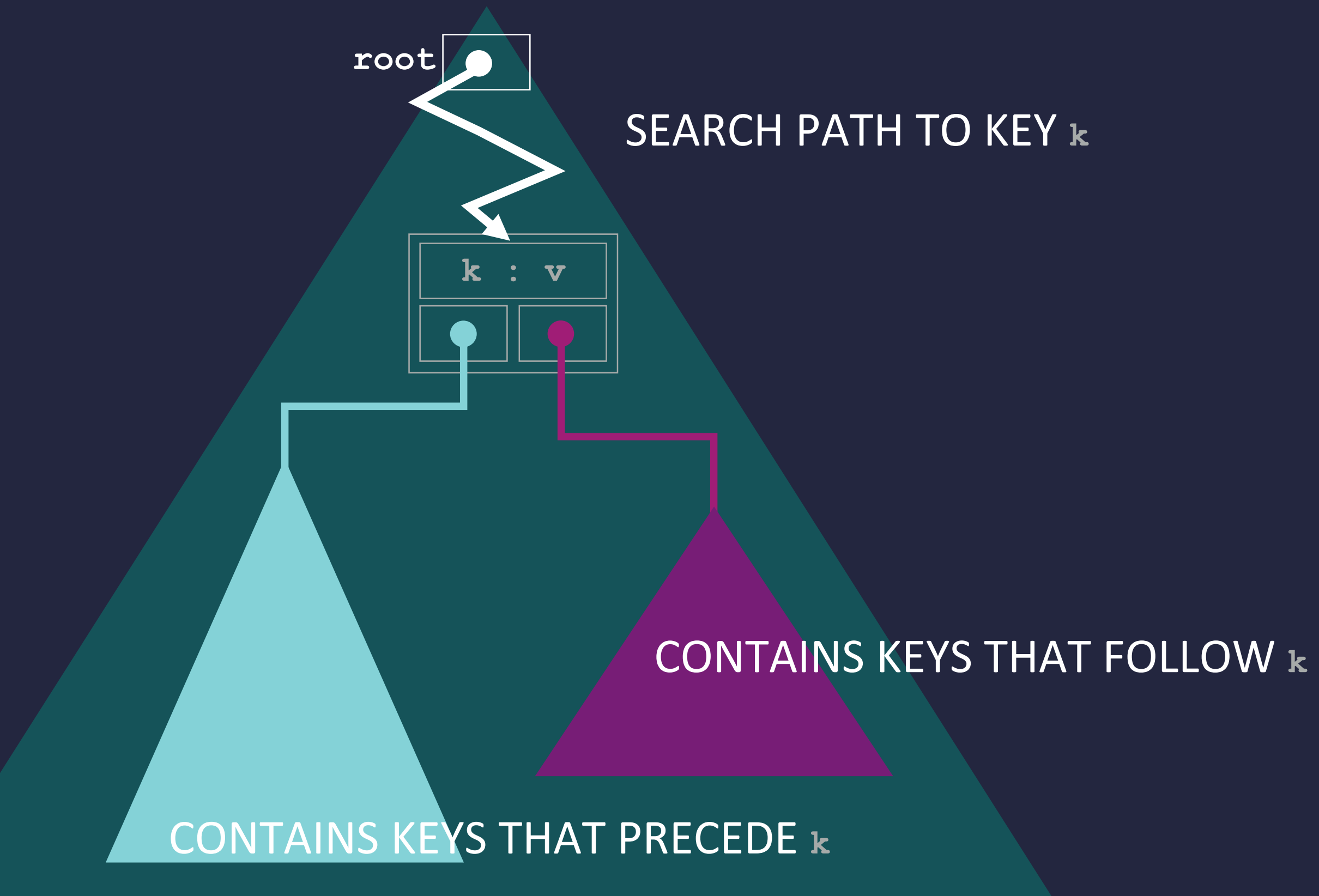
ANOTHER BINARY SEARCH TREE



AND YET ANOTHER BINARY SEARCH TREE



PROPERTIES OF A BINARY SEARCH TREE



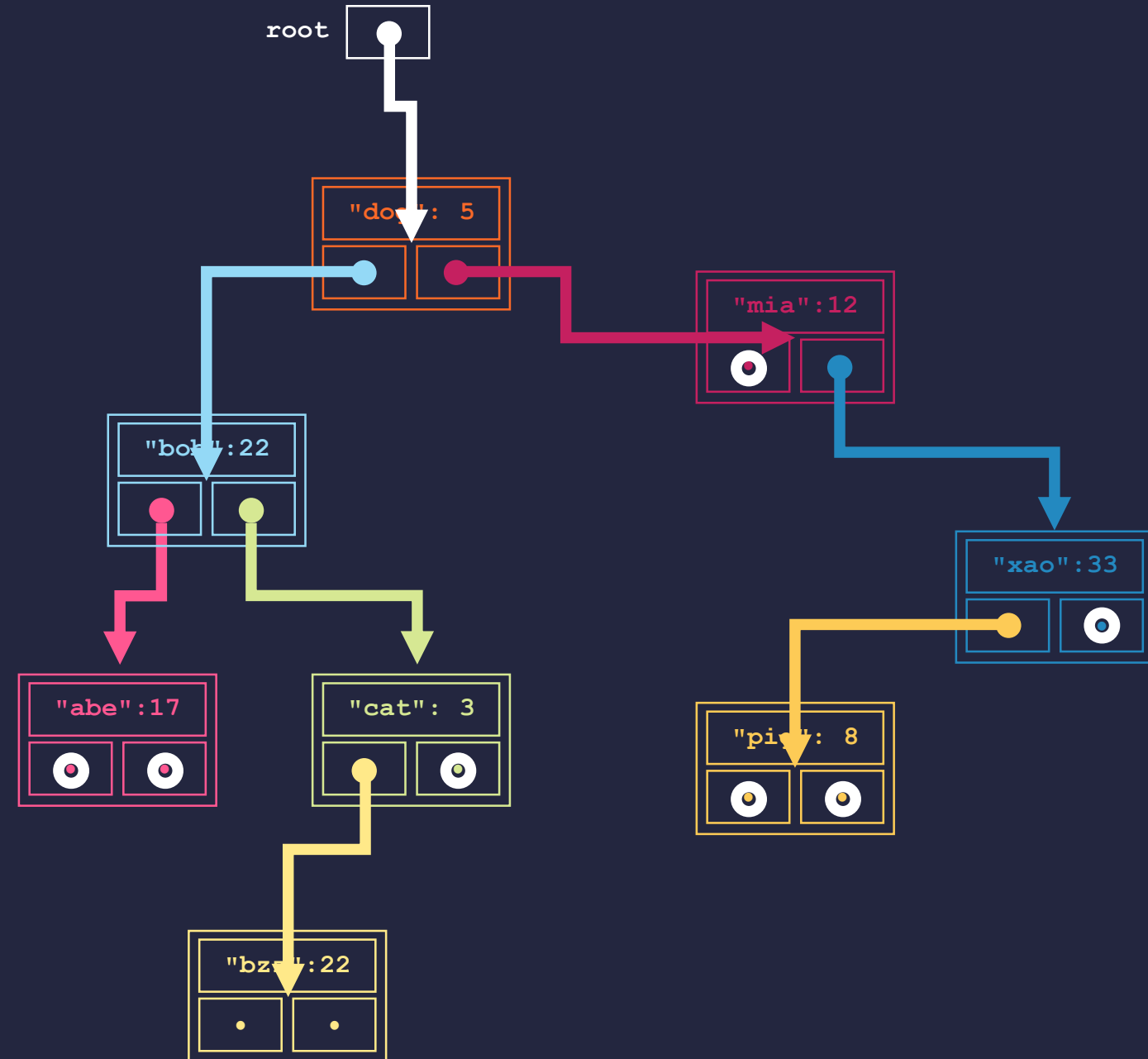
SEARCHING FOR AN ENTRY IN A BST

```

class BSTNode:
    def __init__(self, k, v):
        self.key = k
        self.value = v
        self.left = None
        self.right = None

def search(root, k):
    curr = root
    while curr is not None:
        if k == curr.key:
            return curr.value
        if k < curr.key:
            curr = curr.left
        if k > curr.key:
            curr = curr.right
    return None

```



BINARY SEARCH TREES

- ▶ Binary search trees are a way of keeping track of a *sorted* collection.
- ▶ Here, we are using them as an *ordered dictionary*.
- ▶ For our dictionaries, there is at most one entry per key.
- ▶ The link structure sorts the entries; maintains a sorted order.
 - The keys are usually organized alphabetically when strings.
 - The keys are usually sorted smaller/larger if numbers.
- ▶ (Generally, in binary search trees, keys might appear more than once; have multiple entries.)
- ▶ (Generally, in binary search trees, the nodes might only contain keys without associated values.)

A BST CLASS

► Operations:

- ➡ Adding an entry, ordered according to key.
 - ➡ Searching for an entry by key.
 - ➡ Removing an entry.
 - ➡ Visiting/traversing all the entries in sorted order.
- The first three operations rely on a *search*.
- ➡ This works from the root, moving left or right.
- An *in-order traversal* is a recursive method. Example: printing all the entries
- ➡ You print all of the entries left of the root entry.
 - ➡ Then you print the root entry.
 - ➡ And then you print all of the entries right of the root entry.

A TOUR OF THE BST CLASS CODE

- ▶ **Look at** `BSTree.py`