Java and Python comparison

Lets now compare Python and Java without a specific context.

Consider a function to traverse a data structure (a tree, for instance) to find objects that match certain criteria.
The matching function is generic: It traverses the structure and calls a caller-supplied matching function to match two objects.
It makes sense to have the tree class supply the matching logic (which hides the details of efficiently traversing the tree),
while allowing the caller to specify the matching function.

```java
//Java implementation, using an interface.
import java.util.Collection;
import java.util.ArrayList;

public interface Matcher {
    boolean matches(T o);
}

public class MyTree {
    // details omitted

    Collection matches(Matcher matcher) {
        Collection result = new ArrayList();

        for ( element : this.treeElements ) {
            if ( matcher.matches(element) ) {
                result.add(element);
            }
        }
        return result;
    }
}

MyTree keywords = new MyTree();

//get all keywords starting with 'a'.
Collection matches = keywords.matches (
    new Matcher() {
        public boolean matches( String s ) {
            return s.startsWith("a");
        }
    }
);
```

```python
#Same thing in Python
class MyTree() :

    # details omitted

    def matches( self, matches ):
        return [el for el in self.__tree_elements if matches(el)]

keywords = MyTree()

# Code that fills the tree goes here
# Now, get all keywords starting with 'a'.
matches = keywords.matches( lambda el: el.startswith('a') )
```