OOP TA Session 4

Arrays

Command line arguments

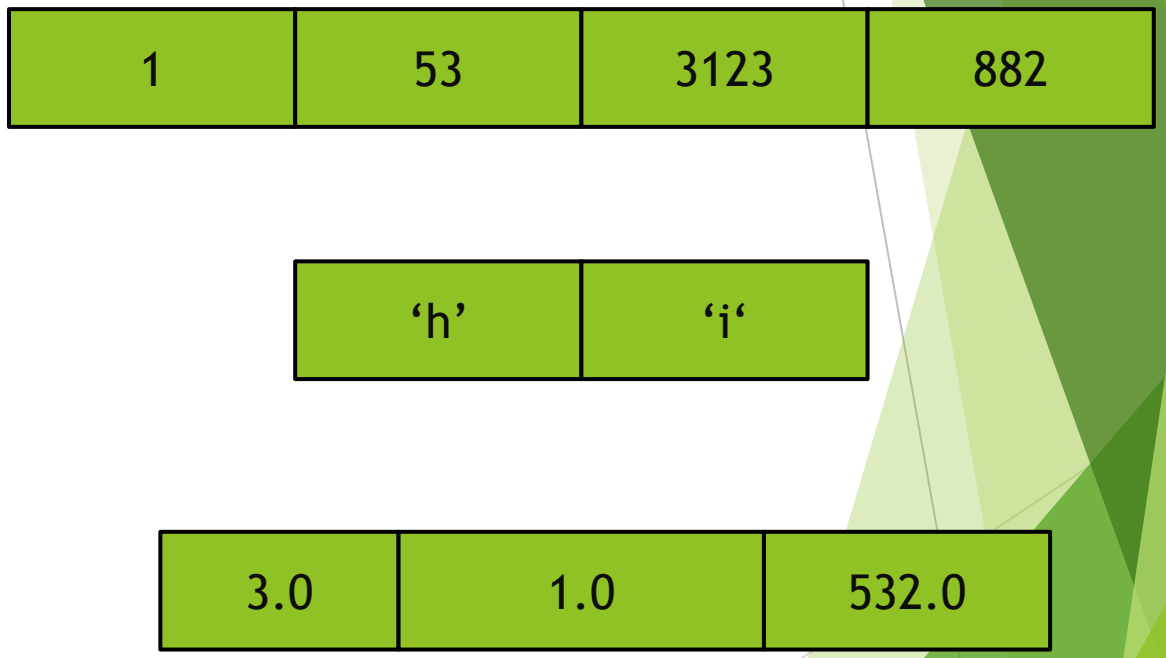
Inheritance

protected

Arrays ?

- Are like nothing in Python!

- A special object
- A collection of
  - A specific, single type
  - A specific size!



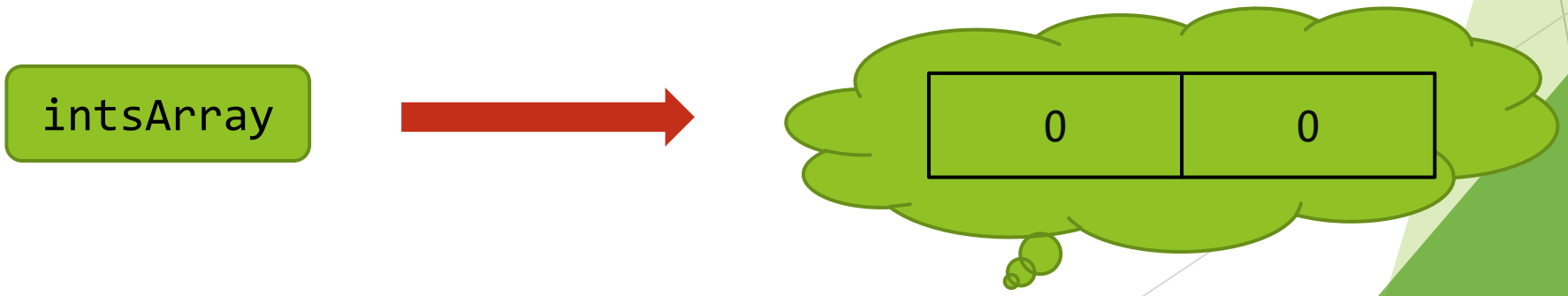
Arrays

A reference to an array of ints

An object type

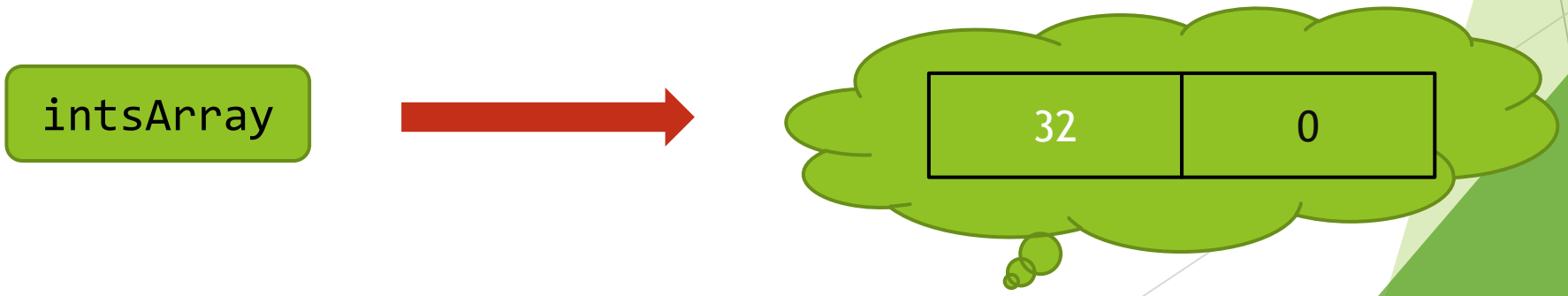
```
int[] intsArray;
intsArray = new int[2];
```

An array object of size 2



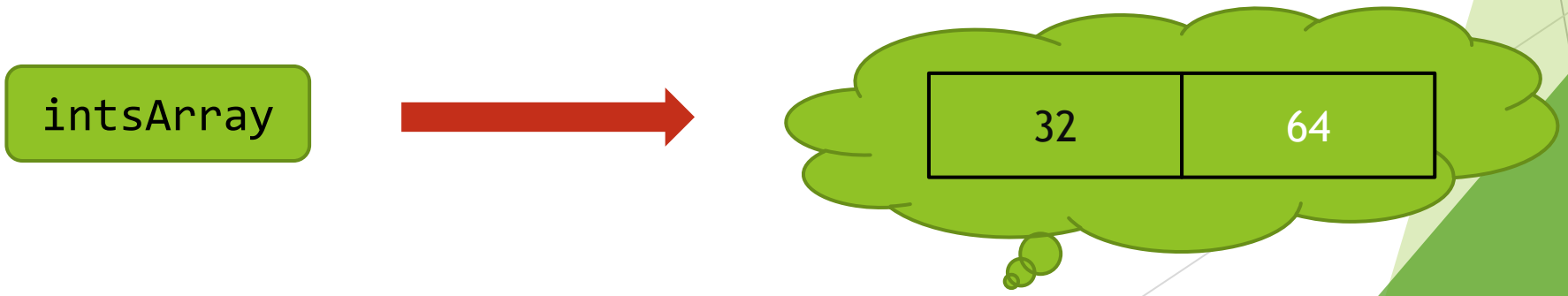
Arrays

```
int[] intsArray;
intsArray = new int[2];
intsArray[0] = 32;
```



Arrays

```
int[] intsArray;
intsArray = new int[2];
intsArray[0] = 32;
intsArray[1] = 64;
```



Arrays

```
int[] intsArray;
intsArray = new int[2];
intsArray[0] = 32;
intsArray[1] = 64;
intsArray[2] = 128;
```

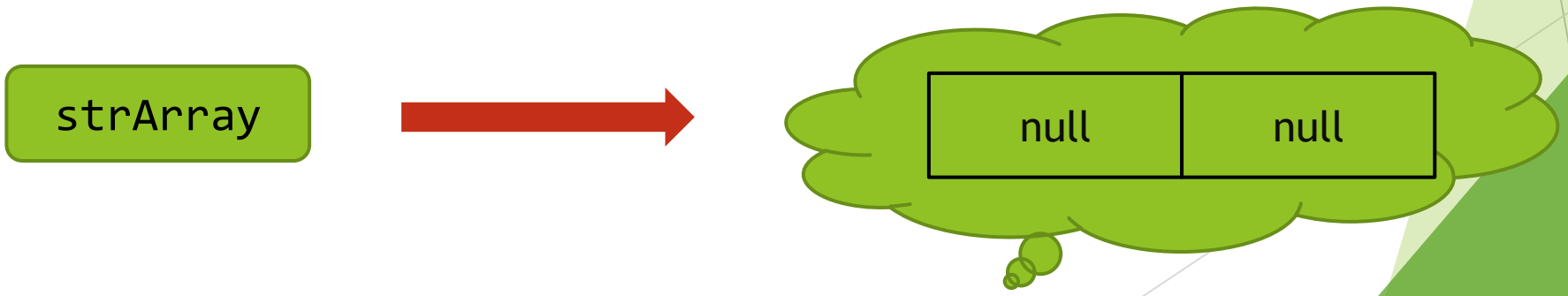
Arrays

```
int[] intsArray;
intsArray = new int[2];
intsArray[0] = 32;
intsArray[1] = 64;
intsArray[-1] = 128;
```

Arrays

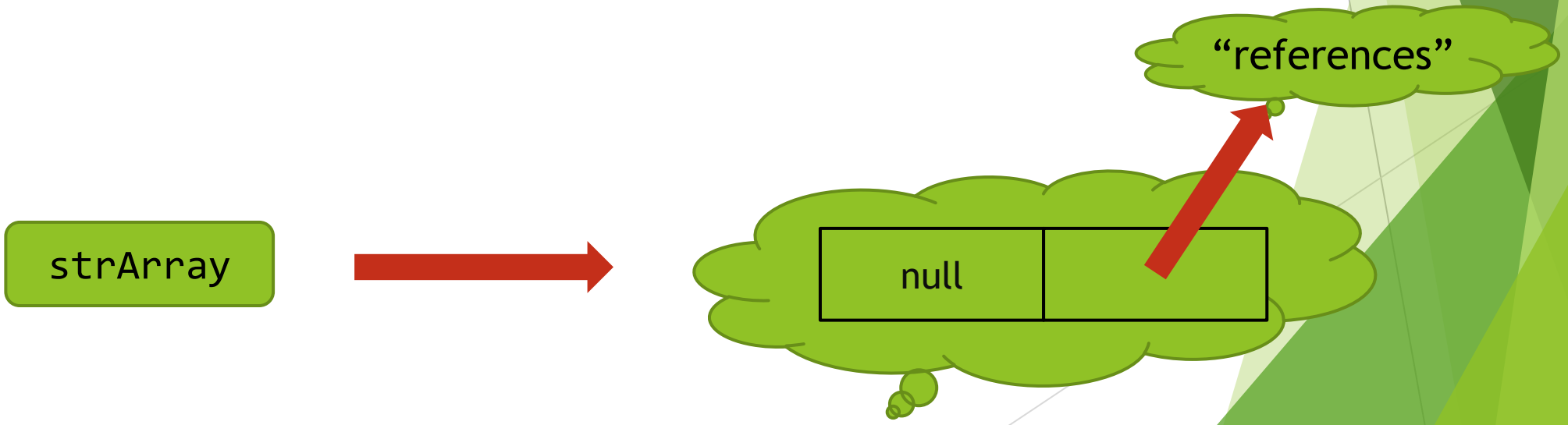
A reference to an array of references to String

```
String[] strArray;
strArray = new String[2];
```



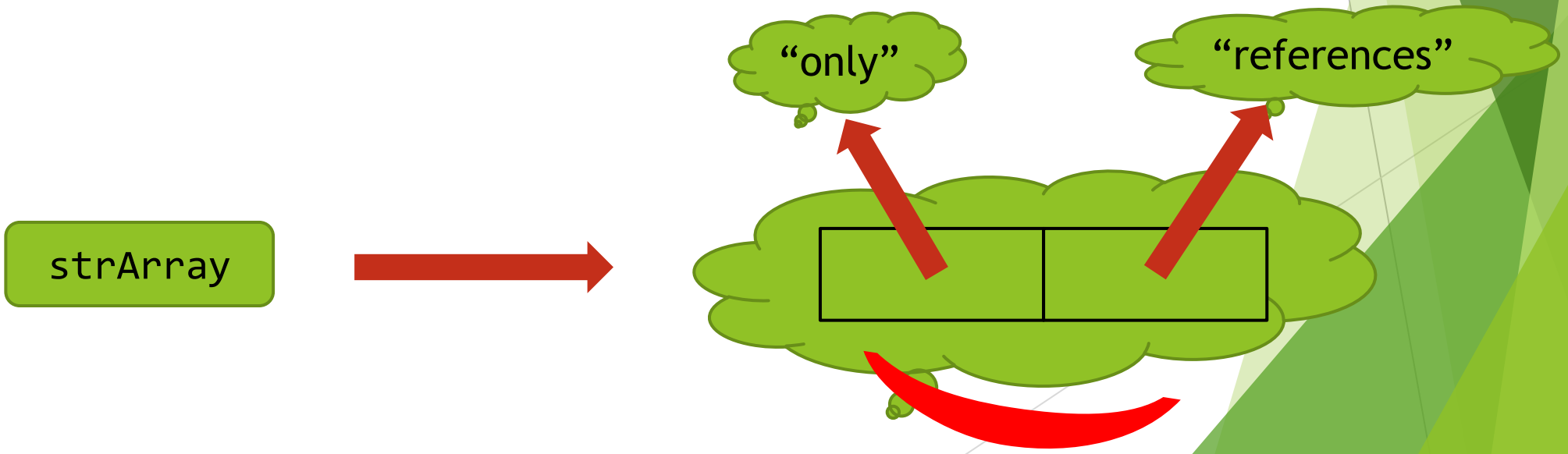
Arrays

```
String[] strArray;
strArray = new String[2];
strArray[1] = “references“;
```



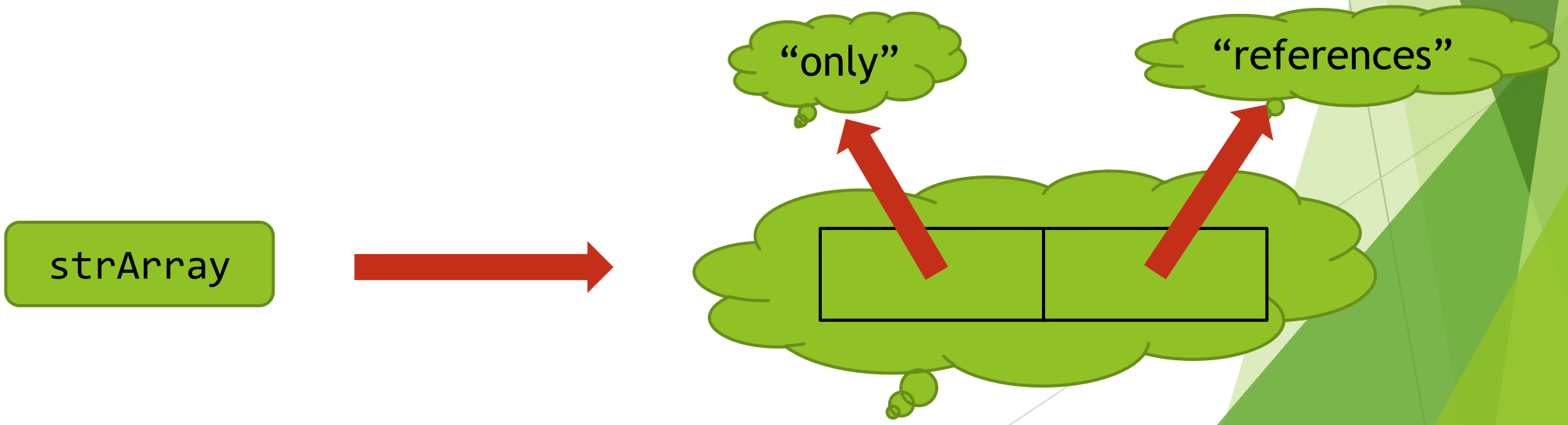
Arrays

```
String[] strArray;
strArray = new String[2];
strArray[1] = “references“;
strArray[0] = “only“;
```



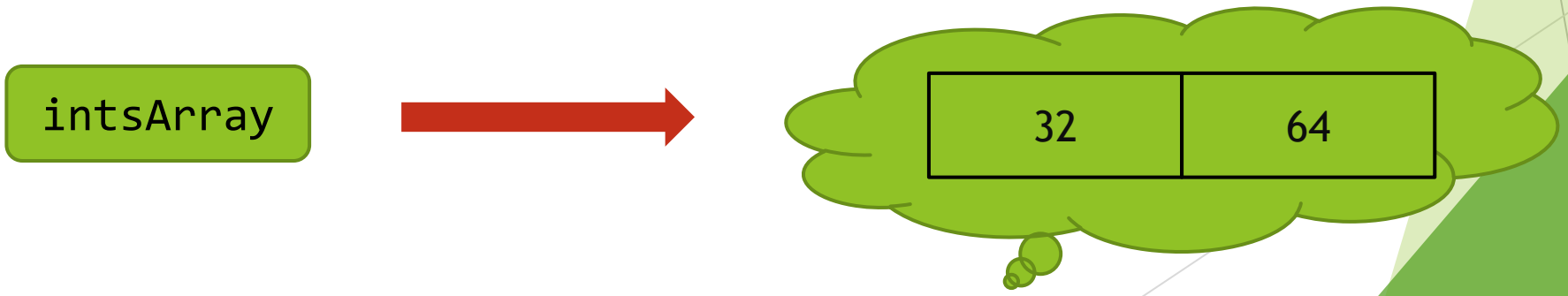
Arrays: shorthand for assignment

```
String[] strArray;
strArray = new String[]{“only“, “references“};
```



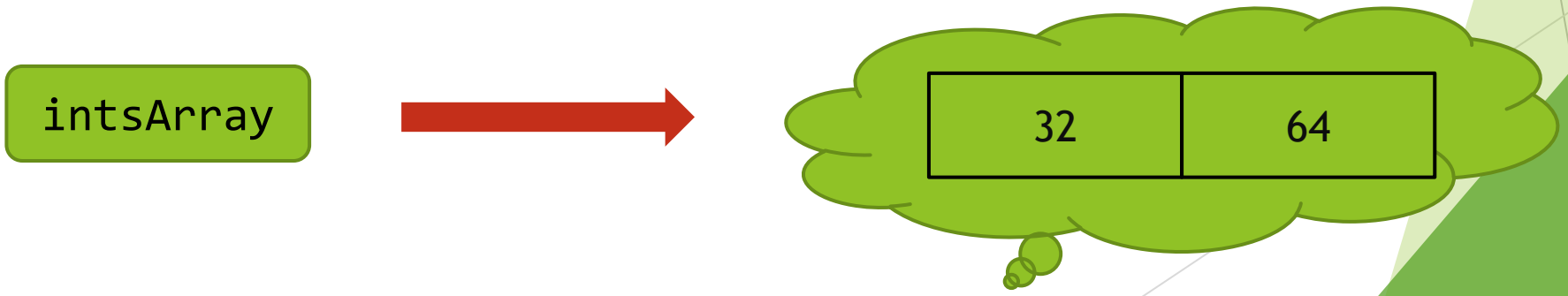
Arrays: shorthand for assignment

```
int[] intsArray;

intsArray = new int[]{32, 64};
```



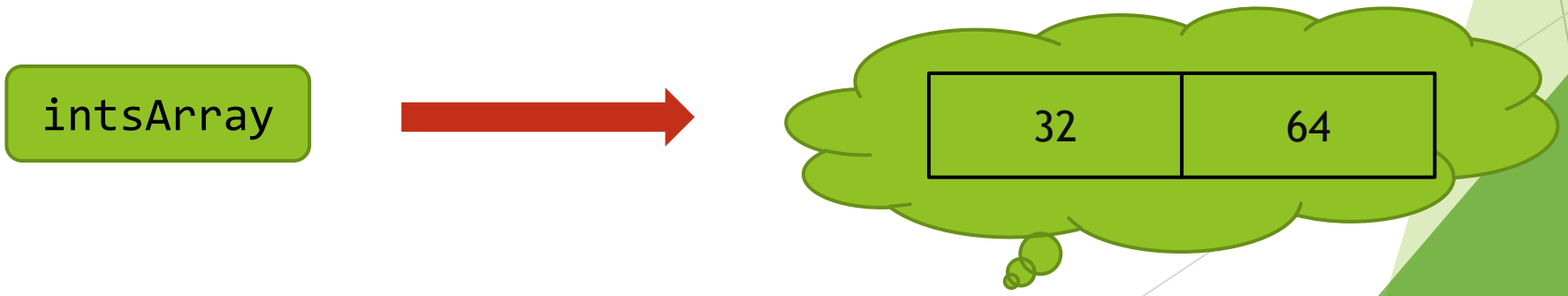
Arrays: shorthand for assignment

```
int[] intsArray = new int[]{32, 64};
```



Arrays: shorthand for INITIALIZATION

```
int[] intsArray = {32, 64};
```



Arrays: shorthand for INITIALIZATION

```
int[] intsArray = {32, 64};
intsArray = {128, 256};
```

Why not allowed?

Very sorry sweety.

Arrays

- Initialization:

```
int[] intsArray = {32, 64};
    OR
int[] intsArray = new int[]{32, 64};
```

- Assignment:

```
intsArray = new int[]{32, 64};
    OR
intsArray[2] = 4;
```

2D Arrays

- What about a matrix?
- An array of arrays:

```
int[][] intsArray = { {1,2}, {4,8}, {16,32,64} };
```

type

Array of type

Common Array Errors:

```
public class Test {
   public static void main(String[] args) {
    String[] words;
    int num = words.length;  //no array there yet!
    words[0]="hi";
    words = new String[15];
    num = words[0].length(); //no object there yet!
    words[15]="Hello"; //out of bounds!
   }
}
```

null !

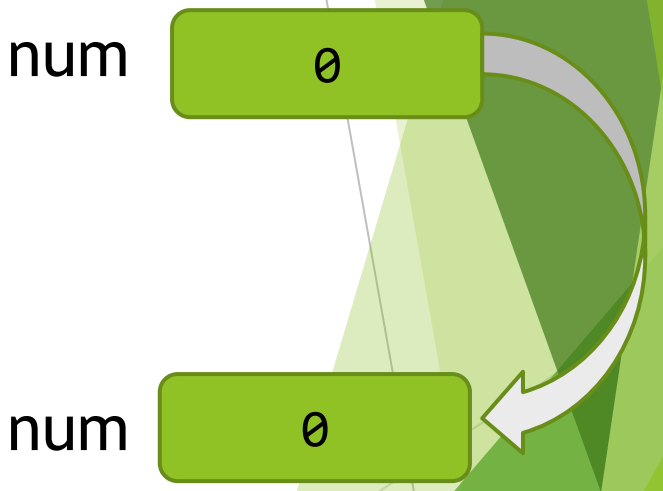
Won't compile

Exception in thread "main" java.lang.NullPointerException
at Test.main(Test.java:7)

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 15
at Test.main(Test.java:8)

Call by value

```
public static void main(String[] args){
   int num = 0;
   increment(num);
{

public static void increment(int num){
   num++;
{
```



Call by value

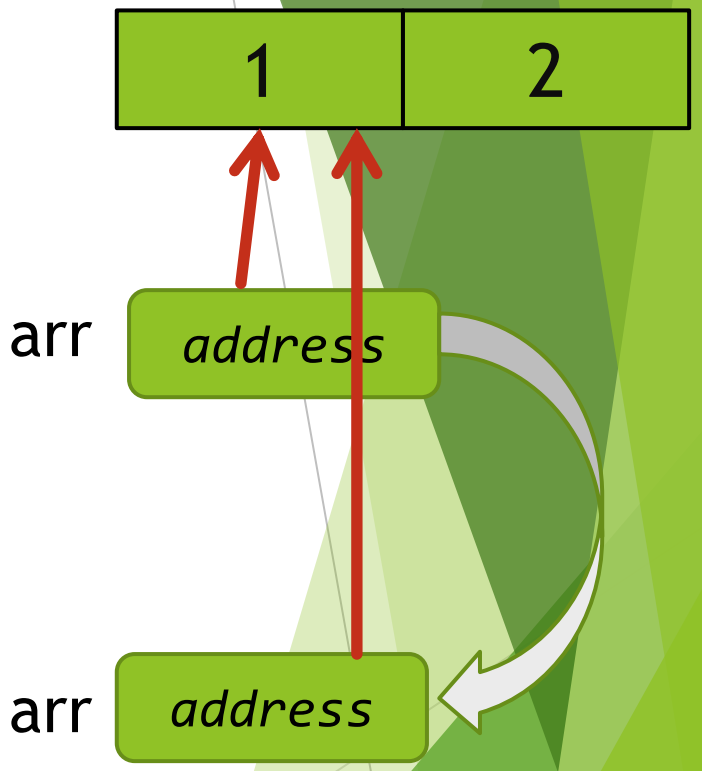
```
public static void main(String[] args){
   int num = 0;
   increment(num);
{

public static void increment(int num){
   num++;
{
```

num 0

num 1

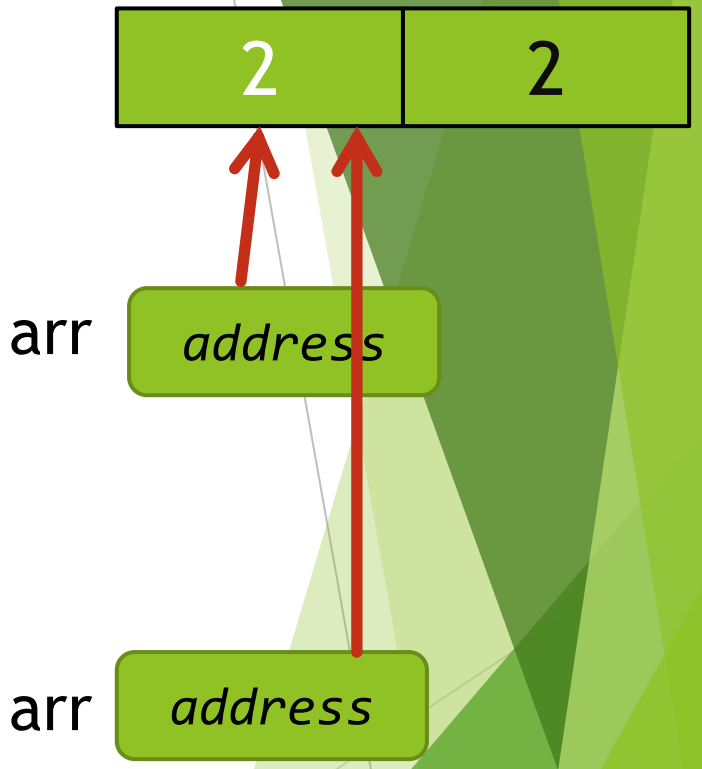
Call by reference

```
public static void main(String[] args){
   int[] arr = {1,2};
   increment(arr);
{

public static void increment(int[] arr){
   arr[0]++;
{
```

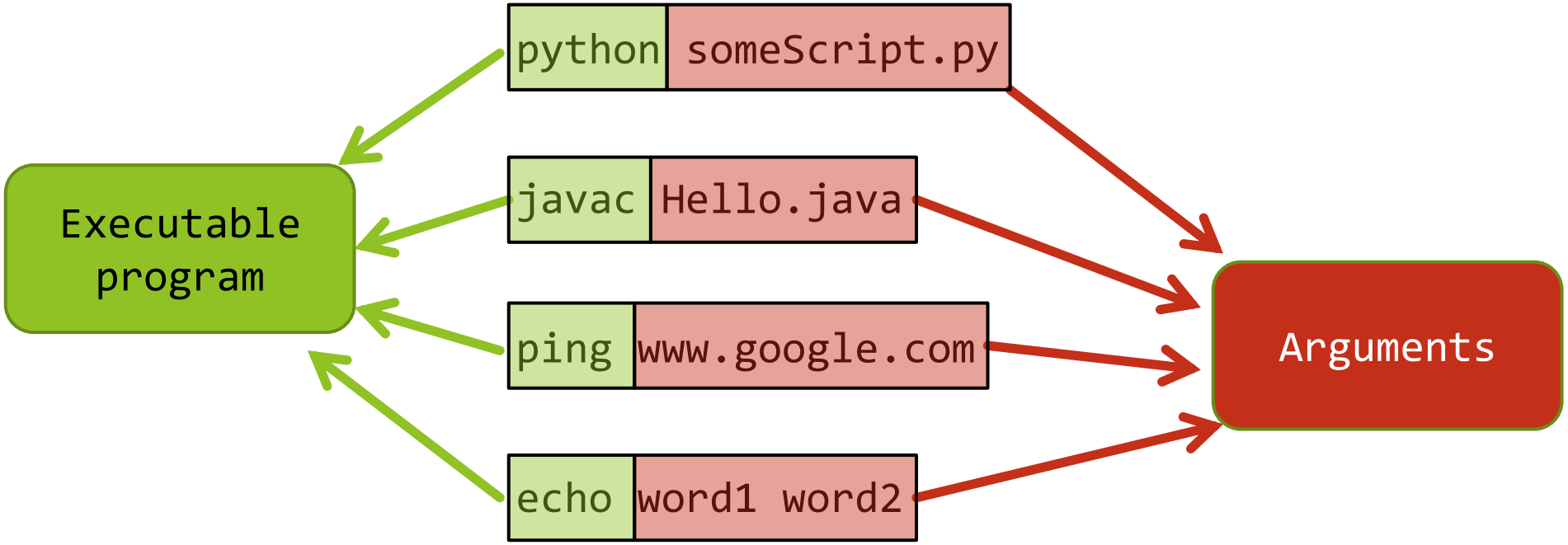


Call by reference

```
public static void main(String[] args){
   int[] arr = {1,2};
   increment(arr);
{

public static void increment(int[] arr){
   arr[0]++;
{
```



Command Line Arguments



Command Line Arguments

- Built in:

```
C:\Users\Rugrat>echo arg ument
arg ument

C:\Users\Rugrat>
```

- Let’s write:

```
>java Echo arg1 arg2
 arg1 arg2
```

Command Line Arguments

```
>java Echo arg1 arg2
 arg1 arg2
```

```java
public class Echo {
    public static void main(String[] args){
        for(String arg : args)
            System.out.print(arg + " ");
        System.out.println();
    }
}
```

Command Line Arguments

```
public class Echo {
    public static void main(String[] args){
        for(String arg : args)
            System.out.print(arg + " ");
        System.out.println();
    }
}
```

Protected

- Between private and public
- The rarer of the three

- Not part of the class's API
- But not internal implementation
- "An interface with the extender"

Protected: Example 1

- Save code for extender

```
public class Mammal {
    //walk, eat...
    public void die() {
        lieOnBack();
        shutdownSystems();
    }
    protected void lieOnBack() {
        // ..
    }
    protected void shutdownSystems() {
        // ..
    }
}
```

Protected: Example 1

- Save code for extender

```
public class Opossum extends Mammal {
    public void playDead() {
        lieOnBack(); // implemented in Mammal
        stickTangueOut();
    }
}
```

Protected: Example 2

- Class is not “stand alone”

```
public class Ant {
    private int id;

    protected Ant(int id) {
        this.id = id;
    }
    // march, attack...
}
```

```java
public class Queen extends Ant {
    private static final int NEST_SIZE = 5000;

    private int nextBabyId = 1;
    private Ant[] nest;

    public Queen() {
        nest = new Ant[NEST_SIZE];
        nest[0] = this;
    }

}
```

```java
public class Queen extends Ant {
    private static final int NEST_SIZE = 5000;

    private int nextBabyId = 1;
    private Ant[] nest;

    public Queen() {
        super(0);
        nest = new Ant[NEST_SIZE];
        nest[0] = this;
    }

    public boolean giveBirth() {
        if(nextBabyId == NEST_SIZE)
            return false;
        nest[nextBabyId] = new Ant(nextBabyId);
        nextBabyId++;
        return true;
    }
}
```

The only way to create regular Ants

Ex 2: LaTeX