GET SMART: WITH JAVA PROGRAMMING

Yaman Omar Alashqar

System.out.println(“WELCOME TO THIS COURSE\n”);

GET SMART: WITH JAVA PROGRAMMING

Arrays

What is a data structure?

Nested For Loop

For Each Loop

OPENING PROBLEM

Read the grades for one hundred students, then compute their average, and find out how many grades are above the average, then if the average is less than 60% add 2 marks to each grade.

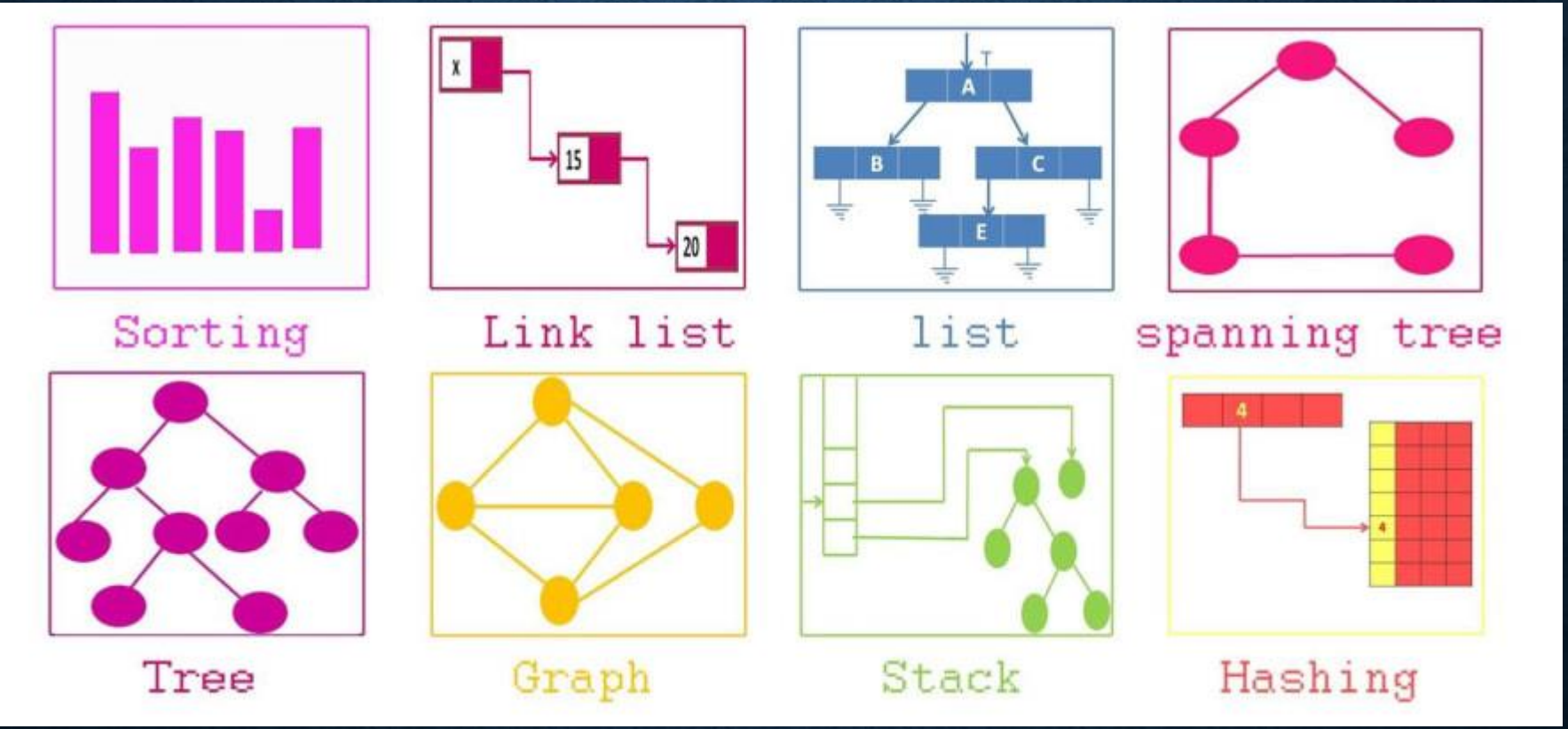
WE WILL NEED A:

data structure that represents a collection of the same types of data

DATA STRUCTURE

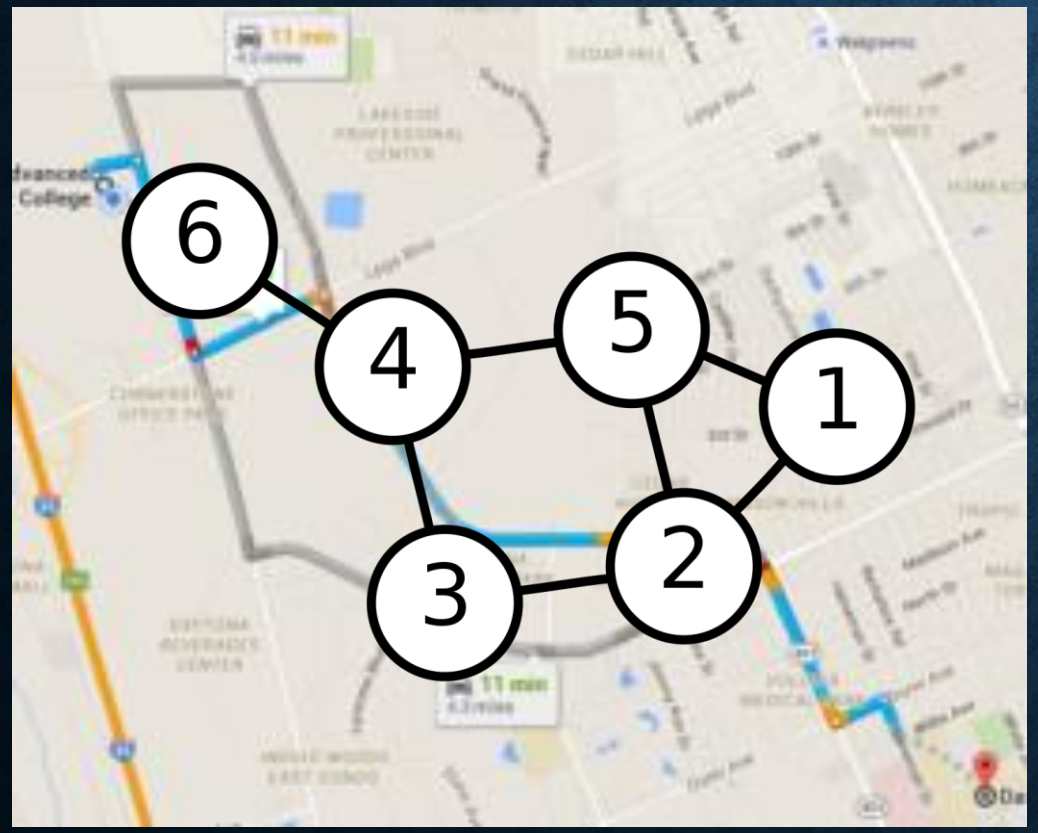
a way to store and organize **data** so that it can be used efficiently

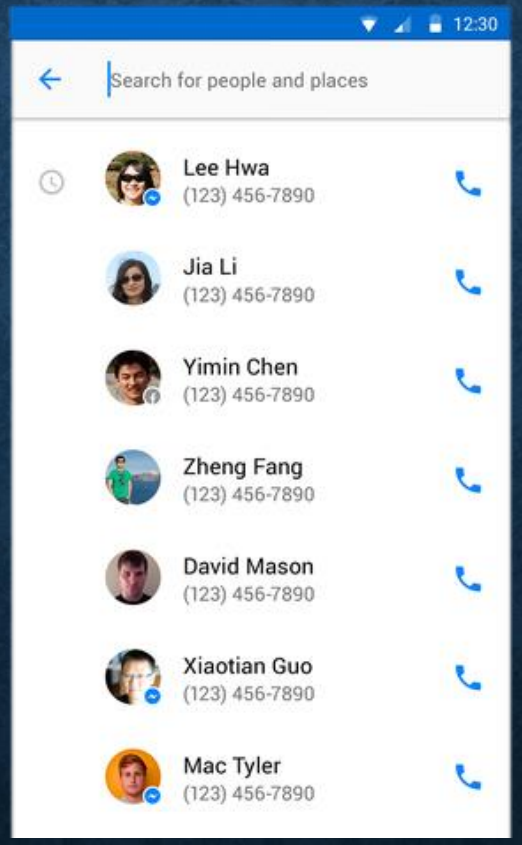
DATA STRUCTURE...

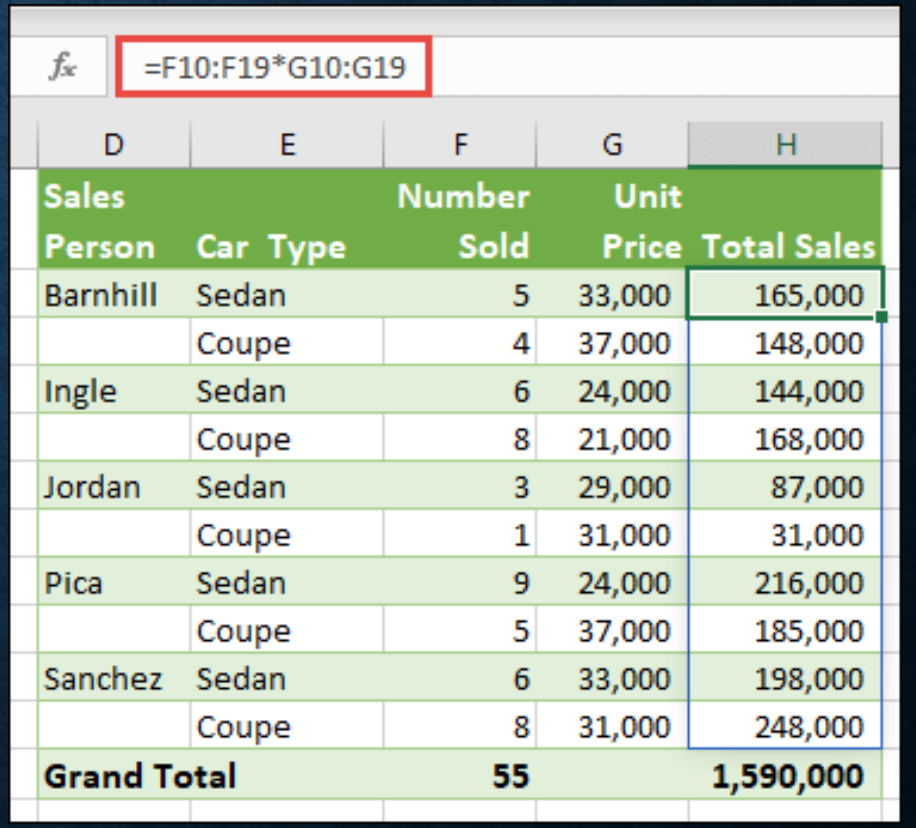


DATA STRUCTURE… WHY?

EXAMPLES







=F10:F19*G10:G19

	D	E	F	G	H
	Sales Person	Car Type	Number Sold	Unit Price	Total Sales
	Barnhill	Sedan	5	33,000	165,000
		Coupe	4	37,000	148,000
	Ingle	Sedan	6	24,000	144,000
		Coupe	8	21,000	168,000
	Jordan	Sedan	3	29,000	87,000
		Coupe	1	31,000	31,000
	Pica	Sedan	9	24,000	216,000
		Coupe	5	37,000	185,000
	Sanchez	Sedan	6	33,000	198,000
		Coupe	8	31,000	248,000
	Grand Total		**55**		**1,590,000**

DATA STRUCTURES

A data structure is a data organization, management, and storage format that enables efficient access and modification.

More precisely, a data structure is a collection of data values, the relationships among them, and the functions or operations that can be applied to the data.

BACK TO THE OPENING PROBLEM

Read the grades for one hundred students, then compute their average, and find out how many grades are above the average, then if the average is less than 60% add 2 marks to each grade.

```
int grade1;
int grade2;
int grade3;
int grade4;
int grade5;
.
.
int grade100;
```

BACK TO THE OPENING PROBLEM

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value

ARRAYS

Ordered collections of values.

- ❑ List of comments on a post
- ❑ Collection of levels in a game
- ❑ Movies in a playlist

ARRAYS

- How to declare an array?
- How to insert values to an array?
- How to access the elements of an array?
- How to change an array element?
- Loop through an array, for each, methods …

ARRAYS

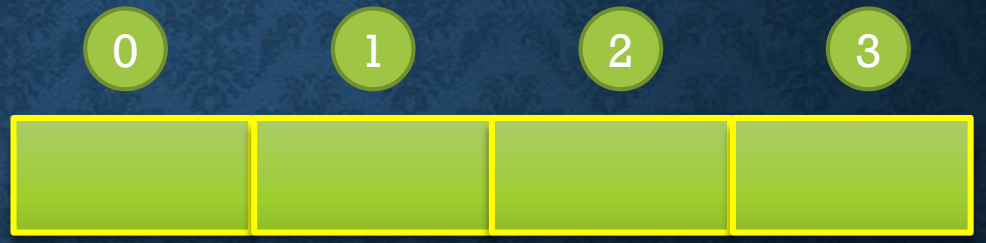
- To declare an array, define the variable type with **square brackets**:
- String[] cars = new String[SIZE];
- We have now declared a variable that holds an array of strings.

- To declare an array and insert values to it right away, place the values in a comma-separated list, inside curly braces:
- String[] cars = {" Mercedes", "BMW", "Ford", "Mazda"};

ARRAYS

```
final int SIZE = 4;

String[] cars = new String[SIZE];
```

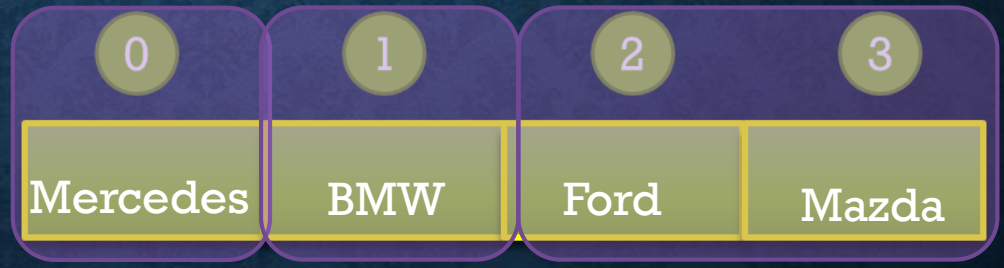



```
cars[0] = "Mercedes";

cars[1] = "BMW";

cars[2] = "Ford";

cars[3] = "Mazda";
```



ARRAYS

- Java Arrays are index based collection.
- The default values of the array:
  - 0 in the case of primitives,
  - null in the case of Objects,
  - false in the case of boolean.
- Arrays are fixed length in size.
- The main advantage of an array is we can represent multiple values under the same name.
  - So that readability of code is improved.

PROCESSING ARRAYS

```
final int SIZE = 4;
String[] cars = new String[SIZE];
    cars[0] = "Mercedes";
    cars[1] = "BMW";
    cars[2] = "Ford";
    cars[3] = "Mazda";
```

- The toString() method returns a string representation of the contents of the specified array.
- This method is helpful in debugging.

```
System.out.println(Arrays.toString(cars));
```

```
run:
[Mercedes, BMW, Ford, Mazda]
BUILD SUCCESSFUL (total time:
```

PROCESSING ARRAYS

```
final int SIZE = 4;
String[] cars = new String[SIZE];
    cars[0] = "Mercedes";
    cars[1] = "BMW";
    cars[2] = "Ford";
    cars[3] = "Mazda";
```

```
for(int i = 0 ; i< cars.length ; i++)
    {
    System.out.println("cars[" + i + "] = " + cars[i]);
     }
```

```
run:
cars[0] = Mercedes
cars[1] = BMW
cars[2] = Ford
cars[3] = Mazda
BUILD SUCCESSFUL (to
```

PROCESSING ARRAYS

- Summation of an array
- Printing an array
- Finding the largest element in an array

ARRAYS METHODS

- toString()
- sort()
- binarySearch()
- copyOf()
- equals()

ARRAYS METHODS

```
int[] a = {5, 2, 4, 3, 1};
        System.out.println(Arrays.toString(a));
        Arrays.sort(a);
        System.out.println(Arrays.toString(a));
```

ARRAYS METHODS

```
String[] planets = { "Mercury", "Venus", "Mars", "Earth", "Jupiter",
        "Saturn", "Uranus", "Neptune", "Pluto"  };
    System.out.println(Arrays.toString(planets));
    Arrays.sort(planets); //SHOULD BE SORTED FIRST
    System.out.println(Arrays.toString(planets));
    System.out.println("Find Mars: " + Arrays.binarySearch(planets, "Mars"));
```

ARRAYS METHODS

```
int[] a = {5, 2, 4, 3, 1};
int[] b = null;
        System.out.println("A= " + Arrays.toString(a));
        System.out.println("B= " + Arrays.toString(b));
b = Arrays.copyOf(a, a.length);
     System.out.println("A= " + Arrays.toString(a));
        System.out.println("B= " + Arrays.toString(b));
```

THE FOR EACH LOOPS

- foreach loop or enhanced for loop,
- enables you to traverse the complete array sequentially without an index variable.
- For-each loops are not appropriate when you want to modify the array
- For-each only iterates forward over the array in single steps
- for(String e : cars) System.out.println("e = " + e);

```
e = Mercedes
e = BMW
e = Ford
e = Mazda
BUILD SUCCESSFU
```

NESTED LOOP IN JAVA

- If a loop exists inside the body of another loop, it's called a nested loop.

```
1*1=1 || 1*2=2 || 1*3=3 || 1*4=4 || 1*5=5 || 1*6=6 ||
2*1=2 || 2*2=4 || 2*3=6 || 2*4=8 || 2*5=10 || 2*6=12 ||
3*1=3 || 3*2=6 || 3*3=9 || 3*4=12 || 3*5=15 || 3*6=18 ||
4*1=4 || 4*2=8 || 4*3=12 || 4*4=16 || 4*5=20 || 4*6=24 ||
5*1=5 || 5*2=10 || 5*3=15 || 5*4=20 || 5*5=25 || 5*6=30 ||
6*1=6 || 6*2=12 || 6*3=18 || 6*4=24 || 6*5=30 || 6*6=36 ||
```

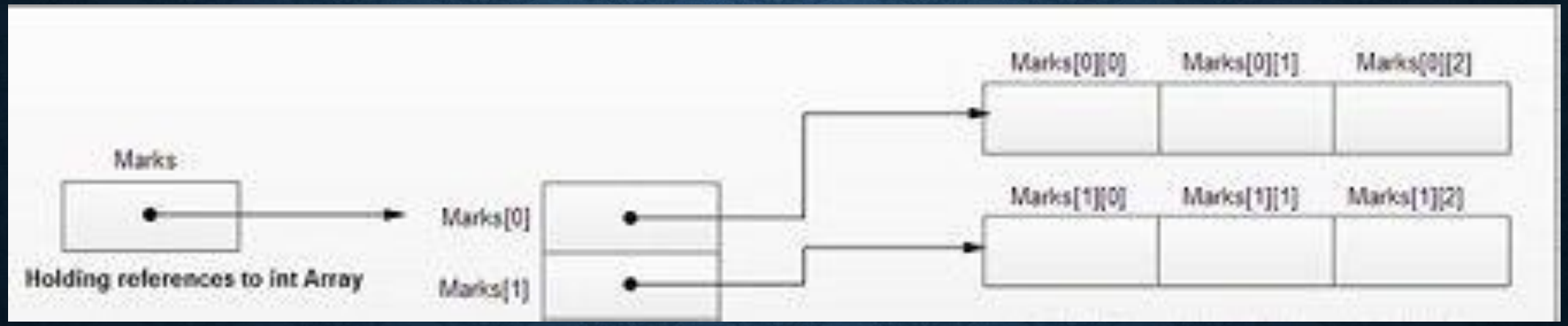
```
for(int i = 1 ; i<=6 ; i++) {

        for(int j = 1 ; j <= 6 ; j++) {
           System.out.print(i + "*" + j + "=" + i*j + " || ");
        }

        System.out.println("");
    }
```

MULTIDIMENSIONAL ARRAYS

- A multidimensional array is an array containing one or more arrays.



- int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
  - int x = myNumbers[1][2];
  - System.out.println(x); // Outputs 7

Exercise 1

One large chemical company pays its sales people on a commission basis. The salespeople receive $200 per week plus 9% of their gross sales for that week.

For example, a salesperson who sells $5000 worth of chemicals in a week receives $200 plus 9% of $5000, or a total of $650.

Develop a JAVA program that will input each salesperson's gross sales for last week and will calculate and display that salesperson's earnings.

NOTE: Process one salesperson's figures at a time

A company has 10 departments and each department has N employees. Write a program that reads for each department the number of employees in the department (N) followed by the employee salaries. The program should read this information for all 10 departments and compute and print the following:

- The average salary in each department.
- The average salary in the whole company.
- The number of employees in the company whose salaries are between 600 JD and 1400 JD inclusive.

```
(rand.nextInt((max - min) + 1) + min)
```

Restaurant System

Display the following statistics for a Restaurant:

- The total number of served Soft Drinks
- The total number of served Sandwiches
- The total sales for the restaurant
- The average cost per order
- The maximum and minimum order cost
- Number of orders with cost above average
- The number of orders with no Soft Drinks
- Number of orders
- Send an auto E-mail on Wednesday

```
4  int number = 373;
5  int lastDigit , tempNumber = number;
6  int reverse=0 , armstrong=0 , counter=0;
7
8  while (tempNumber!=0)
9      {
10     lastDigit = tempNumber % 10;
11
12     reverse = reverse * 10 + lastDigit;
13
14     armstrong += (lastDigit*lastDigit*lastDigit);
15
16     tempNumber = tempNumber / 10;
17
18     counter++;
19     }
```

This code solves the palindrome & Armstrong number problem (note that the Armstrong only works for 3 digits), also counts the number of digits …

In our example, the number is ‘373’

Note that I stored the original number in a temp variable so I could compare the original number to other numbers later…

‘Line 10’ we separated the last digit from the right (373 % 10 = 3)

‘Line 12’ is used to store the original number reversed (e.g. 123 becomes 321)

‘Line 14’ is used to store the cube of each digit and sum it up …

(e.g. 153 is a Armstrong because
((1*1*1) + (5*5*5) + (3*3*3) = 153))

‘Line 16’ chops off the last digit
(373 / 10 = 37)

‘Line 17’ counts how many time the number chopped off a digit…