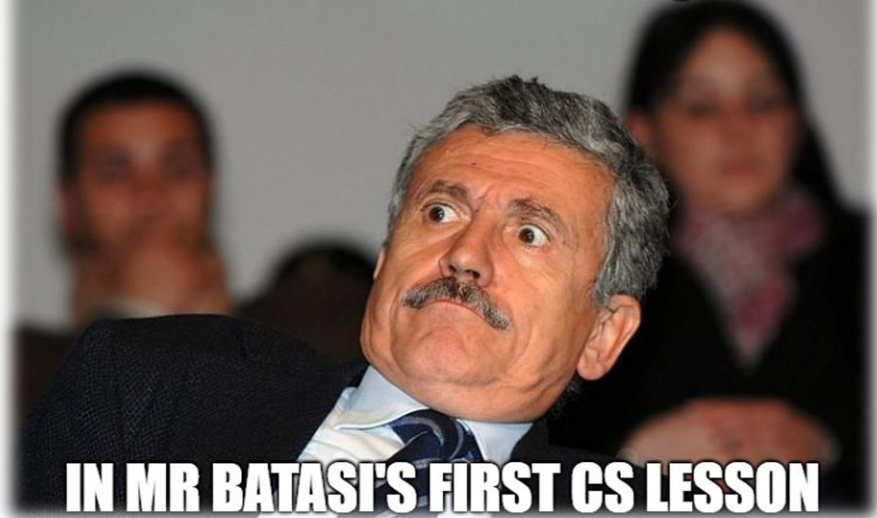


Guide to Pseudocode

By Mr Batasi

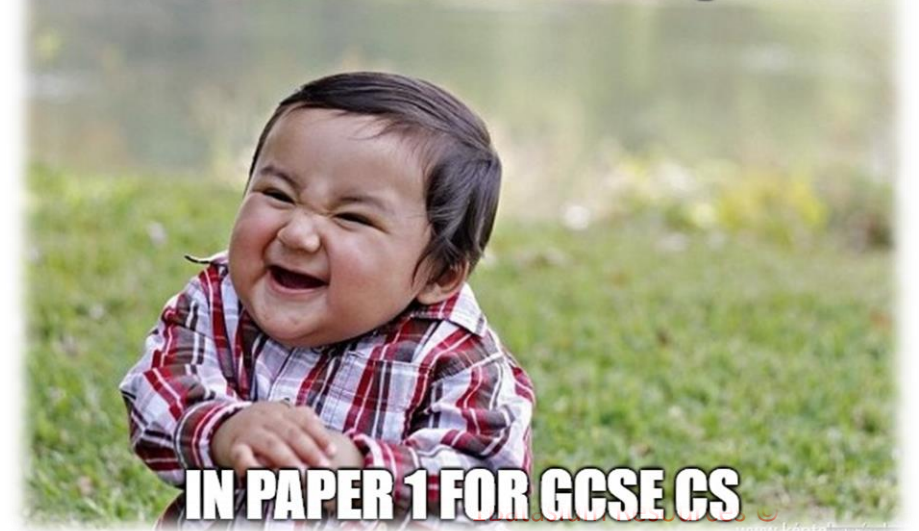
GO FROM THIS

WHEN YOU SEE A PSEUDOCODE QUESTION



TO THIS

WHEN YOU SEE A PSEUDOCODE QUESTION



What is Pseudocode?

- We use programming languages (high level) to create programs.
- Examples of programming languages
 - Python
 - Java
 - C++
- However, **pseudocode** is a generic form of creating programs (not specific to any language)

Pseudocode: **Output** commands

- In python we use the *print* function to display values to the user
- However, in pseudocode we can still use **print** the commands **output** or **display** to do the same thing.

Python	Pseudocode
<pre>print("My name is Mr Batasi")</pre>	<pre>print("My name is Mr Batasi") output "My name is Mr Batasi"</pre>

Pseudocode: **Input** commands

- In python we use the *input* function to ask the **user** to enter values (NOTE: these **MUST** be stored in an *appropriately named variable*)
- In pseudocode, we use the command **INPUT** to do the same thing (no need for data type e.g. str or int)

Python	Pseudocode
<pre>name = str(input()) score = int(input())</pre>	<pre>input name OR name = input input score OR score = input</pre>

Pseudocode: Input and Output

- We can use both input and output commands together to ask the user for *values (which are stored in variables)* and then display these as *variables*.

Python	Pseudocode
<pre>score = int(input()) print("Your score is", score)</pre>	<pre>score = input("Enter score") display "Your score is", score OR score = input("Enter score") output "Your score is", score</pre>

Learning Review 1

- The user is asked to enter their name. It is then displayed to the user with an appropriate message.

```
name = input("Enter name: ")    #(entering part of question)
output "Your name is", name     #(displaying part of question)
```

Task 1

- 1. The user is prompted to enter their age. This age is then shown to the user
- 2. Ask the user to enter their username and password. These are then displayed to the user

Pseudocode: Validation (Selection)

- In programming, a key **control structure** is selection (allowing the program to choose *different* paths based on a *specific* value)
- In python, we use the *if* command to check values and then direct the program in different directions.
- Similarly, we can also use the IF command in pseudocode without **any** changes (except to the *condition*)

Python	Pseudocode
<pre>if score >= 50: print("You have passed")</pre>	<pre>if score >= 50 then output "You have passed" endif</pre>

Pseudocode: Validation (Selection) continued...

- Sometimes we need more than one condition (if statement) based on the scenario given to us
- In python we use `if, elif, elif...else` (*elif = else if*)
- When writing this in pseudocode, it is very similar (except we have to write *elseif* instead of `elif`)
- See next page for example

Pseudocode: Validation (Selection) continued...Example

Python	Pseudocode
<pre>if score < 40: print("Grade F") elif score >= 40 and score < 50: print("Grade C") elif score >= 50 and score < 60: print("Grade B") elif score >= 60 and score < 70: print("Grade A") else: print("Grade A*")</pre>	<pre>if score < 40 then output "Grade F" else if score >= 40 AND score < 50 then output "Grade C" else if score >= 50 AND score < 60 then output "Grade B" else if score >= 60 AND score < 70 then output "Grade A" else print("Grade A*") endif</pre>

Learning Review 2

- A program asks the user to enter the top speed of a car. If the speed is between 1 and 100, the program outputs 'Slow car'. If the car's speed is more than 100 mph, then the program outputs 'Fast car'. Otherwise, the algorithm outputs 'Invalid input'.

```
speed = input("Enter top speed")
if speed >=1 and speed <=100 then
    output "Slow car"
else if speed > 100 then
    output "Fast car"
else
    output "Invalid input"
endif
```

```
#enter top speed
#check speed between 1 and 100
#program outputs "Slow car"
#check if speed is more than 100
#program outputs "Fast car"
#otherwise
#program outputs "Invalid input"
```

Task 2

- 1. The user is prompted to enter their age. If their age is less than 16, then “You cannot drive” is shown. Otherwise, “You can drive” is displayed.
- 2. The user is prompted to enter two passwords. If both passwords match, a message stating “passwords match” is displayed. Otherwise the message “please try again” is displayed
- 3. A program asks the user to enter their height. If their height is less than 100cm, they *cannot enter* the ride. If their height is between 100cm and 140cm, they *need an adult* to go onto the ride. Otherwise, they can enter the ride *unsupervised*.

Pseudocode: **Repeating** instructions (Iteration)

- In programming, we often need to repeat some instructions a specific number of times OR until a condition is met.
- This involves the use of **for** loops and **while** loops within python.
- **for** loops – repeating a **set** number of times
(e.g. 5, length of “hello”)
- **while** loops – repeat continuously **until** the condition *does not meet*
(e.g. until value is > 0)

Pseudocode: **Repeating** instructions (Iteration) – **for** loops

For loops need to have a ***starting*** point and ***ending*** point as it repeats/iterates a fixed number of times:

From ***starting*** (e.g. 1) To ***ending*** (e.g. 10) so this is 10 repetitions

Python	Pseudocode
<pre>number = 0 for i in range(0,10): number=number+2</pre>	<pre>number ← 0 for i = 1 to 10 number ← number + 2 next i endfor</pre> <p>(Note: i starts from 1 because only python indexing starts at 0)</p> <p>Final value of <u>number = 20</u></p>

Learning Review 3

- A program asks the user how many *total* times they would like to deposit money into their account (the initial value is 0). The program continuously deposits money to the account, based on their *total* input. Finally, the program displays the account value.

```
account = 0
total = input
for i = 1 to total
    money = input
    account = account + money
next i
endfor
output account
```

#account initial value is set to 0

#ask the total times they need to add money

#for loop starts at 1, ends at **total** value (e.g. 5)

#ask the money for their deposit

#add deposit value to the **account**

#display the account value

Task 3

- 1. Ask the user to enter a number. The program displays the times table of that number (up to $\times 10$).
 - E.g. – if 7 is entered, the program displays **$7 \times 1 = 7$** up to **$7 \times 10 = 70$**
- 2. The user is prompted to enter the total number of pages in a book. For every page, the user must add 20 seconds reading time. The total reading time is displayed at the end.

Pseudocode: Repeating instructions (Iteration) – **while** loops

While loops have a repeating condition(s) that continues *FOREVER*
(until the condition(s) **does not match**)



Python	Pseudocode
<pre>totalMoney = 0 while totalMoney < 100: money = int(input("Enter money")) totalMoney = totalMoney + money print("Your final amount is", totalMoney)</pre>	<pre>totalMoney = 0 while totalMoney < 100 money = input totalMoney = totalMoney + money endwhile output "Your final amount is ", totalMoney</pre>

Pseudocode: Repeating instructions (Iteration) – **while** loops continued...

Furthermore, you can use **while** loops similar to **for** loops...

But you **MUST manually** increase the *counter* variable

That's why for **fixed** repetitions, **for** loops are **more** efficient to use

Python	Pseudocode
<pre>mySum = 0 counter = 0 while counter != 10: mySum = mySum + 2 counter = counter + 1 mySum = 0 for counter in range(0,10): mySum = mySum + 2</pre>	<div><pre>mySum = 0 counter = 0 while counter ≠ 10 mySum = mySum + 2 counter = counter + 1 endwhile</pre> #manually</div> <div><pre>mySum = 0 for counter = 1 to 10 mySum = mySum + 2 next i endfor</pre></div>

Learning Review 4

- The user is asked to enter their *height* (in cm). The program does not continue until the height entered is greater than (or equal to) 50. The height is then converted into *inches* by multiplying by 0.4, and the *inches* are then displayed to the user.

```
height_cm = input
while height_cm < 50
    height_cm = input
height_inches = height_cm x 0.4
endwhile
print(height_inches)
```

#enter the height
#do not continue until height meets
condition of 50 or more,
otherwise repeat above point
#calculate new height
#display the new height

Task 4

- 1. The user is asked to enter two *usernames*. The program displays 'Usernames match' if they are the same, otherwise it repeatedly asks the user to re-enter the *usernames* until they match.
- 2. A program asks the user to enter a value for the *miles*. The program only continues if the value entered is greater than zero. The program then calculates the *kilometres* by dividing the miles by 1.6. The value for the *kilometres* is then displayed to the user.

Exam Question 1*

Figure 5

```
OUTPUT 'enter the 24 hour number (0-23) '  
hour ← USERINPUT
```

The algorithm in **Figure 5** asks the user to enter a number between 0 and 23 that represents an hour using the 24 hour clock. The input is stored in a variable called `hour`.

Extend the algorithm in **Figure 5**, using either pseudo-code or a flowchart, so that it outputs the equivalent time using the 12 hour clock, ie a number between 1 and 12, followed by either `am` or `pm`.

For example:

- If the user enters 0, the program outputs 12 followed by `am`.
- If the user enters 4, the program outputs 4 followed by `am`.
- If the user enters 12, the program outputs 12 followed by `pm`.
- If the user enters 15, the program outputs 3 followed by `pm`.

You can assume that the variable `hour` is an integer and that the value that the user inputs will be between 0 and 23.

[7 marks]

Pseudocode: **Arrays** and **Lists** (Data Structures)

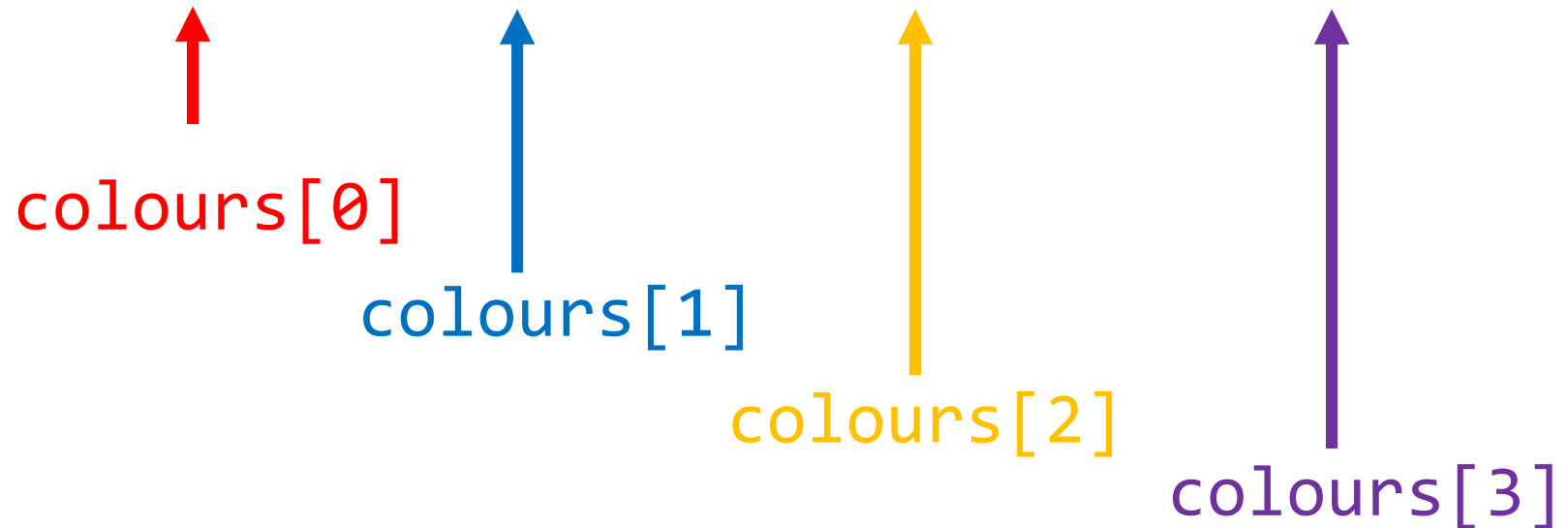
- Data structures are ***sets of organised data, which can be easily edited and changed.***
- We use the **[]** syntax to create data structures (in python & pseudocode)
- Instead of using several/multiple variables we can use data structures because they are more **memory efficient** and can allow us to store **ordered data**
- ***Arrays, Records, and Lists are examples of data structures but exams will mainly use Arrays***

Pseudocode: **Arrays** and **Lists** (Data Structures) continued...

- When using data structures, we refer to each position of the list with an **index**
- `colours = ["red", "blue", "orange", "lilac"]`
- Index starting at ZERO
- If we write `colours[2]` in python, we are pointing to the **index value 2** which is the **3rd value** in the **list** → "orange"
- Index starting at ONE
- If we write `colours[2]` in pseudocode, we are pointing to the **index value 2** which is the **2nd value** in the **list** → "blue"
- ***Exams - the indexing will always start from 0 (ZERO)***

Pseudocode: **Arrays** and **Lists** (Data Structures) continued...

- Example – Indexing starting at 0
- `colours = ["red", "blue", "orange", "lilac"]`



Pseudocode: **Arrays** and **Lists** (Data Structures) continued...

- Data structures allow us to perform various mathematical operations such as finding the **lowest**, **highest**, **average** and **total** in a set of data.

Python	Pseudocode
<pre>scores = [4, 10, 3, 15, 9] total = 0 for x in range(len(scores)): total = total + scores[x] print("The total score is", total)</pre> <p>Note: len(scores) = <u>length</u> of the scores list which is 5 scores[x] is used because x changes in the loop from 1 to the length of the list (5)</p>	<pre>scores = [4, 10, 3, 15, 9] total = 0 for x = 1 to scores.length total = total + scores[x] next i endfor display "The total score is", total</pre> <p>So the final value of total = <u>41</u></p>

Learning Review 5

- An algorithm wants to calculate how many coaches are needed for a school trip. Each coach can hold 50 students. Your algorithm must:
 - Allow user to enter total **students** into an array
 - Display the total **coaches** needed

```
students = input
total = 0
for i = 0 to students.length-1
    total = total + students[i]
next i
endfor
coaches = int(total / 50)
printcoaches
```

#enter **students** as an array

#set **total** to 0 (as it is currently unknown)

#Iterate over the **students** array

#Add current value in **students** array to **total**

#Divide the **total** by 50 to work out **coaches**

#Display the **coaches** to the user

Task 5

- 1. An algorithm is used to calculate the total marks in an array. It should display the total marks of the array to the user at the end.
 - Assume the array has any possible 10 values
- 2. An algorithm is used to calculate the lowest mark from a list of given marks. The algorithm stores the marks as an array and then it checks each position to see which one is the lowest. The algorithm will then display this lowest mark to the user.

Exam Question 2*

An English teacher wants to estimate how long it should take his students to read a book. You have been asked to develop an algorithm to calculate this estimate. The algorithm must do the following:

- ask the teacher how many pages the book has and store this in an appropriately named variable
- for every page in the book the algorithm should:
 - ask the teacher if the page looks 'easy' or 'difficult'
 - if a page is 'difficult' then the total number of seconds should increase by 100
 - if a page is 'easy' then the total number of seconds should increase by 40
- after the teacher has entered the difficulty level for all the pages, the algorithm should output the estimated number of seconds that it should take to read the book.

Write pseudocode or draw a flowchart that represents this algorithm.

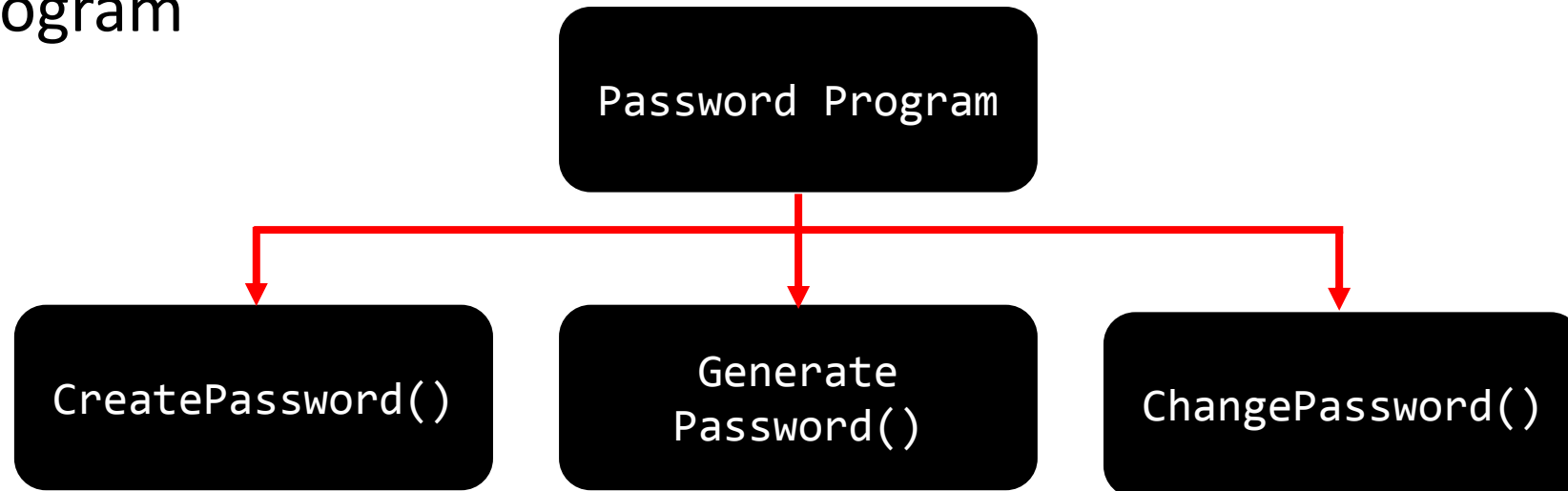
[9 marks]

Pseudocode: Procedures (Structured Approach)

- Computer Science helps us break large problems down into smaller, more manageable problem to solve.
- We can do this with a large program, and use mini sub-programs inside the program, called **procedures (or subroutines/functions)**.
- A procedure is a ***pre-defined block of code which can be used by calling its unique name.***
- Only difference between procedure and a function is that functions have **RETURN** values (a variable that is sent back to the program)

Pseudocode: Procedures (Structured Approach) Advantages

- We use **procedures** (or *structured approach*) because they:
 1. Can be created independently
 2. Are easier to test for bugs/errors
 3. Can be updated/changed without affecting the rest of the program



Pseudocode: Procedures (Structured Approach) - Parameters

- To use procedures, they must first be created.
- Procedures can have *input variables* called **parameters** (these are like the ingredients the procedure needs to work).
- Procedure ***without*** parameters – `procedure depositMoney()`
- Procedure ***with*** parameters – `procedure depositMoney(value)`
- The procedure is called **depositMoney** and it has one parameter - **value**

Pseudocode: **Procedures** (Structured Approach) – Calling Procedures

- Once procedures have been created, they can be called by writing its **name** (and passing in any needed parameters)
- **Example**
- `char_to_code("C")` returns the value **67** (which is the ASCII code for the letter 'C')

Pseudocode: Procedures (Structured Approach) – Writing Procedures

- Writing subroutines is not too different than the previous concepts we have learnt
- The code below only *creates* the subroutine
- To use it we need to *call* its name – `personalDetails("batasi", 24)`

Python	Pseudocode
<pre>def personalDetails(name, age): print("Hello", name) print("You are", age, "years old")</pre> <p>This subroutine has 2 parameters – name and age So we can enter any name and age and the subroutine will print them out</p>	<pre>procedure personalDetails (name, age) : print("Hello", name) print("You are", age, "years old") endprocedure</pre>

Pseudocode: Procedures (Structured Approach) – Writing Procedures (Advanced)

- Let's try a more challenging example

Python

```
def largerNum(num1, num2):  
    if num1 > num2:  
        print("num1 ('',num1,'') is bigger")  
    elif num2 > num1:  
        print("num2 ('',num2,'') is bigger")  
    else:  
        print("Both numbers are equal")
```

This subroutine takes in **2 parameters** (*two numbers*) and will print out a message depending on which number is bigger.

Pseudocode

```
procedure largerNum(num1, num2):  
    if num1 > num2 then  
        display('num1 ('',num1,'') is  
        bigger')  
    else if num2 > num1 then  
        display('num2 ('',num2,'') is  
        bigger')  
    else  
        display('Both numbers are equal')  
    endif  
Endprocedure
```

Pseudocode: Procedures (Structured Approach) – Writing Procedures (Advanced)

- So by writing `largerNum(10, 14)` in the program, the message
`num2 (14) is bigger`
- is displayed to the user.
- What is printed out with the following subroutine calls?
 - `largerNum(10, 9)`
 - `largerNum(-2, -5)`
 - `largerNum(4, 4)`

Learning Review 6

- Develop a procedure which takes in an array of integers, as a parameter. The subroutine will find the highest number in the list and then return this number (array indexing starts at 0)

```
procedure findHighest(numbers)
    highest = numbers[0]
    for count = 1 to (numbers.length)-1
        if numbers[count] > highest then
            highest = numbers[count]
        endif
    next i
    return highest
endprocedure
```

#make the subroutine with the unique name and array
#assume **first** number is the highest (any list)
#iterate over the list starting at **2nd value**
#check if the **current** value of array is higher
#change **highest** to the current value – **new highest**

#return the **highest** number (output of the subroutine)

Exam Question 3*

A developer is using the structured approach to developing a solution and wants to write a subroutine to solve one of the sub-problems. This subroutine should have the following interface:

- The subroutine is called `find_minimum`
- The subroutine takes an array (of integers) as a parameter
- The subroutine returns the smallest value in the parameter array.

Using pseudo-code or a flowchart, write a subroutine that solves this sub-problem. The subroutine must have the correct interface.

[7 marks]

Exam Question 4*

- Develop a function that performs linear search on a list of integers. The function should have the following interface:
 - It should be called **linearSearch**
 - It takes in a **list** of integers and a **target** value, as parameters
 - It has a boolean variable called **found** (False when target is not found, True otherwise)
 - It returns the total number of **searches** taken to check if the **target** value was present within the list of integers.