

Lecture 1

1.1 Data and Types

1.2 Operations, Functions, and Algorithms

In the beginning of our study we focus on two aspects of computing:

data and **operations**

We start exploring each of them individually, although quite soon we begin to view them in tandem.

1.1 Data and Types

The book intentionally differentiates the use of terms *information* and *data*

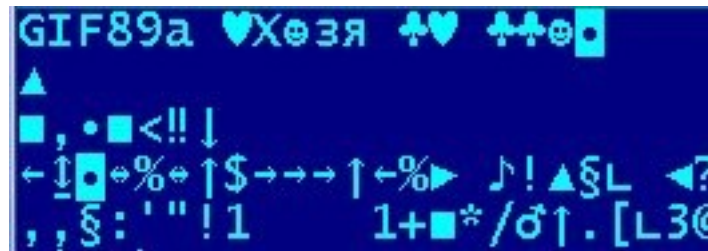
information – is a higher-level abstraction

data – is a low-level representation of information

Example :



What do we see?



← representation
in gif
(beginning)

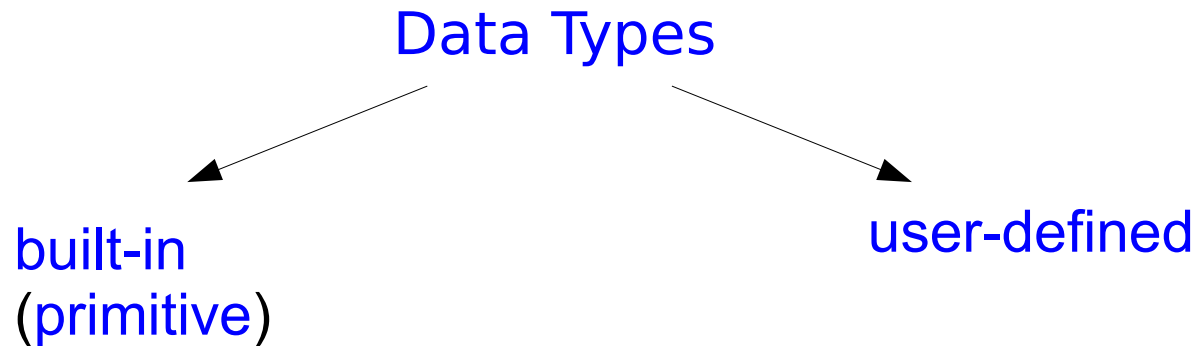


← representation
in jpg
(beginning)

How is it stored (encoding)?

Is the picture stored in JPG format the same *data* as that picture stored in GIF format? Is it the same *information*?

1.1 Data and Types



- Numeric types:
 - plain and long Integers,
 - floating point numbers,
 - complex numbers;
 - booleans
(a subtype of plain integers), ...
- Sequence types:
 - strings,
 - lists,
 - tuples, ...
- Set types, File objects, Classes, ...
see Python documentation

A programmer can define custom data types that can be subsequently used

1.2 Operations, Functions, and Algorithms

We saw that some data types that are commonly used have built-in support, while others are custom-defined by a programmer, if needed.

The same basic principle holds with operations.

Also we have built-in *control structures* (if – else, if-elif, for loop, while loop).

algorithm

(by Merriam Webster): a step-by-step procedure for solving a problem or accomplishing some end especially by a computer.

Let's take a look at two different algorithms for computing the greatest common divisor (gcd) of two integers.

Finding Greatest Common Divisor (gcd)

Algorithm 1

Initialization:

let a be the first integer, b be the second integer (without any loss of generality),
let $guess$ be the smallest of a and b .

For each a and b , we check whether $guess$ divides both original numbers.

If it does - then we got the gcd ($guess$),

If it doesn't - then we decrement $guess$ by 1 and try to divide both a and b by $guess$ again.

If we succeed - we got the gcd ($guess$), if we do not succeed we decrement $guess$ by 1 and try the division again.

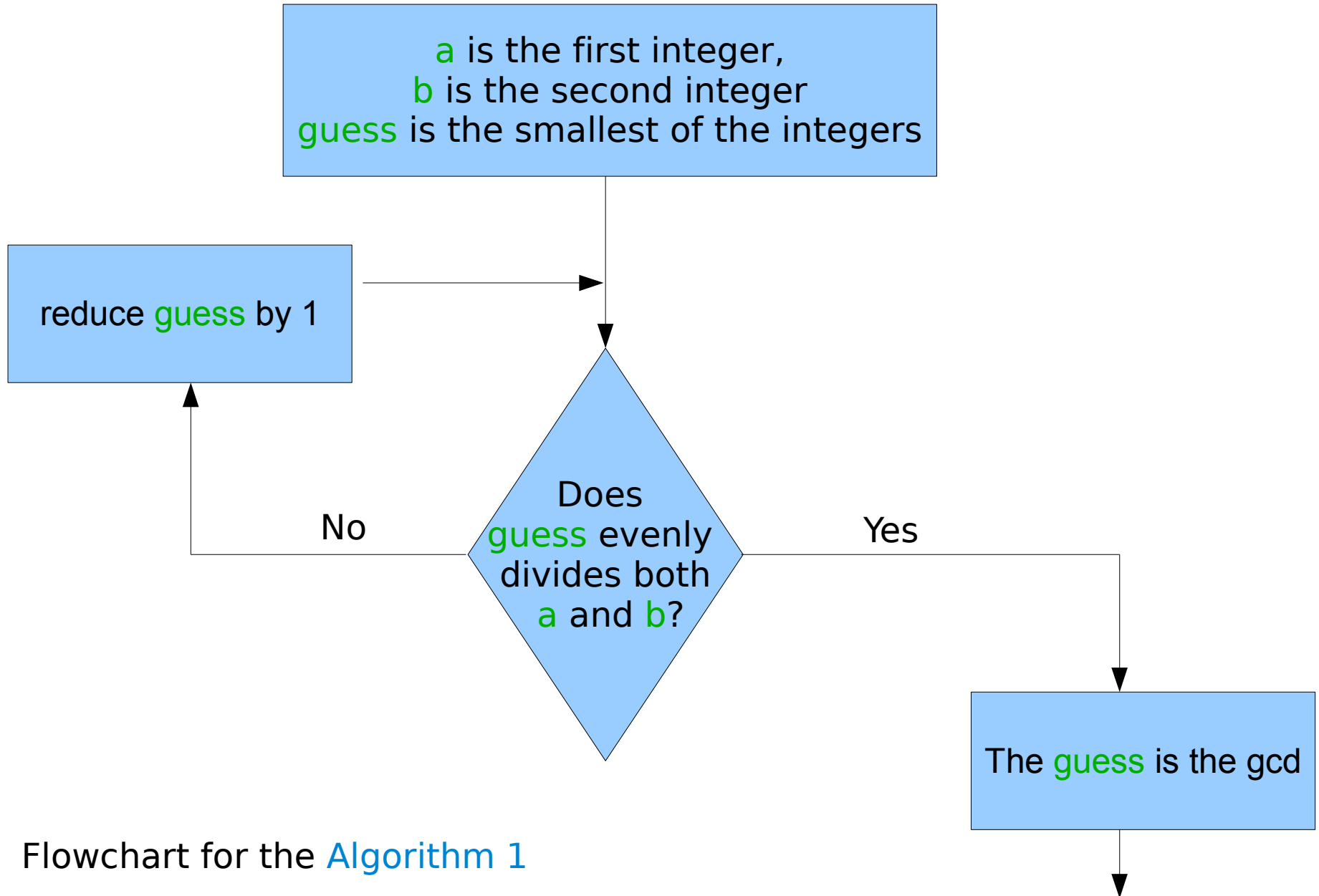
And so forth.

Analysis:

1. The algorithm will eventually stop (since 1 is guaranteed to be a divisor of both original numbers).

2. We are guaranteed to get the gcd, because we are testing from bigger to smaller numbers.

Finding Greatest Common Divisor (gcd)



Flowchart for the [Algorithm 1](#)

Finding Greatest Common Divisor (gcd)

Algorithm 2 Euclid's algorithm

Initialization:

let u be the first integer, v be the second integer (without any loss of generality),

(when we reach the point where v is 0, the current value of u is the gcd)

If v is greater than 0, divide u by v , and store value of v in u and the remainder of the the division in v .

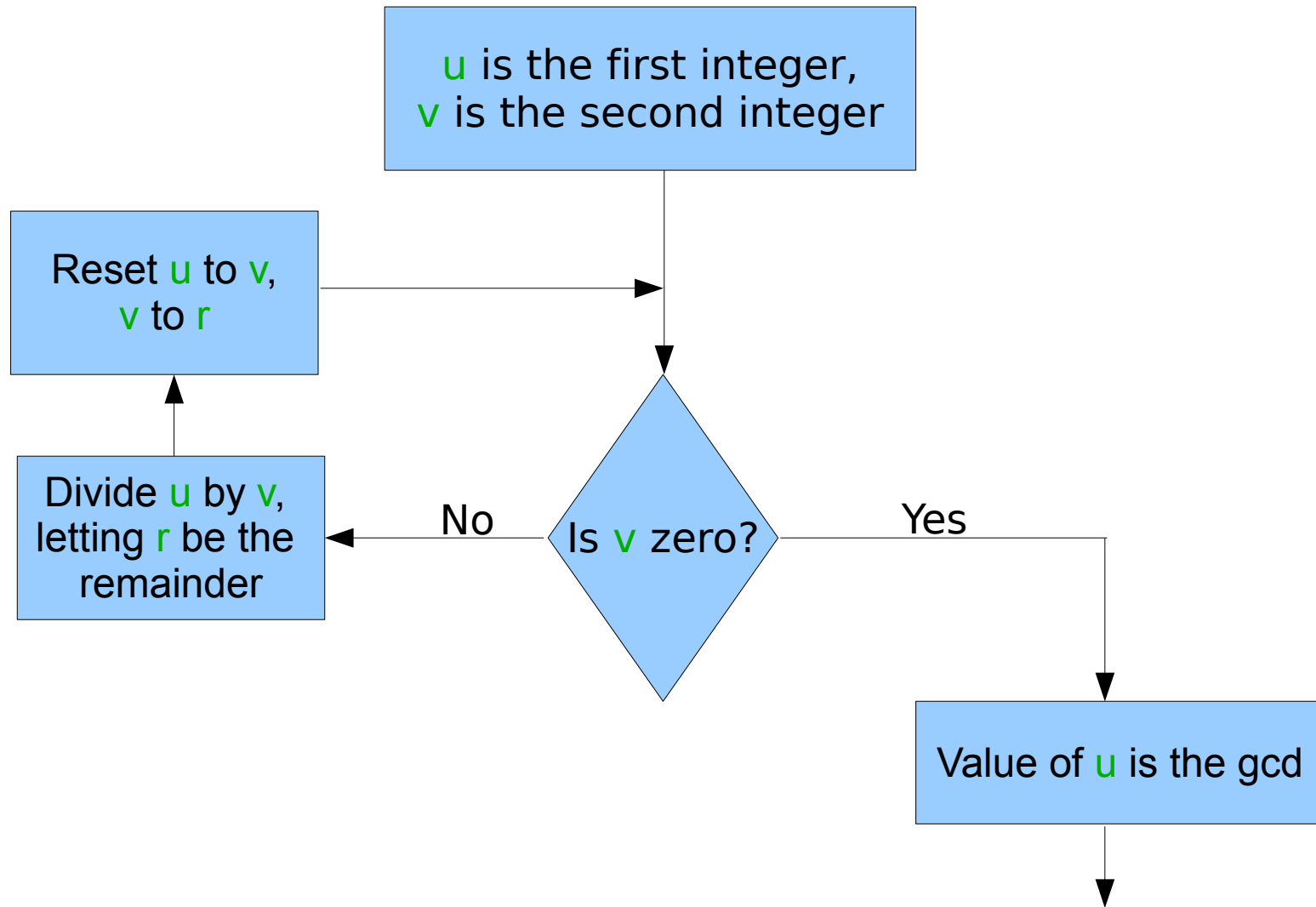
Repeat the process.

Analysis:

1. The algorithm will eventually stop since each time the value of v is updated it becomes smaller (since we store the remainder of the division in v , and remainder is less than the divisor) and will eventually reach 0.

2. We are guaranteed to get the gcd - see an explanation of page 10 (FOR THE GURU).

Finding Greatest Common Divisor (gcd)



Flowchart for the [Algorithm 2](#)

Exercise 1: Implement both algorithms.

Comments: it can be done as one program

Input: from keyboard

Modifications:

1. Input from a file
2. Ability to run the algorithms on as many input pairs as the user wants
3. Use graphics

Exercise 2: p. 29 № 1.5 or 1.8