Lecture 1

1.1 Data and Types

1.2 Operations, Functions, and Algorithms

4.4 Conditional Statements (review)

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)

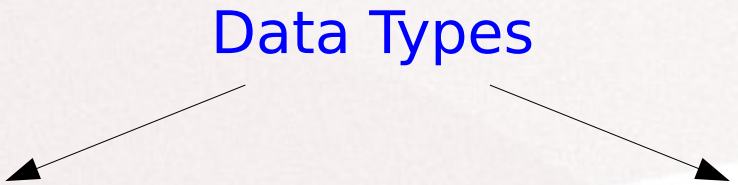
representation in jpg (beginning)

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

1.1 Data and Types



built-in
(primitive)

- 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

user-defined

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 if 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.

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 if 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 WHY?

2. We are guaranteed to get the gcd, WHY?

Finding Greatest Common Divisor (gcd)

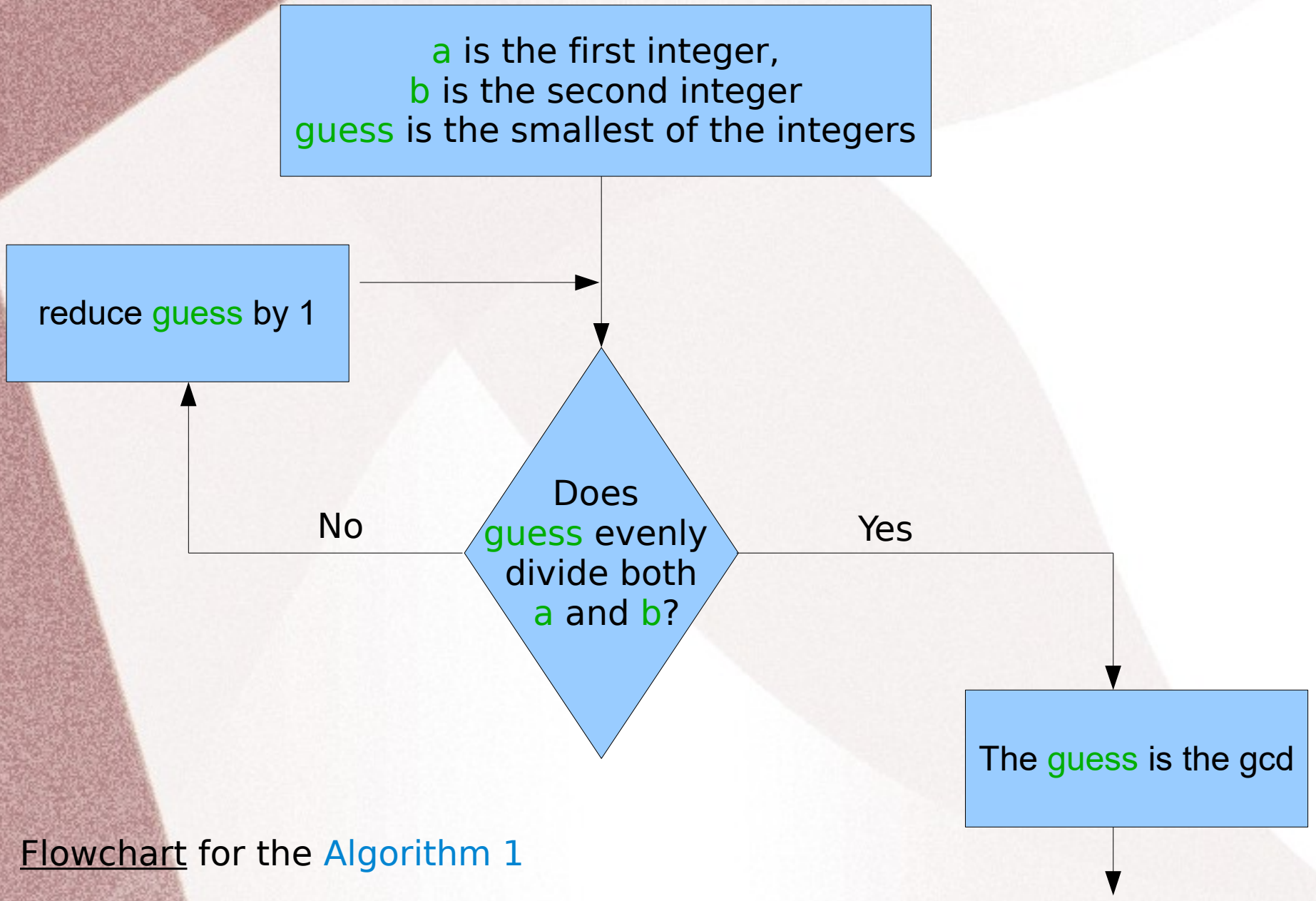
Algorithm 1

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 if 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.

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 on page 10 (FOR THE GURU).

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).

The next two slides are from Lecture18 of CSI 30 Fall 2010 class (you can ignore them if you wish)

The method described in the previous lecture for finding the GCD (using prime factorizations) is quite inefficient.

Euclidean Algorithm gives a more efficient way of finding GCD.
It is named after Greek mathematician Euclid.

procedure *gcd*(a, *b*: positive integers)
x := *a*
y := *b*
while $y \neq 0$:
    r := *x* ***mod*** *y*
    x := *y*
    y := *r*
{gcd(a,b) is *x*}

This algorithm uses the following proved lemma:

[Lemma 1]
Let *a*=*bq*+*r*, where *a,b,q*, and *r* are integers. Then *gcd(a,b) = gcd(b,r)*

Example 4: find gcd(1001,1331)

Solution: gcd(1001,1331) = gcd(1331,1001)
– we prefer to take smaller as divisor and larger as dividend.

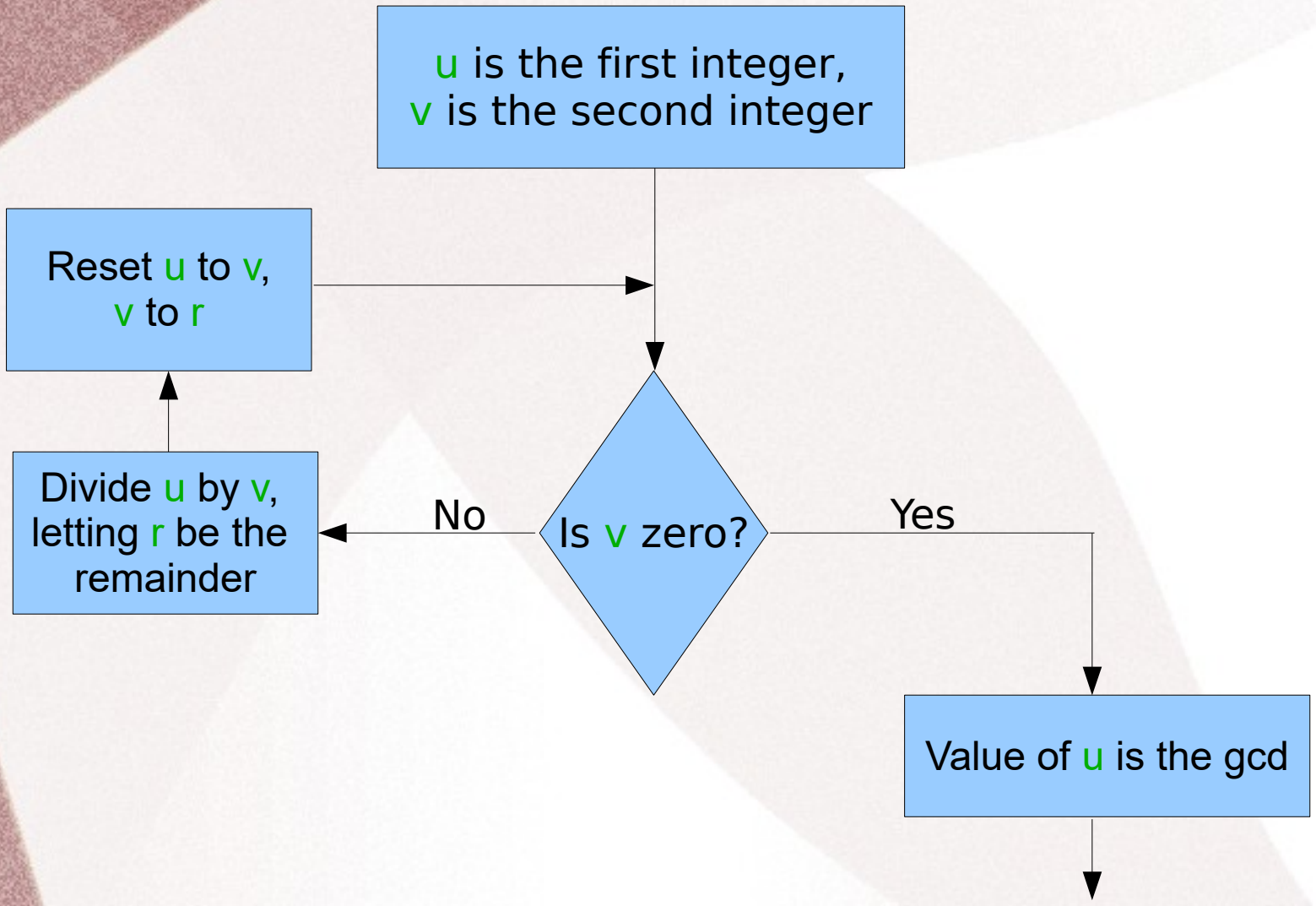
$$\frac{1331}{1001} = 1\ R\ 330$$

Thus gcd(1331,1001) = gcd(1001,330) = gcd(330,11) = gcd(11,0) STOP

Answer: gcd(1001,1331) = 11

$$\frac{1001}{330} = 3\ R\ 11 \qquad \frac{330}{11} = 30\ R\ 0$$

Finding Greatest Common Divisor (gcd)



Flowchart for the Algorithm 2

4.4 Conditional Statements

Syntax of the *if-statement*:

```
if <condition>:
    body
```

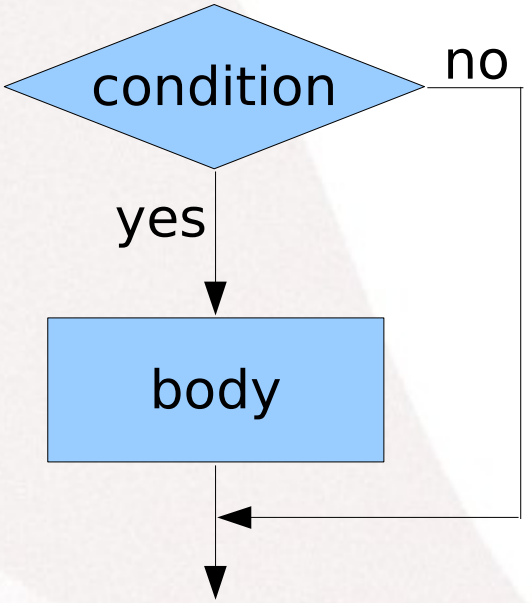
example:

condition

```
if t>90:
    print("Heat Warning!")
```

body

activity diagram (flowchart) of a simple if-statement:

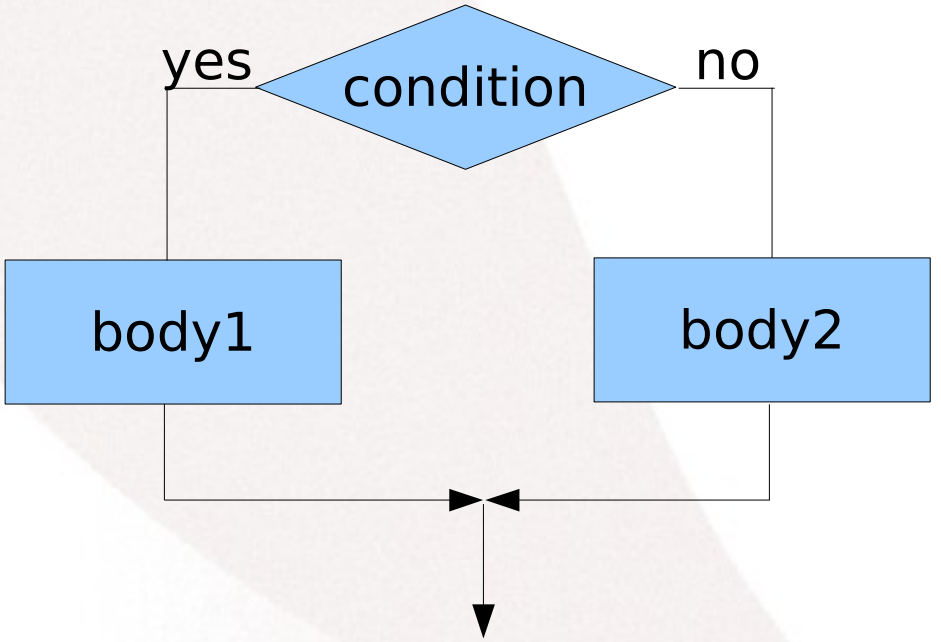


4.4 Conditional Statements

Syntax of the *if-else statement*:

```
if <condition>:
    body1
else:
    body2
```

activity diagram (flowchart) of an if-else statement:



example:

condition

body1

```
if t>90:
   print("Heat Warning!")
else :
   Print('everything is fine')
```

body2

4.4 Conditional Statements

Syntax of the *if-elif statement*:

```
if <condition>:
   body1
elif <condition2>:
   body2
…
elif <conditionN>:
   bodyN
else:
   bodyN1
```

Same as to the right, but more compact and elegant →

```
if <condition>:
   body1
else:
   if <condition2>:
      Body2
   else:
      if <condition3>:
         Body3
      else: ...
         ...
            else:
               if <conditionN>:
                  bodyN
               else:
                  bodyN1
```

Example 1 : Assigning a grade for a test

Let's write a program that is based on the test score assigns the test grade.

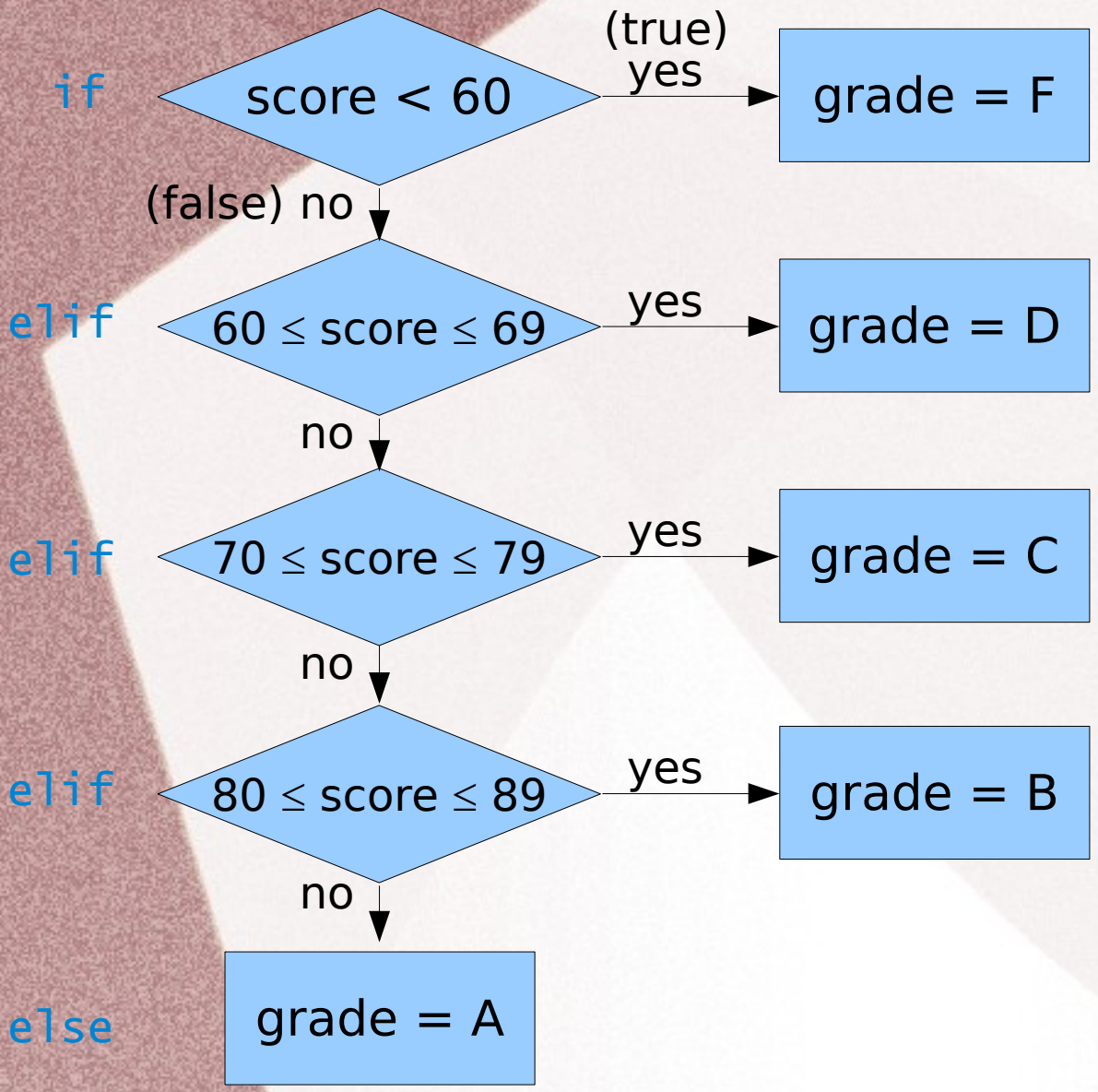
Analysis:

Input: a whole number less than 101 (which is the test score)

Output: grade that corresponds to the test score

90-100: A
80-89: B
70-79: C
60-69: D
<60 : F

Example 1 : Assigning a grade for a test



Activity diagram (flowchart)

Example 1 : Assigning a grade for a test

a program:

```
def main():
    score=input("Please input your Test score:")
    if (score < 60):
        print("Test Grade is F")
    elif (score >= 60 and score <= 69):
        print("Test Grade is D")
    elif (score >= 70 and score <= 79):
        print("Test Grade is C")
    elif (score >= 80 and score <= 89):
        print ("Test Grade is B")
    else :
        print ("Test Grade is A")

main()
```

see program scoreToGrade.py

In-class work

- Write a simple program that asks the user to input three decimal numbers and returns their average if the numbers are all positive, otherwise returns warning message.

- Simulate Euclid's algorithm for computing the **gcd** of 180 and 75 on a piece of paper, listing all pairs of numbers that were considered along the way.

- Consider the following piece of program:

```
x,y are values provided by the user (from keyboard)
if x > 5:
   if y<=3 and x>8:
      print("answer is A")
   else:
      print("answer is B")
elif y<6 or x<2:
   print("answer is C")
else:
   print("answer is D")
```

Predict the output if the user enters:

1) 4 and then 4

2) 9 and then 4

3) 1 and then 9

4) 6 and then 2

Homework assignment

Exercise 1: Implement both algorithms for finding **gcd**.

Comments: *it can be done as one program*

Input: from keyboard

Python operators you can use:
a%b returns the remainder of the division
(in math: *a* **mod** *b*)

Modifications (not mandatory):

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

More Exercises: p. 29 № 1.4, 1.5 or 1.8, and page 154 № 4.22