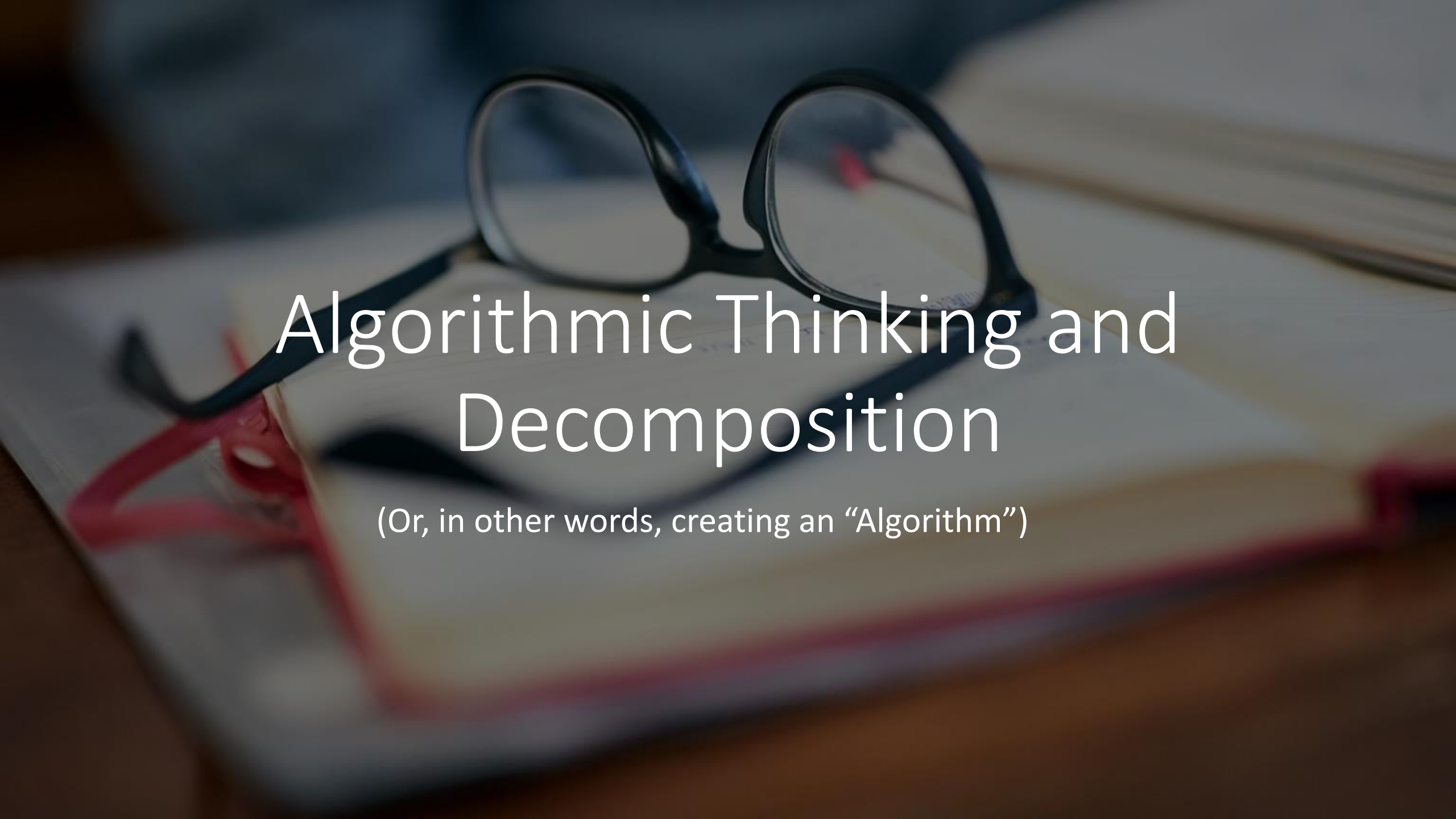
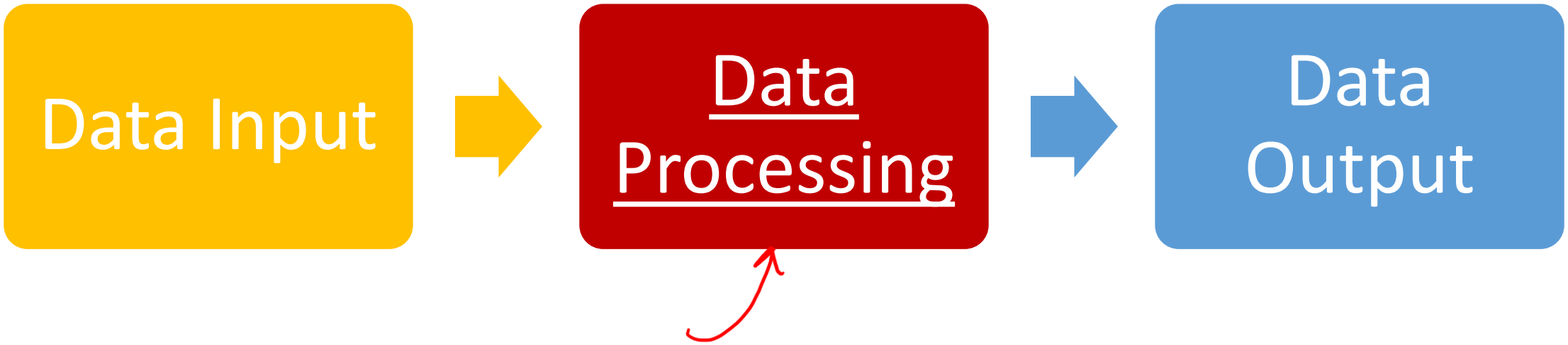


Algorithmic Thinking and Decomposition

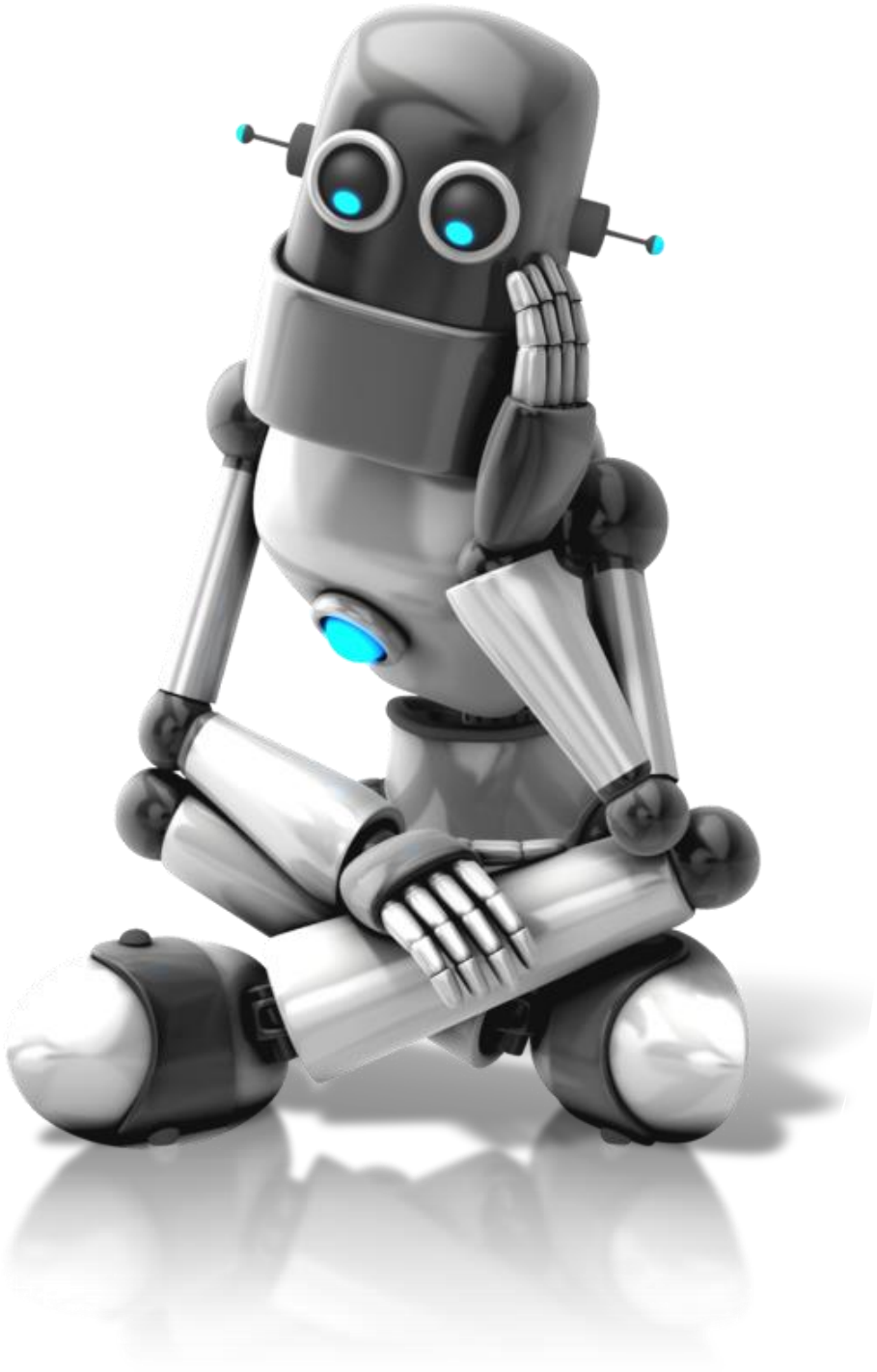
(Or, in other words, creating an “Algorithm”)

Three main stages



How do we do the “Data Processing” bit?

- The “data processing” part of the previous slide is where our skills as software engineers comes in.
- Learning syntax and semantics of Python (or other languages) is important, but...
- The *skill* we want to learn here is: **how do we *think* computationally** (emphatize with a computer, so to speak), **so that we can then write *algorithms* that solve a given problem**.
- Intuitions, guesstimates, gut-feelings, are not going to work *on their own* (though will very likely *inform* the process)
- So, what does work then?
  - Less of Picard, more of Data



Thinking like a Computer?

- The key skills we need to start thinking more “computationally”, so that we can write programmatic solutions that computers can solve:
  - **Algorithmic Thinking**
  - Analysis (coming up soon)
  - **Decomposition**

- We also need more general “thinking skills”, which we use very often even when not working with computers and/or machines
  - Abstraction
  - Generalization

CASE STUDY

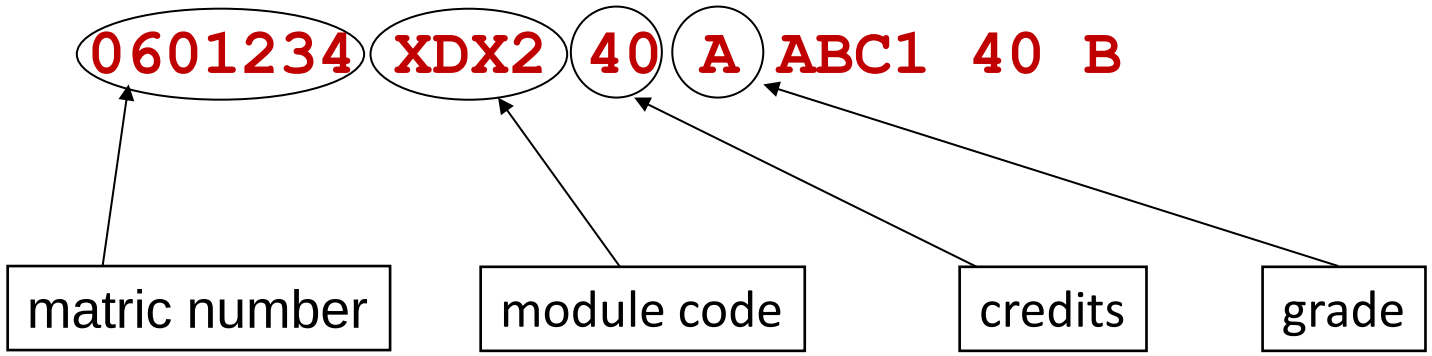
(Algorithmic Thinking & Decomposition)

Case Study: Planning / Problem Solving

- Calculating students' grade point averages

- Data on students, their modules and grades is stored in a file (one student can have multiple modules)

```
0606559 VBN2 30 B FGH9 50 A SDF2 50 N DFG5 60 C
0500300 HHJ3 40 D FGH3 10 E JHG3 10 F
0605456 LKJ1 10 G CDS2 30 C
0601234 XDX2 40 A ABC1 40 B
```



Grade Point Averages

- For each student, the following calculation must be done.
- Each grade is converted into grade points using the following table:

A	B	C	D	E	F	G	All others
16	14	12	10	8	6	2	0

- **Grade points for each module**:
  - the number of credits is multiplied by the grade points corresponding to the grade
- **Total grade points**:
  - Grade points for all modules are added up to give the total grade points for the student.
- **Grade point average (GPA)**:
  - Total grade points divided by number of credits

Grade Point Averages

- For example, for this student:

```
0605456 LKJ1 10 G CDS2 30 C
```

Matric number	0605456
Credits for LKJI	10
Grade points for LKJI (G grade)	10 x 2 = 20
Credits for CDS2	30
Grade points for CSD2 (C grade)	30 x 12 = 360
Total grade points	360+20 = 380
Grade point average	380 / (30+10) = 9.5

Note: You need both (a) total grade points and (b) total credits to then compute the grade point average

The Problem

- Write a program to:
  1. read in students' data from a file,
  2. calculate the GPA for each student, &
  3. print the GPA of each student, separated into two categories:
     - students with a GPA of at least 10.0, and
     - students with a GPA of less than 10.0

```
Students with GPA >= 10.0

0606559 190 10.2
0601234  80 15.0

Students with GPA < 10.0

0500300  60  9.0
0605456  40  9.5
```

Expected output

Planning

Three main stages

Data Input → Data Processing → Data Output

```
0606559 VBN2 30 B FGH9 50 A SDF2 50 N DFG5 60 C
0500300 HHJ3 40 D FGH3 10 E JHG3 10 F
0605456 LKJ1 10 G CDS2 30 C
0601234 XDX2 40 A ABC1 40 B
```

matric number | module code | credits | grade

```
Students with GPA >= 10.0

0606559 190 10.2
0601234  80 15.0

Students with GPA < 10.0

0500300  60  9.0
0605456  40  9.5
```

Planning

- How do we begin to write the program? We need a plan.

- To develop a plan, we need to think about:
  - **data structures** for the problem, and
  - the necessary **algorithms**
    - i.e., the **sequence of steps** to solve the problem

Algorithms + *Data Structures* = *Programs*

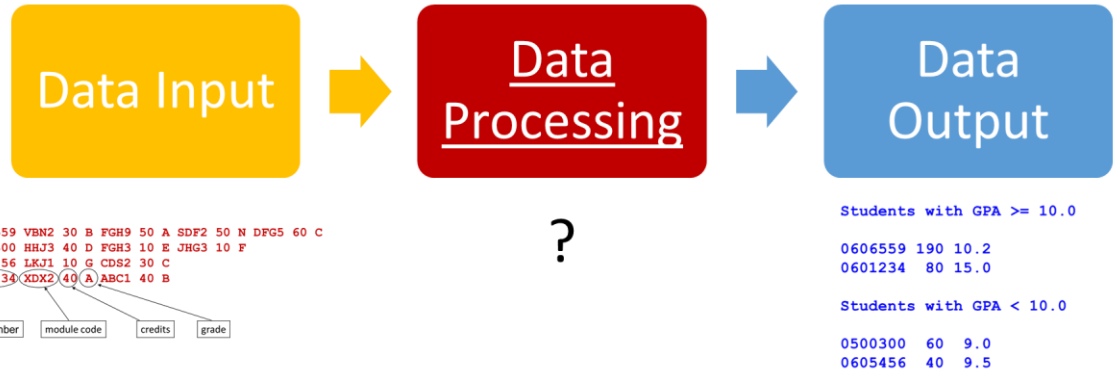
- In other words:
  - what data do we need to store, and what is the appropriate structure to store it; and
  - how do we process it?

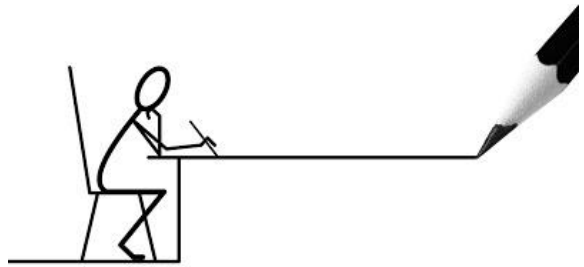
Do we need to store anything at all?

- Do we need to **store** information about students and their GPAs that we read from the file
  - for example in a list or dictionary

OR

- Can we just do the **processing** and produce the output ***while we are reading*** the original data in, line by line?





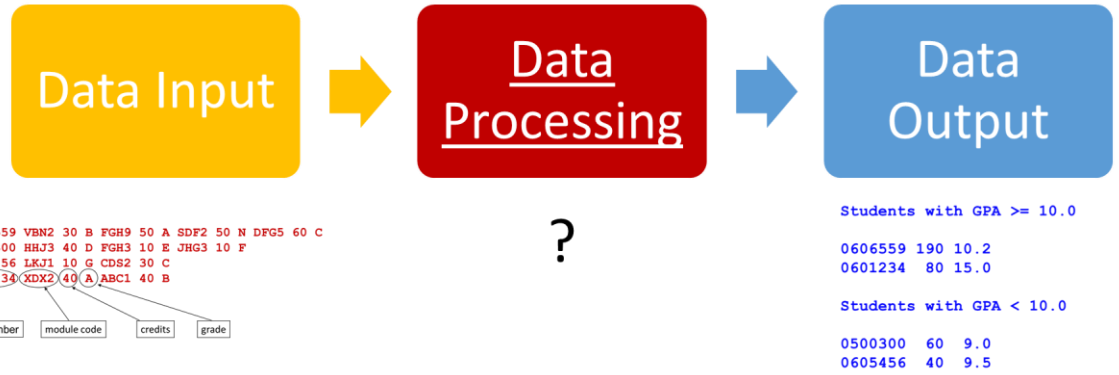
Storing information about students?

- We could read one line at a time, compute GPA for that student ***and display it***, and move on the next one, right?

Storing information about students?

- We could read one line at a time, compute GPA for that student ***and display it***, and move on the next one, right?

- **NOPES!** The output has to be separated into two groups (high GPAs and then low GPAs). So:
  - We need to **store** all total credits AND computed GPAs first, and
  - Then display them in two groups



Data Structure to Store Processed Information?

- We will compute information about each student, identified by their matric numbers, with no particular sequence:

- So… list? array? tuple? dictionary?

Data Structure to Store Processed Information?

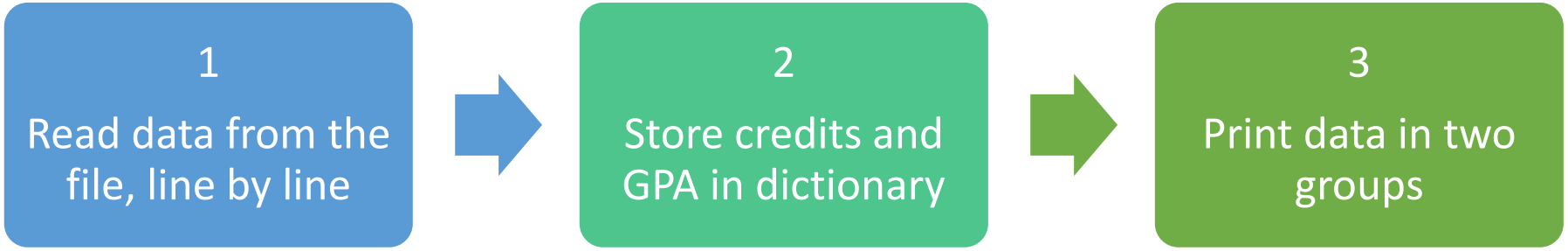
- We have information about each student, identified by their matric numbers, with no particular sequence:

- So… we'll use a **dictionary** in which the keys are matric numbers, represented by strings.
  - The **values** are not just a number: we need to store *total credits* **and** *GPAs*
    - So we use a **nested dictionary**
    - Though we could have used a dictionary whose values were a tuple*.

```
{ "0605456": { "credits":40, "gpa":9.5 },
  ...
}
```

**We could use a list as well, but seeing that the data will be immutable, tuple is better (more efficient)*

Top Level Plan

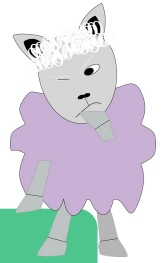


- Steps 1 looks straightforward enough (basically line by line input from a file)
- Step 2 (mostly) and 3 require a bit more thought
  - Break it down into smaller steps?

Decomposition

1

Read data from the file, line by line

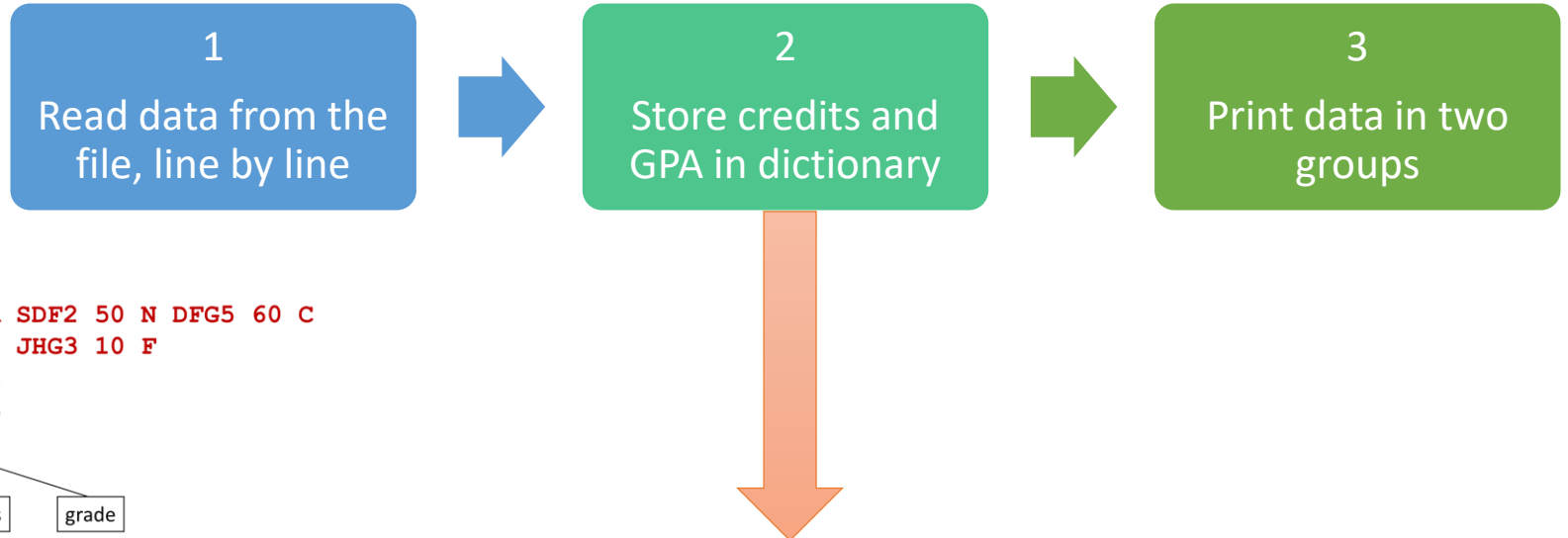


2

Store credits and GPA in dictionary

3

Print data in two groups



Are we there yet?
Can you go from this stage of planning to writing code?
Let's refine this further…

1
Read data from the file, line by line

2
Store credits and GPA in dictionary

3
Print data in two groups

```
0606559 VBN2 30 B FGH9 50 A SDF2 50 N DFG5 60 C
0500300 HHJ3 40 D FGH3 10 E JHG3 10 F
0605456 LKJ1 10 G CDS2 30 C
0601234 XDX2 40 A ABC1 40 B
```

matric number | module code | credits | grade

Calculate the total credits and GPA

Store in dictionary against matric number

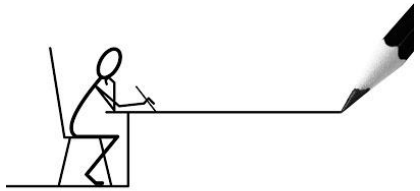
Split the current line into a **list of strings**

For each module, add the credits to a running total

For each module, add the grade points to a running total

Calculate the GPA

Are we there yet?



1
Read data from the file, line by line

2
Store credits and GPA in dictionary

3
Print data in two groups

```
0606559 VBN2 30 B FGH9 50 A SDF2 50 N DFG5 60 C
0500300 HHJ3 40 D FGH3 10 E JHG3 10 F
0605456 LKJ1 10 G CDS2 30 C
0601234 XDX2 40 A ABC1 40 B
```

matric number

module code

How will we get the **grade points (integers)** against the **letter grades?**

Store in dictionary against matric number

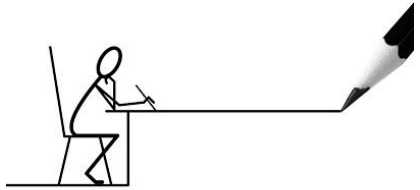
Split the current line into a **list of strings**

For each module, add the credits to a running total

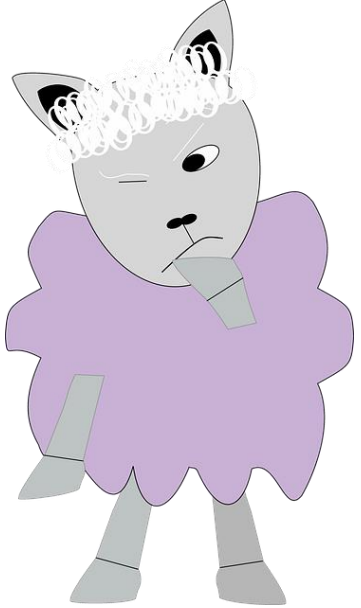
For each module, add the grade points to a running total

Calculate the GPA

Are we there yet?



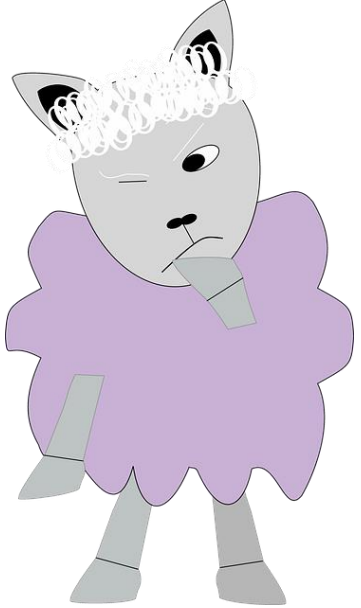
Hmm…



- How do we store (and retrieve) this information:

A	B	C	D	E	F	G	All others
16	14	12	10	8	6	2	0

Hmm…



- How do we store (and retrieve) this information:

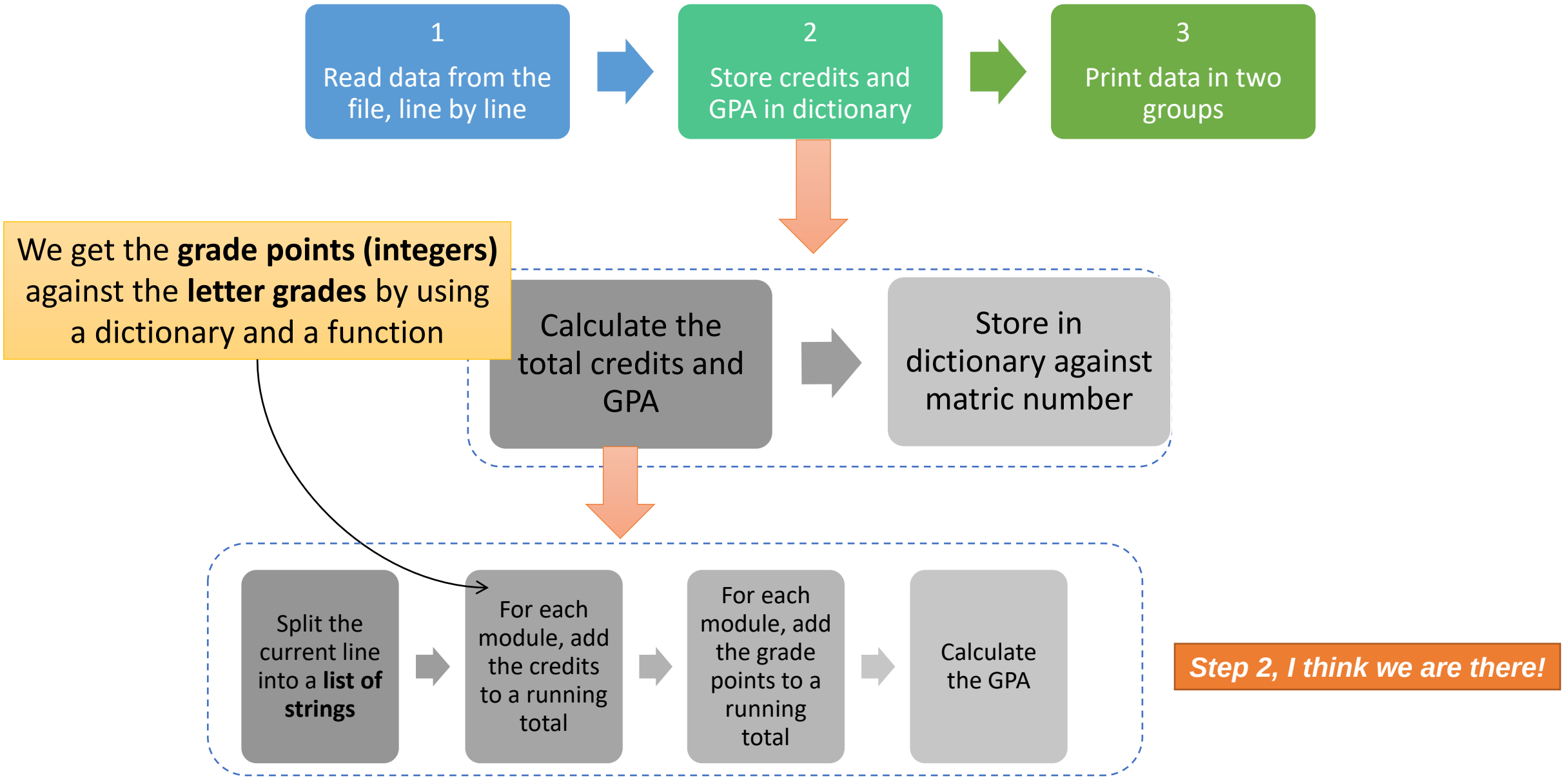
A	B	C	D	E	F	G	All others
16	14	12	10	8	6	2	0

- This looks suspiciously like a lookup table, doesn't it?
  - Dictionary?
- Also, there should be some sort of a… **function** to give you the grade point against each letter grade
- **Dictionaries** and **Functions** to the rescue!

Converting grades to grade points

A	B	C	D	E	F	G	All others
16	14	12	10	8	6	2	0

```
def gp(g):
    table = { 'A':16, 'B':14, 'C':12, 'D':10,
              'E':8,  'F':6,  'G':2 }
    return table.get(g,0)
# 0 is returned if key is not present
```


1
Read data from the file, line by line
2
Store credits and GPA in dictionary
3
Print data in two groups
We get the **grade points (integers)** against the **letter grades** by using a dictionary and a function
Calculate the total credits and GPA
Store in dictionary against matric number
Split the current line into a **list of strings**
For each module, add the credits to a running total
For each module, add the grade points to a running total
Calculate the GPA
Step 2, I think we are there!

1

Read data from the file, line by line

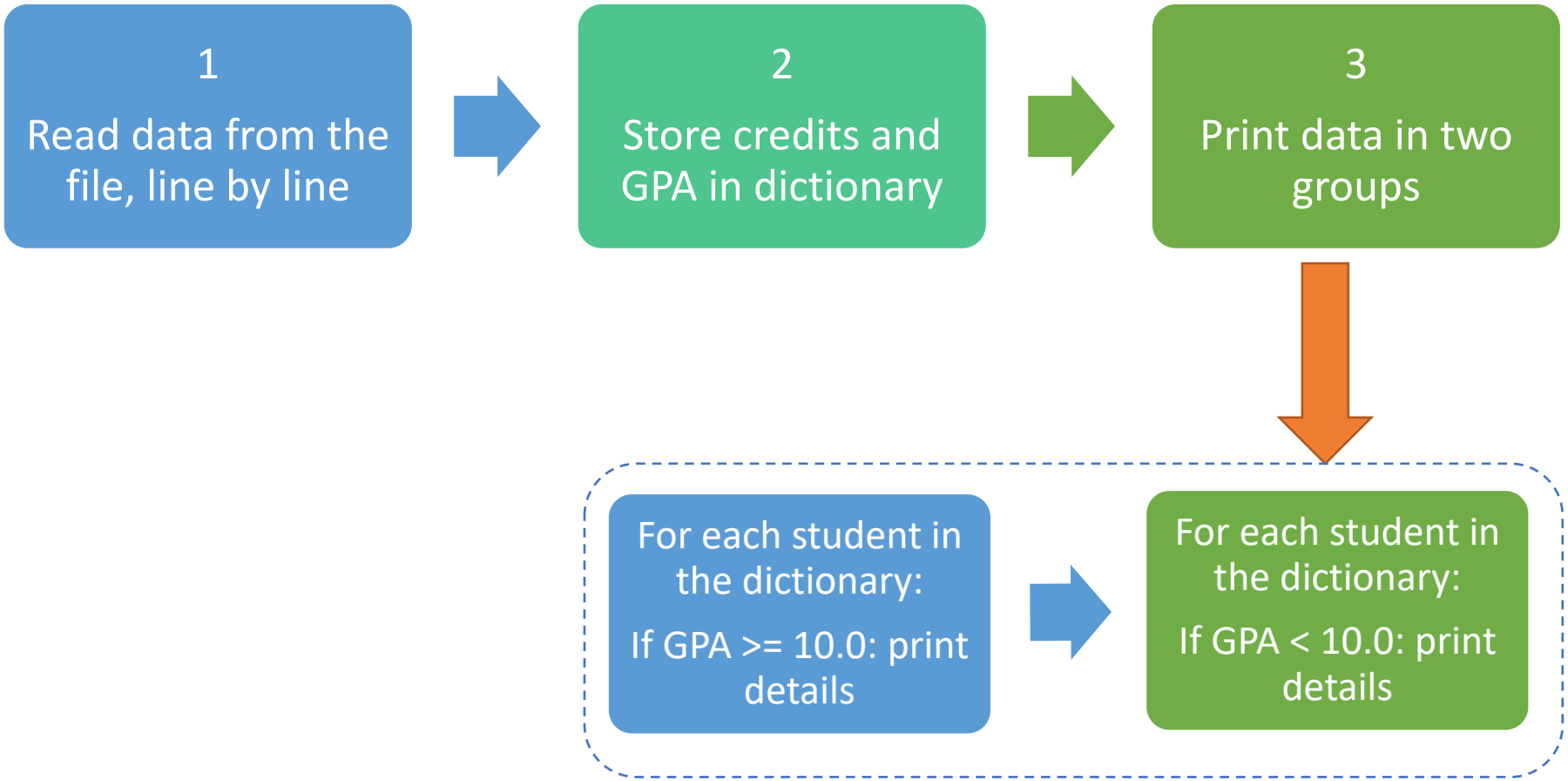
2

Store credits and GPA in dictionary

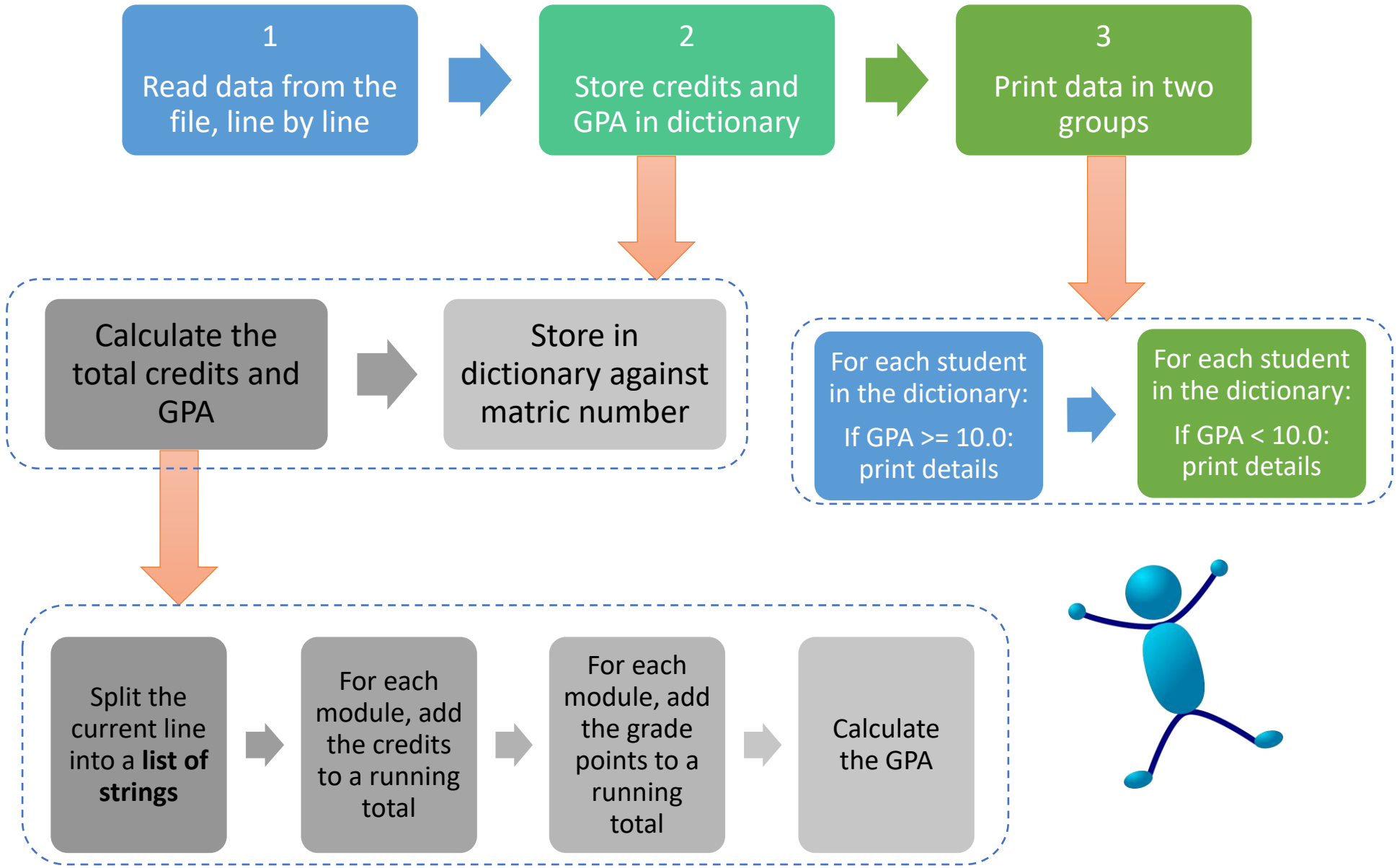
3

Print data in two groups

```
Students with GPA >= 10.0

0606559 190 10.2
0601234  80 15.0

Students with GPA < 10.0

0500300  60  9.0
0605456  40  9.5
```


1
Read data from the file, line by line
2
Store credits and GPA in dictionary
3
Print data in two groups
For each student in the dictionary:
If GPA >= 10.0: print details
For each student in the dictionary:
If GPA < 10.0: print details

The Complete Plan

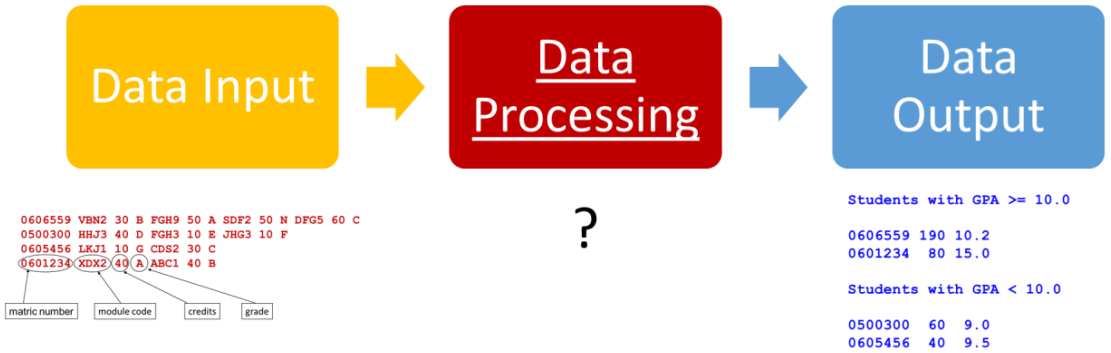


Let’s get coding

yeh yeh yeh!

Developing the Program

```
# Step 1 and 2
with open("grades.txt","r") as f:
        record = process(f)

# Step 3
output(record)
```

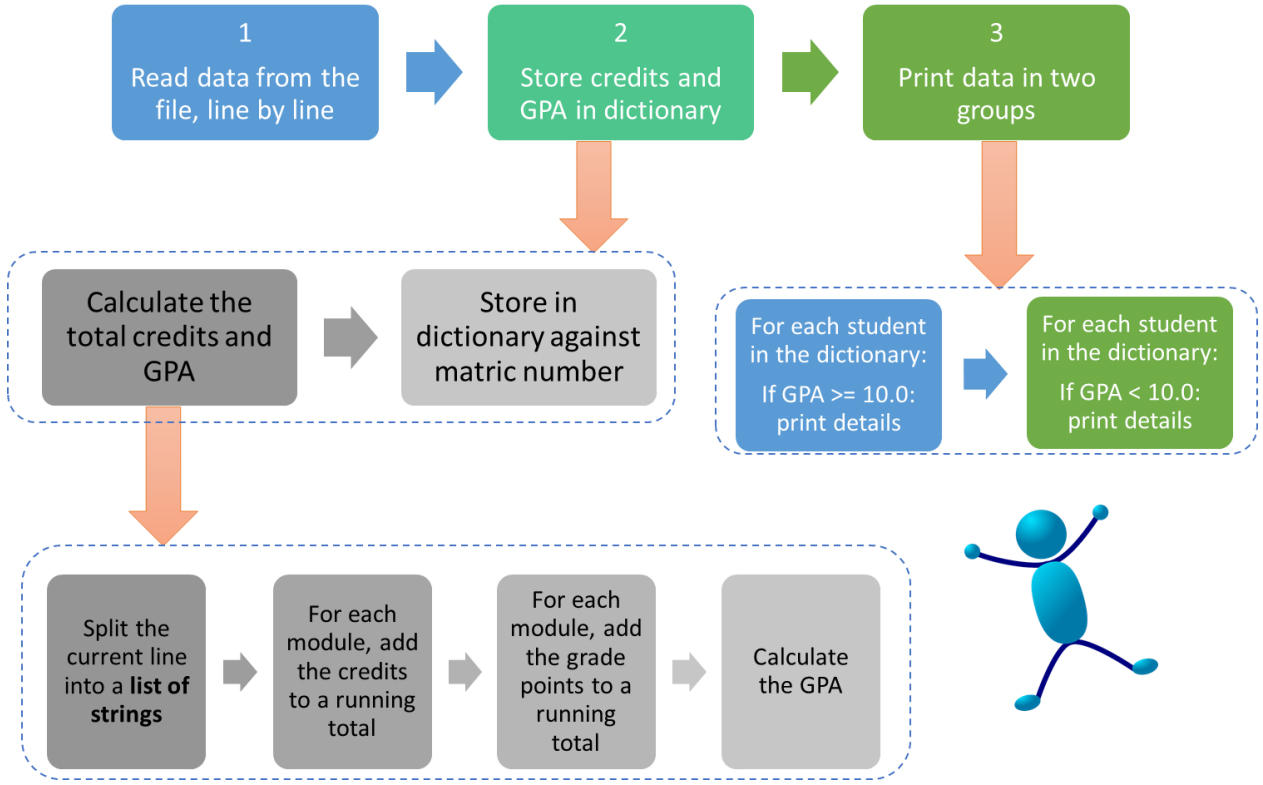


- We are taking a **top-down** approach
  - Write the "main" of the program first, using "stub" (empty) functions
    - input, processing, and output are identified
  - Then, define the functions, which may themselves call on stub functions
  - And so on until you get to "leaf" functions (functions that don't call any other functions)

- So all we need to do now is define the functions to
  - **`process`** and
  - **`output`**

The function `process`

```
def process(f):
    record = {}
    # For each line (i.e. student) in the file f,
    # do something -->
    return record
```



The function `process`

```
def process(f):
    record = {}
    line = f.readline()
    while line != '':
        # Process the information in line -->
        line = f.readline()
    return record
```

The function `process`

```
def process(f):
    record = {}
    line = f.readline()
    while line != '':
        line = line[:-1] # Remove the final newline character
        # Process the information in line -->
        line = f.readline()
    return record
```

The function process

```
def process(f):
    record = {}
    line = f.readline()
    while line != '':
        line = line[:-1]
        #Split the current line into a list of strings
        data = line.split(' ')

        # The matric is the first split element
        matric = data[0]

        # Initialise running totals
        credits = 0
        points = 0

        # For each module, add credits and grade points to running total -->
        line = f.readline()
    return record
```

The function process

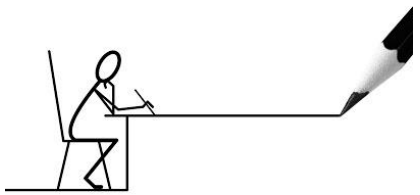
```
def process(f):
    record = {}
    line = f.readline()
    while line != '':
        line = line[:-1]
        #Split the current line into a list of strings
        data = line.split(' ')

        # The matric is the first split element
        matric = data[0]

        # Initialise running totals
        credits = 0
        points = 0

        # For each module, add credits and grade points to running total -->
        line = f.readline()
    return record
```

0606559 VBN2 30 B FGH9 50 A SDF2 50 N DFG5 60 C

data = ["0606559", "VBN2", "30", "B", "FGH9", "50", …]

The function process

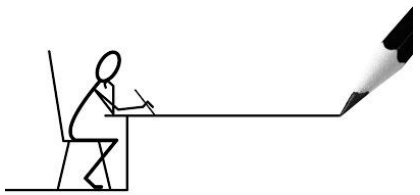
```
def process(f):
    record = {}
    line = f.readline()
    while line != '':
        line = line[:-1]
        data = line.split(' ')
        matric = data[0]
        credits = 0
        points = 0
        modules = ?                                  # for each
        for i in range(modules):                     # module ...
            credit = int(data[?])                    #
            credits = credits + credit               # add credit
            grade = data[?]                          #
            points = points + gp(grade)*credit       # add points
        # calculate GPA -->
        line = f.readline()
    return record
```



0606559 | VBN2 30 B | FGH9 50 A | SDF2 50 N | DFG5 60 C

The function process

```
def process(f):
    record = {}
    line = f.readline()
    while line != '':
        line = line[:-1]
        data = line.split(' ')
        matric = data[0]
        credits = 0
        points = 0
        modules = int((len(data)-1)/3)          # for each
        for i in range(modules):                # module ...
            credit = int(data[?])               #
            credits = credits + credit          # add credit
            grade = data[?]                     #
            points = points + gp(grade)*credit  # add points
        # calculate GPA -->
        line = f.readline()
    return record
```



0606559 | VBN2 30 B | FGH9 50 A | SDF2 50 N | DFG5 60 C

The function process

```
def process(f):
    record = {}
    line = f.readline()
    while line != '':
        line = line[:-1]
        data = line.split(' ')
        matric = data[0]
        credits = 0
        points = 0
        modules = int((len(data)-1)/3)          # for each
        for i in range(modules):                # module ...
            credit = int(data[i*3+2])           #
            credits = credits + credit          # add credit
            grade = data[i*3+3]                 #
            points = points + gp(grade)*credit  # add points
        # calculate GPA -->
        line = f.readline()
    return record
```

0606559 | VBN2 30 B | FGH9 50 A | SDF2 50 N | DFG5 60 C

The function `process`

```
def process(f):
    record = {}
    line = f.readline()
    while line != '':
        line = line[:-1]
        data = line.split(' ')
        matric = data[0]
        credits = 0
        points = 0
        modules = int((len(data)-1)/3)
        for i in range(modules):
            credit = int(data[i*3+2])
            credits = credits + credit
            grade = data[i*3+3]
            points = points + gp(grade)*credit
        gpa = points/credits
        record[matric] = { "credits":credits, "gpa":gpa }
        line = f.readline()
    return record
```

The function `output`

```
def output(r):
    #Print students with GPA >=10
    print("Students with GPA >= 10.0")
    print()
    for m in r:  # m is each matric number in turn
        if r[m]["gpa"] >= 10.0:
            print(m, r[m]["credits"], r[m]["gpa"])
    print()

    #Print students with GPA < 10
    print("Students with GPA < 10.0")
    print()
    for m in r:  # m is each matric number in turn
        if r[m]["gpa"] < 10.0:
            print(m, r[m]["credits"], r[m]["gpa"])
```