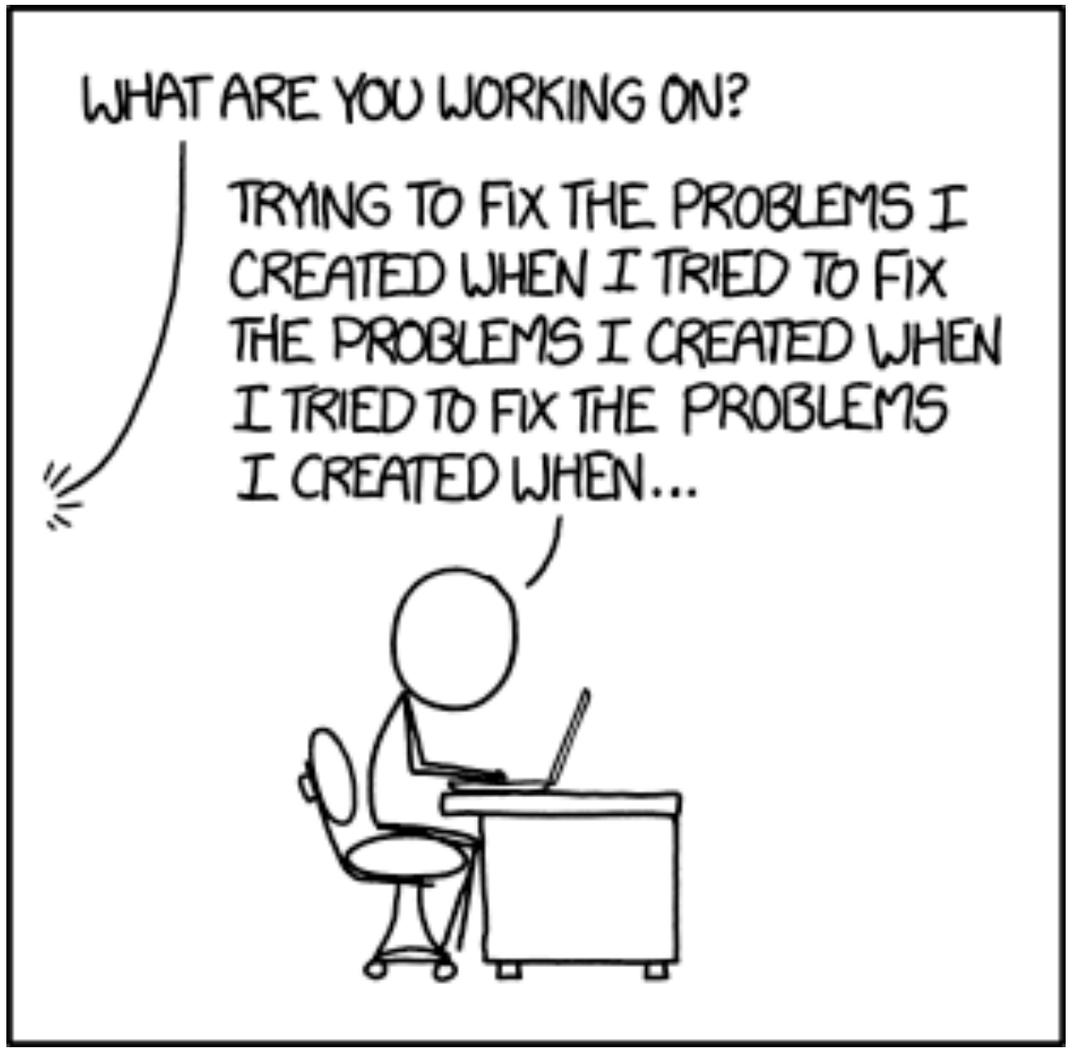


CSCI 141

Lecture 2
Inclusive Learning Environment
Computers and Hardware
Algorithms and Pseudocode
Function Calls

Announcements

Announcements

- Your first lab is on Monday or Tuesday.

Announcements

- Your first lab is on Monday or Tuesday.
- The CS department has its own computer network and labs, with separate user accounts.

Announcements

- Your first lab is on Monday or Tuesday.
- The CS department has its own computer network and labs, with separate user accounts.
- You need to activate your CS Account **before** lab:

Announcements

- Your first lab is on Monday or Tuesday.
- The CS department has its own computer network and labs, with separate user accounts.
- You need to activate your CS Account **before** lab:
  - The username will be the same as your WWU username, but you will set a different password.

Announcements

- Your first lab is on Monday or Tuesday.
- The CS department has its own computer network and labs, with separate user accounts.
- You need to activate your CS Account **before** lab:
  - The username will be the same as your WWU username, but you will set a different password.
  - You **must** activate your CS account from a **non-CS** computer **before** you arrive at your first lab next week.

Announcements

- Your first lab is on Monday or Tuesday.
- The CS department has its own computer network and labs, with separate user accounts.
- You need to activate your CS Account **before** lab:
  - The username will be the same as your WWU username, but you will set a different password.
  - You **must** activate your CS account from a **non-CS** computer **before** you arrive at your first lab next week.
  - Go to http://password.cs.wwu.edu and follow the instructions there.

Announcements

- The first lab will get you acquainted with Thonny and the CS labs.
  - If you want to install Thonny on your own computer, you can download it from http://www.thonny.org
  - If you want to use a different IDE or editor, that's fine too.
- Today's QOTD is about the syllabus. You can look at the syllabus when answering the questions!

Last time: Takeaways

- This course covers the basics of programming, and is the beginning of a journey towards a new way of thinking and solving problems.

- Programming and problem-solving are useful skills, whether you plan to go into computer science or not.

- Comments (beginning with #) are ignored by Python. We use them to help out other humans reading our code.

- A program can display text on the screen using a line such as:

```
print("Hello, world!")
```

  or equivalently:

```
print('Hello, world!')
```

Goals

- A slide (or two) like this will appear at the beginning of each lecture.
- This tells you what I want you to get out of the lecture
  - I will use it when writing exams
  - You can use it when studying for exams
- The goal is transparency: you know what I want you to know.

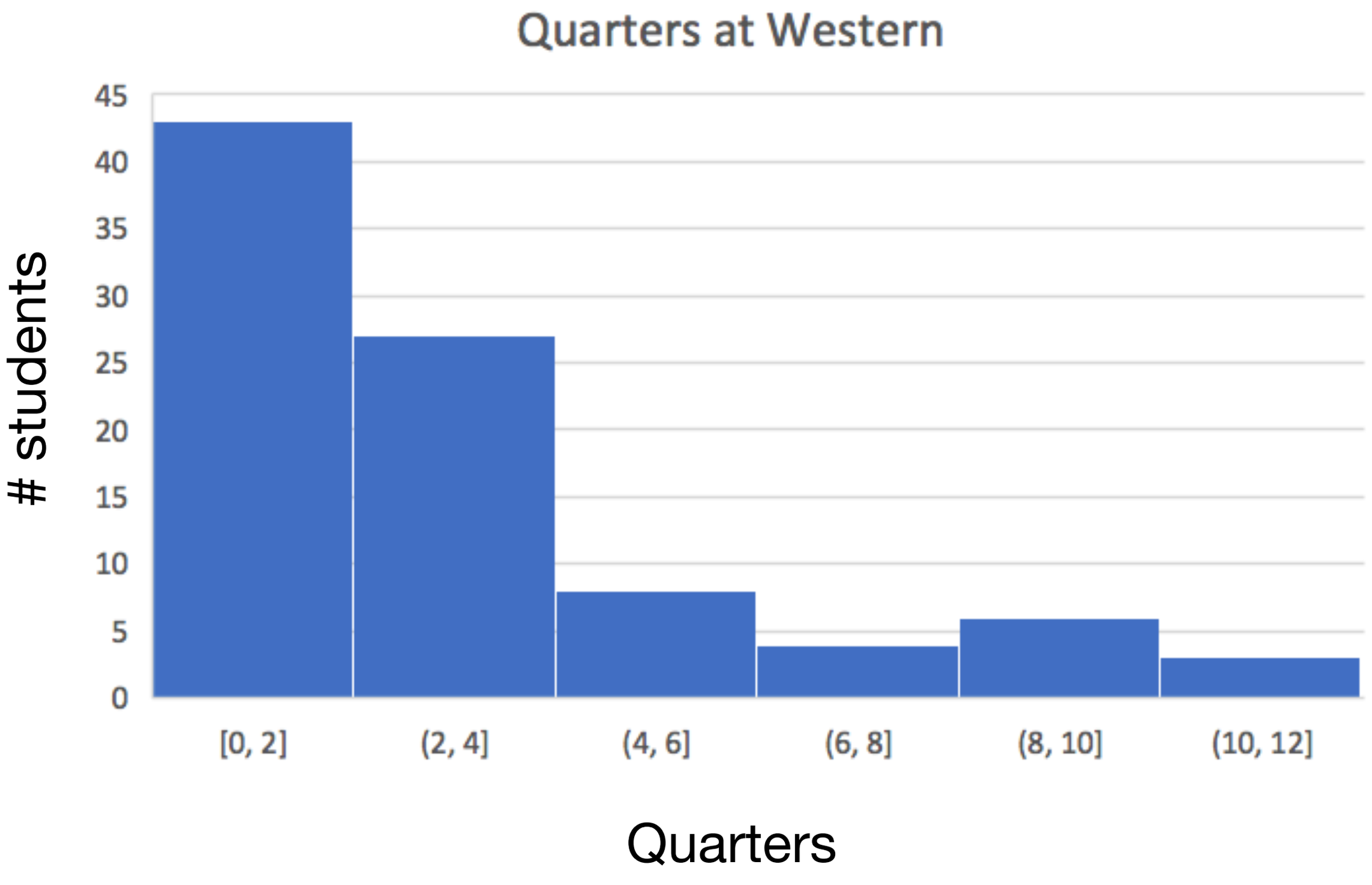
Goals (1)

- Appreciate the value of an inclusive learning environment, and the steps you can take to maintain it.
- Gain a basic understanding of the components of a computer, and how they interact:
  - Input and output devices
  - Central Processing Unit
  - Storage
  - Programs

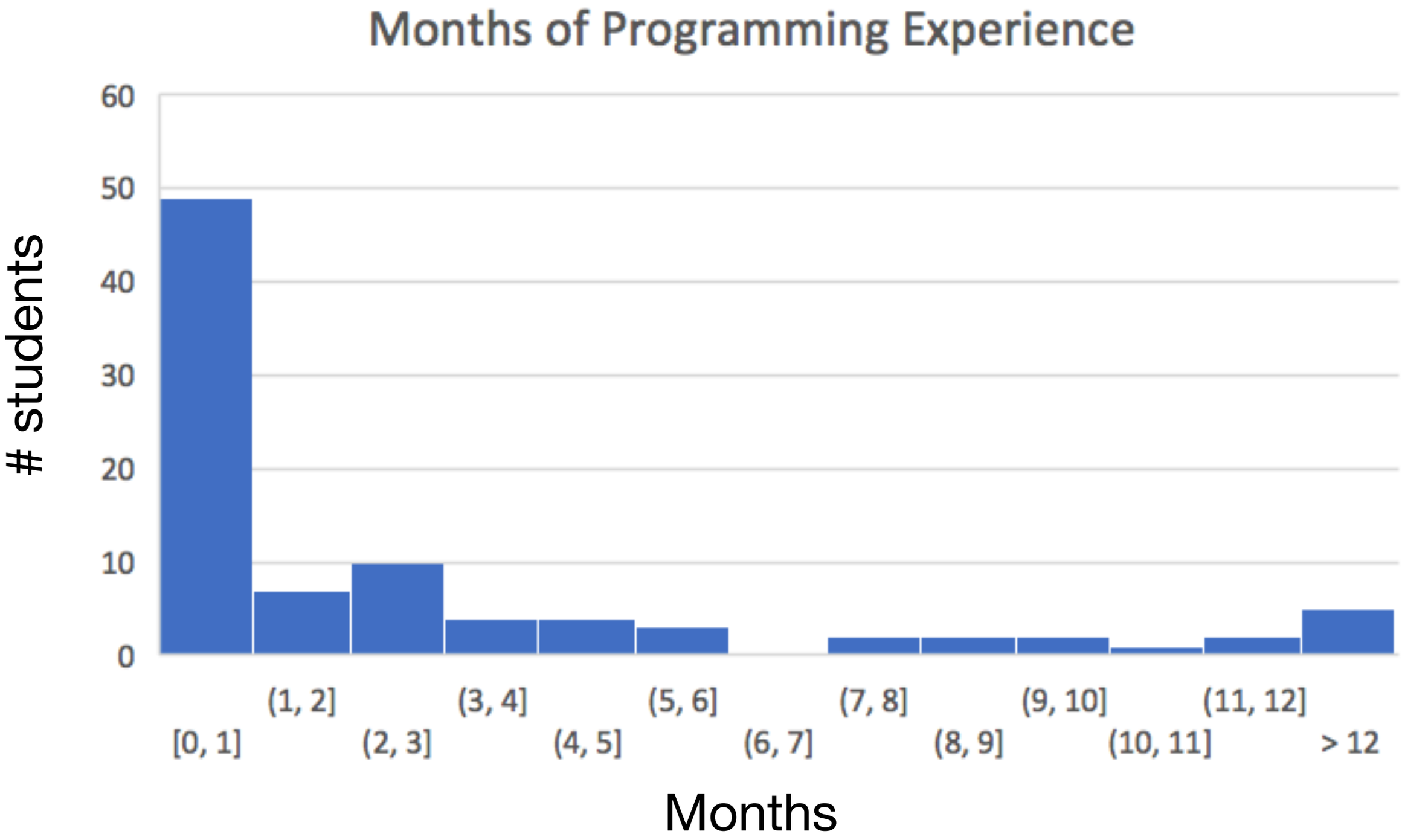
Goals (2)

- Know the definition and purpose of **algorithms** and **pseudocode** and how they fit into the software development process.
- Know the syntax used to to call a function with or without arguments.
- Understand the behavior of the `print` function with multiple arguments.
- Know how to use the `input` function to pause a program.

QOTD Survey Results



QOTD Survey Results



QOTD Survey Results

Planning to major?	
Yes	30
No	40
Considering	16

QOTD Survey Results: Hobbies

Inclusive Learning Environment

Inclusive Learning Environment

- There's a lot of variation among people who take this class.
  - Prior programming experience
  - Age, Gender, Race/ethnicity
  - 1st-generation college students
  - Goals for what you want to get out of this class

Inclusive Learning Environment

- There's a lot of variation among people who take this class.
  - Prior programming experience
  - Age, Gender, Race/ethnicity
  - 1st-generation college students
  - Goals for what you want to get out of this class
- Varied experiences, varied strengths, and varied perspectives lead to better solutions to problems!

Inclusive Learning Environment

- There's a lot of variation among people who take this class.
  - Prior programming experience
  - Age, Gender, Race/ethnicity
  - 1st-generation college students
  - Goals for what you want to get out of this class
- Varied experiences, varied strengths, and varied perspectives lead to better solutions to problems!
- Notice that not all of the above characteristics are immediately apparent.

Inclusive Learning Environment

- My goal: A learning environment in which everyone feels comfortable, curious, and excited to learn.
- **Anyone know how to ride a bike?**

Inclusive Learning Environment

- My goal: A learning environment in which everyone feels comfortable, curious, and excited to learn.

- My ideal outcome from this course:

Inclusive Learning Environment

- My goal: A learning environment in which everyone feels comfortable, curious, and excited to learn.

- My ideal outcome from this course:

Remember what it was like to learn?

Inclusive Learning Environment

- My goal: A learning environment in which everyone feels comfortable, curious, and excited to learn.

- My ideal outcome from this course:

What are the steps to making this happen?

Inclusive Learning Environment

- Recipe for success:

Inclusive Learning Environment

- Recipe for success:

Inclusive Learning Environment

- Recipe for success:

Inclusive Learning Environment

- Recipe for success:

Inclusive Learning Environment

- Recipe for success:

Inclusive Learning Environment

- Recipe for success:

Inclusive Learning Environment

- Recipe for success:

Inclusive Learning Environment

- Recipe for success:

Inclusive Learning Environment

- Recipe for success:

Inclusive Learning Environment

- What does this

have to do with an inclusive learning environment or computer science?

Inclusive Learning Environment

- My goal: A learning environment in which everyone feels comfortable, curious, and excited to learn.
  - You learn by **doing**.
  - This involves **making mistakes** and **asking questions**.
  - **Nobody** writes perfect code on the first try, not even professionals.
- Keep this in mind when:

Inclusive Learning Environment

- My goal: A learning environment in which everyone feels comfortable, curious, and excited to learn.
  - You learn by **doing**.
  - This involves **making mistakes** and **asking questions**.
  - **Nobody** writes perfect code on the first try, not even professionals.

- Keep this in mind when:

This is you.

Inclusive Learning Environment

- My goal: A learning environment in which everyone feels comfortable, curious, and excited to learn.
  - You learn by **doing**.
  - This involves **making mistakes** and **asking questions**.
  - **Nobody** writes perfect code on the first try, not even professionals.
- **Also** keep this in mind when:

Inclusive Learning Environment

- My goal: A learning environment in which everyone feels comfortable, curious, and excited to learn.
  - You learn by **doing**.
  - This involves **making mistakes** and **asking questions**.
  - **Nobody** writes perfect code on the first try, not even professionals.

- **Also** keep this in mind when:

This is you.

Inclusive Learning Environment

Inclusive Learning Environment

- A key computer science skill: empathy.

Inclusive Learning Environment

- A key computer science skill: empathy.
  - Empathize with the stupid computer

Inclusive Learning Environment

- A key computer science skill: empathy.
  - Empathize with the stupid computer
  - Empathize with other programmers reading your code

Inclusive Learning Environment

- A key computer science skill: empathy.
  - Empathize with the stupid computer
  - Empathize with other programmers reading your code
  - Empathize with your peers and understand that they learn in their own way, at their own pace.

Inclusive Learning Environment

- A key computer science skill: empathy.
  - Empathize with the stupid computer
  - Empathize with other programmers reading your code
  - Empathize with your peers and understand that they learn in their own way, at their own pace.
- Try to keep this in mind in the classroom, in labs, in the hallways, and in general.

One more thing about me

One more thing about me

One more thing about me

This is only my second time teaching CSCI 141!

One more thing about me

This is only my second time teaching CSCI 141!

And I'm overhauling it to try to make it better, so much of what I'm doing is brand new!

Getting Stuck and Getting Un-Stuck

Getting Stuck and Getting Un-Stuck

- At some point when programming, you will probably get stuck.

Getting Stuck and Getting Un-Stuck

- At some point when programming, you will probably get stuck.

- Ideal case: you puzzle through the problem, refer to your notes, the slides, or the textbook, and you independently arrive at that “ah-ha!” moment.

Getting Stuck and Getting Un-Stuck

- At some point when programming, you will probably get stuck.

- Ideal case: you puzzle through the problem, refer to your notes, the slides, or the textbook, and you independently arrive at that “ah-ha!” moment.

- Common case: half hour later you’re no less confused; maybe you don’t even know what question to ask. **This is when you should get help.**

Getting Stuck and Getting Un-Stuck

- Ways to get help when you're stuck:
  - My office hours and TA office hours (see the webpage)
  - CS mentor hours: 4:00pm-7:00pm in CF 162/164.
  - Piazza - an online Q&A forum for students in this class. Details to be announced next week.
- This only works if you have time between now and the deadline.
- Don't underestimate the programming assignments: start early.

Last time: Hello, world!

Last time: Hello, world!

- **Python** is our chosen programming language in this course.

Last time: Hello, world!

- **Python** is our chosen programming language in this course.

- A programming language is a language a computer can “understand” and execute (more on what this means next time)

Last time: Hello, world!

- **Python** is our chosen programming language in this course.

- A programming language is a language a computer can “understand” and execute (more on what this means next time)

- We’ll use a program called **Thonny** to write our Python code.

Last time: Hello, world!

- **Python** is our chosen programming language in this course.

- A programming language is a language a computer can “understand” and execute (more on what this means next time)

- We’ll use a program called **Thonny** to write our Python code.

- Thonny is an example of an “Integrated Development Environment” (IDE): a program that provides all the features you need to write, run, and fix errors in programs.

Last time: Hello, world!

Our first Python program:

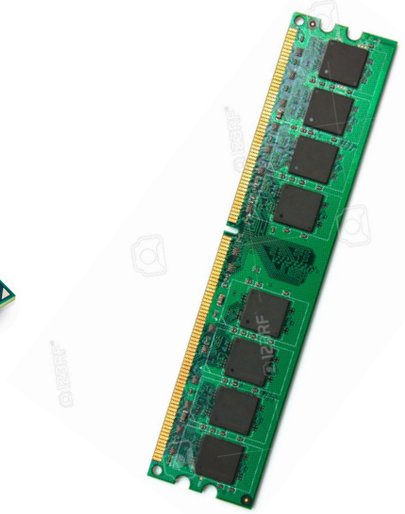
```
# Author: Scott Wehrwein
# Date: 9/25/2019
# Description: A program that prints
# "Hello, World!" to the screen.

print("Hello, World!")
```

What just happened?

- A lot! This course won't get into the details.
- A simple model of a computer:

Input Devices

CPU



Main Memory

Secondary Storage

Output Devices

Hardware

- A simple model of a computer:

Input Devices

Supply input from a user to the computer.

Hardware

- A simple model of a computer:

Output Devices Transmit information back to the user.

Hardware

- A simple model of a computer:

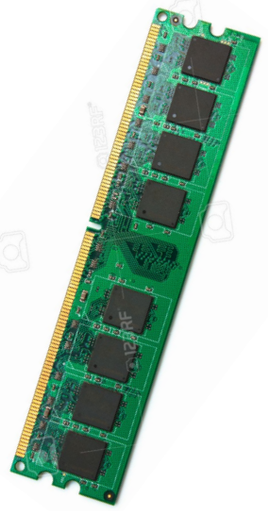
CPU:
Central Processing Unit

Executes instructions to run computer programs.

Hardware

- A simple model of a computer:

Short-term storage:
Does not persist when the computer is turned off or the program quits.

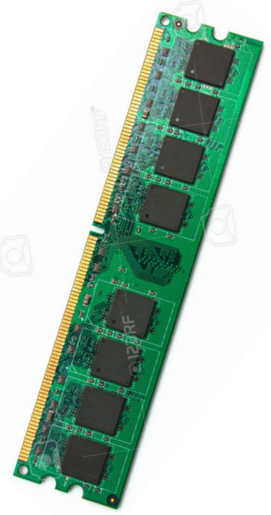


Main Memory

Hardware

- A simple model of a computer:

Short-term storage:
Does not persist when the computer is turned off or the program quits.



Main Memory

also known Random Access Memory (RAM)

Hardware

- A simple model of a computer:

Long-term information storage:
Stays around even if computer is off, or if program quits.

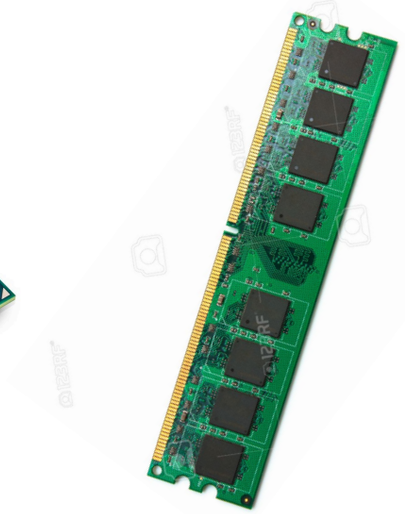
Secondary Storage

Hardware

- A simple model of a computer:

Input Devices

CPU



Main Memory

Secondary Storage

Output Devices

Socrative

is a tool for collecting instant feedback in class.

Login instructions:

1. Go to www.socrative.com
2. Click "Login" then "Student Login"
3. Enter "1pm141" for the Room Name
4. Enter your WWU username (e.g., wehrwes; not your W#)
5. Pick "D" so I can see that you've gotten it to work.

Socrative

is a tool for collecting instant feedback in class.

Not working? Don't worry.

- Today's polls don't count towards the poll grade.
- Email me and I'll make sure you're on the roster

Login instructions:

1. Go to www.socrative.com
2. Click "Login" then "Student Login"
3. Enter "1pm141" for the Room Name
4. Enter your WWU username (e.g., wehrwes; not your W#)
5. Pick "D" so I can see that you've gotten it to work.

Socrative Practice

The instructor of this course prefers that you address him as:

A. Professor Wehrwein

B. Scott

C. Dr. Wehrwein

D. Dude

CPU

CPU stands for:

A. Coronary Pulse Upkeep

B. Critical Process Undertaker

C. Computer Process User

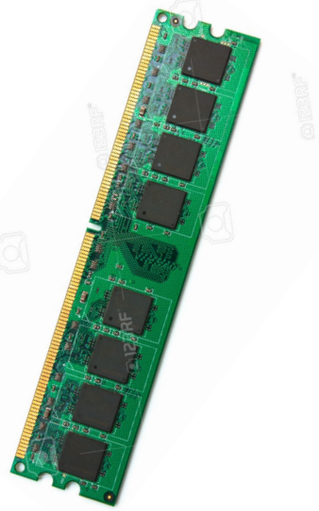
D. Central Processing Unit

What can computers do?

- Run programs (software).

Input Devices

CPU



Main Memory

Secondary Storage

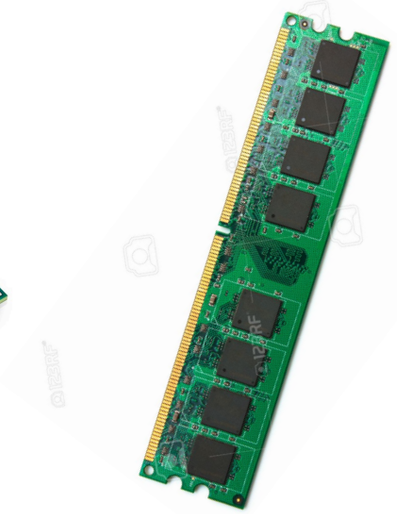
Output Devices

What can computers do?

- Run programs (software).
- **That's it!**

Input Devices

CPU



Main Memory

Secondary Storage

Output Devices

How do computers run programs?

CPU

Executes instructions to run computer programs.

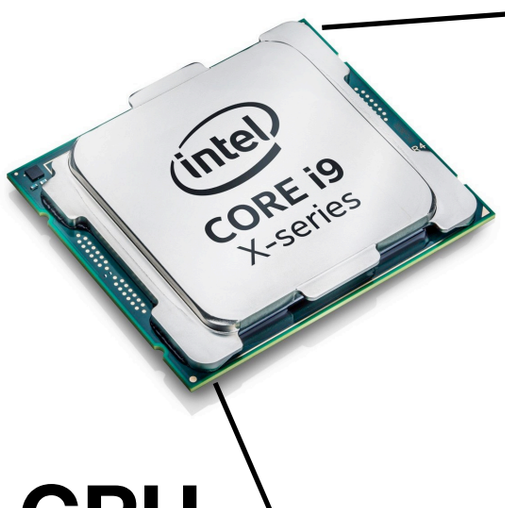
How do computers run programs?

Let’s take a closer look…

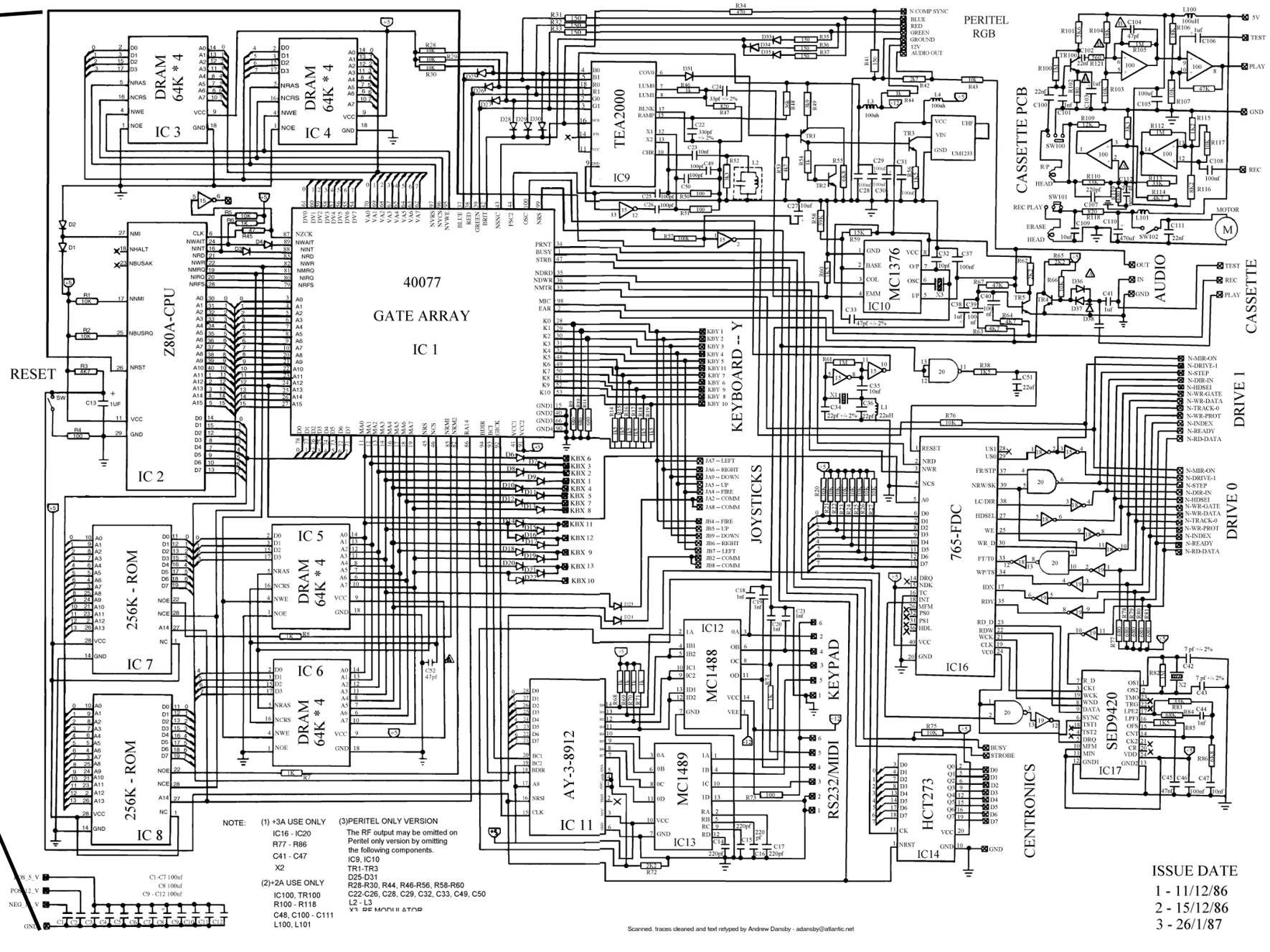
CPU

How do computers run programs?

Let’s take a closer look…



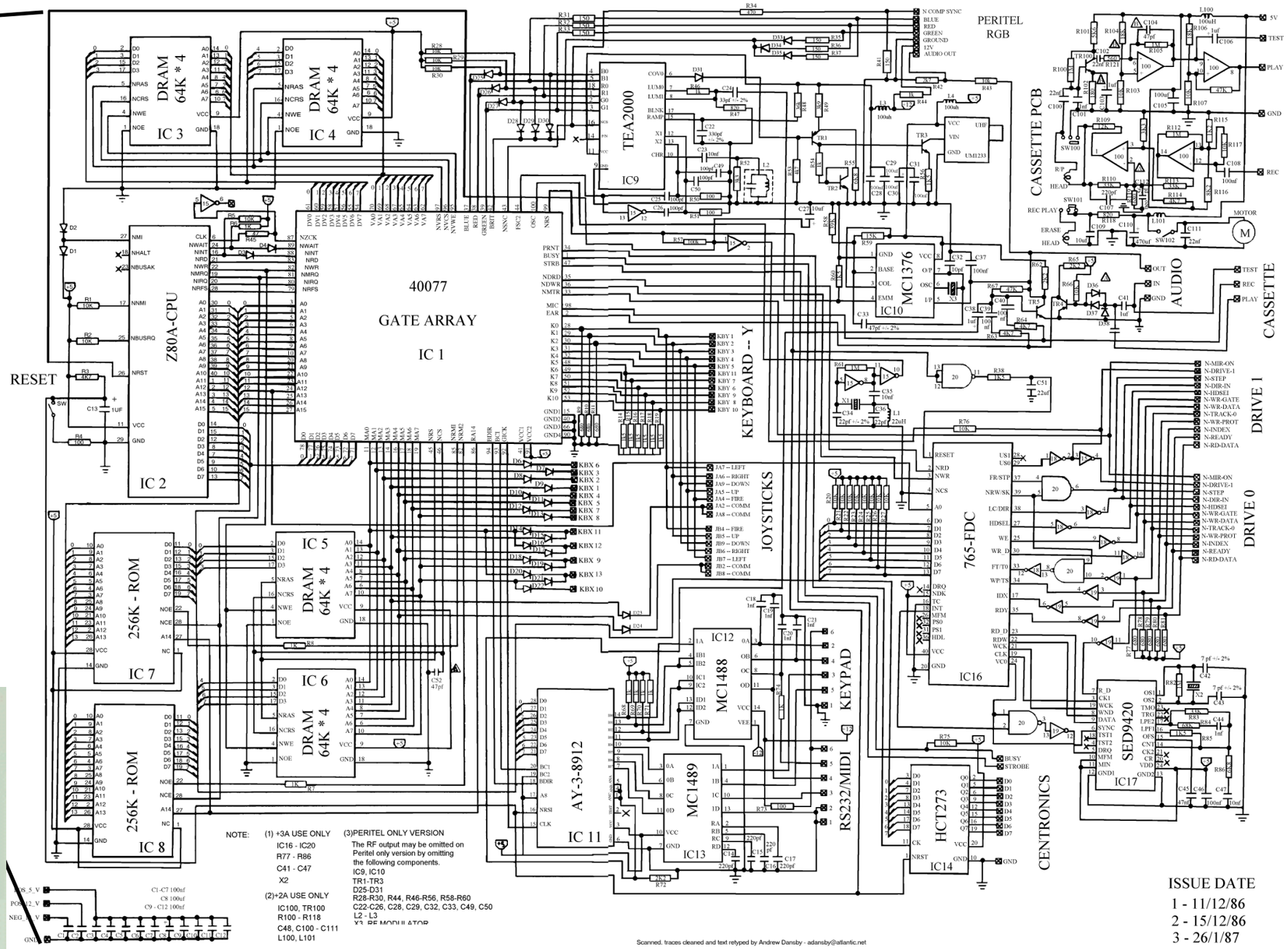
CPU



How do computers run programs?

Let’s take a closer look…

CPU



How do computers run programs?

Let’s take a closer look…

CPU

How do computers run programs?

Let’s **not** take a closer look.

CPU

How do computers run programs?

Let's **not** take a closer look.

CPU

We don't need to know the hardware details!
This is an example of **abstraction**.

In brief...

In brief...

- Your code is translated into simpler code

In brief...

- Your code is translated into simpler code
- The simpler code is translated into even simpler code

In brief...

- Your code is translated into simpler code
- The simpler code is translated into even simpler code
- and so on...

In brief...

- Your code is translated into simpler code
- The simpler code is translated into even simpler code
- and so on...
- ...until the instructions are so "simple"
  that an electronic circuit can do it

In brief...

- Your code is translated into simpler code
- The simpler code is translated into even simpler code
- and so on...
- ...until the instructions are so "simple"
  that an electronic circuit can do it

- The "simple" instructions are stored in **main memory**
- All the CPU does is:
  1. Fetch the next instruction from memory and "decode" it
  2. Execute it

In brief...

- Your code is translated into simpler code
- The simpler code is translated into even simpler code
- and so on...
- ...until the instructions are so "simple"
  that an electronic circuit can do it

- The "simple" instructions are stored in **main memory**
- All the CPU does is:
  1. Fetch the next instruction from memory and "decode" it
  2. Execute it

Examples of such "simple" instructions:

- Copy a piece of data from memory into the CPU
- Do arithmetic on pieces of data in the CPU
- Copy a piece of data from the CPU to memory

How do computers run programs?

Consider a program that performs the following tasks:

- Multiply 3 by 4
- Add 2 to the result
- Print the final result to the screen.

Here are the steps that might get translated to:

How do computers run programs?

Consider a program that performs the following tasks:

- Multiply 3 by 4
- Add 2 to the result
- Print the final result to the screen.

Here are the steps that might get translated to:

- Load 3 into CPU slot A
- Load 4 into CPU slot B
- Multiply CPU slot A by CPU slot B
- Store the result in CPU slot A
- Load 2 into CPU slot B
- Add CPU slot A to slot B
- Store the result in slot A
- Print the value in slot A

Our Simple Program

Multiply 3 by 4
Add 2 to the result
Print the final result to the screen.

Is this a Python program?

Our Simple Program

Multiply 3 by 4
Add 2 to the result
Print the final result to the screen.

Is this a Python program?

Let's find out...

Our Simple Program

Multiply 3 by 4
Add 2 to the result
Print the final result to the screen.

Is this a Python program?

No!

Algorithms

Multiply 3 by 4
Add 2 to the result
Print the final result to the screen.

Is this a Python program?

No, but it is an **algorithm**.

An algorithm is a sequence of steps that solve a problem.

Remember from last time...

Problem solving and software engineering

Correct Python syntax

Remember from last time...

Problem solving and software engineering

Designing an algorithm: what sequence of steps?

Correct Python syntax

Remember from last time...

Problem solving and software engineering

Designing an algorithm: what sequence of steps?

Correct Python syntax

Implementing an algorithm: writing the steps in Python.

Pseudocode

Problem solving and software engineering

Designing an algorithm: what sequence of steps?

Pseudocode

Problem solving and software engineering

Designing an algorithm: what sequence of steps?

Ignore Python syntax: describe the steps in English or pseudocode.

Pseudocode

Problem solving and software engineering

Designing an algorithm: what sequence of steps?

Ignore Python syntax: describe the steps in English or pseudocode.

Pseudocode is a halfway-point between English and Python. Think of it as an **informal but precise*** description of an algorithm.

Pseudocode

Problem solving and software engineering

Designing an algorithm: what sequence of steps?

Ignore Python syntax: describe the steps in English or pseudocode.

Pseudocode is a halfway-point between English and Python. Think of it as an **informal but precise*** description of an algorithm.

*For our purposes:
precise enough that a programmer could translate it into Python.

Pseudocode

Pseudocode:

Multiply 3 by 4
Add 2 to the result
Print the final result to the screen.

Pseudocode

Pseudocode:

Multiply 3 by 4
Add 2 to the result
Print the final result to the screen.

Python implementation:

Pseudocode

Pseudocode:

Multiply 3 by 4
Add 2 to the result
Print the final result to the screen.

Python implementation:

```
print(3 * 4 + 2)
```

Function Calls

```
print("Hello, world!")

print(3 * 4 + 2)

print("Hello", 3 * 4 + 2)
```

Function Calls

```
print("Hello, world!")

print(3 * 4 + 2)

print("Hello", 3 * 4 + 2)
```

Each of these lines makes a call to the `print` function.

Function Calls

```
print("Hello, world!")

print(3 * 4 + 2)

print("Hello", 3 * 4 + 2)
```

Each of these lines makes a call to the `print` function.

What exactly is a function? More on this later.

Function Calls

```
print("Hello, world!")

print(3 * 4 + 2)

print("Hello", 3 * 4 + 2)
```

Each of these lines makes a call to the `print` function.

What exactly is a function? More on this later.

For now: it's a thing that does stuff for us.

Function Calls

```
print("Hello, world!")

print(3 * 4 + 2)

print("Hello", 3 * 4 + 2)
```

Each of these lines makes a call to the `print` function.

What exactly is a function? More on this later.

For now: it's a thing that does stuff for us.

(We make it do stuff for us by calling it.)

Function Calls

```
print("Hello, world!")

print(3 * 4 + 2)

print("Hello", 3 * 4 + 2)
```

Each of these lines makes a call to the `print` function.

What exactly is a function? More on this later.

For now: it's a thing that does stuff for us.

(We make it do stuff for us by calling it.)

Here's another example of a function call:

Function Calls

```
print("Hello, world!")

print(3 * 4 + 2)

print("Hello", 3 * 4 + 2)
```

Each of these lines makes a call to the `print` function.

What exactly is a function? More on this later.

For now: it's a thing that does stuff for us.

(We make it do stuff for us by calling it.)

Here's another example of a function call:

```
input()
```

Function Calls

```
print("Hello, world!")

print(3 * 4 + 2)

print("Hello", 3 * 4 + 2)
```

Each of these lines makes a call to the `print` function.

What exactly is a function? More on this later.

For now: it's a thing that does stuff for us.

(We make it do stuff for us by calling it.)

Here's another example of a function call:

```
input()
```

Let's see if we can figure out what stuff it does...

Demo: print and input

Demo: print and input

- the Shell pane in Thonny
- `print(3 * 4 + 2)`
- Print with multiple arguments
  - A space is printed between each
- input() to pause the program