Why computational thinking?
Problem Solving by Pattern Matching

- Critical skill, not just in computer science
- Break the problem down
  - What are you trying to solve?
  - What do you know/have?
  - Do you know what you don’t know?
    - How do you find out what you don’t know?
  - Do you notice any patterns?
    - After solution, can you simplify/optimize the solution further?
Examples

- Helping my son with math homework
- Assembling a wheelbarrow
- Building a house in Minecraft
My Story
Tools to help teach computational thinking
Software Resources

• MIT Scratch
  • http://scratch.mit.edu
Software Resources

• MIT Scratch
  • https://scratch.mit.edu

• Code.org
  • https://code.org
Software Resources

- **MIT Scratch**
  - https://scratch.mit.edu
- **Code.org**
  - https://code.org
- **Alice**
  - https://www.alice.org
What’s the difference?

- **MIT Scratch**
  - More open ended
  - Community based
- **Code.org**
  - Aligned with Common Core
  - Step by Step
  - Hour of Code
- **Alice**
  - Focuses more on visual and interactive
  - Not as widely used as other two
Hardware Resources

- Lab computers, laptops, & tablets
Hardware Resources

- Lab computers, laptops, & tablets
- Arduinos
  - https://www.arduino.cc
Hardware Resources

- Lab computers, laptops, & tablets
- Arduinos
  - https://www.arduino.cc
- BeagleBone
  - https://beagleboard.org/bone
Hardware Resources

• Lab computers, laptops, & tablets
• Arduinos
  • https://www.arduino.cc
• BeagleBone
  • https://beagleboard.org/bone
• PINE64
  • https://www.pine64.org
Hardware Resources

- Lab computers, laptops, & tablets
- Arduinos
  - https://www.arduino.cc
- BeagleBone
  - https://beagleboard.org/bone
- PINE64
  - https://www.pine64.org
- Raspberry Pi
What’s the difference?

- **BeagleBone, PINE64, & Raspberry Pi**
  - Full fledged computers
  - Run an OS
  - Programmed with many different languages
  - More general purpose
  - More easily use networks
  - Large amount of RAM

- **Arduino**
  - Microcontroller
  - No OS, what you “flash” on it is the only thing that runs
  - Great for “real time” applications
  - Low power & can be tiny

- **All have General Purpose I/O (GPIO)**
  - Sensors, motors, relays
Why the Raspberry Pi?

• It’s cheap: $35
• Works with common components
  • TV, keyboard, mouse, wireless, Bluetooth
• Updated versions regularly
  • Faster, more RAM, better I/O, etc.
• Flexible
  • Runs Windows & Linux
• Huge user community
  • Many existing projects and examples
• MagPi
  • Free to download monthly magazine
• Lots of accessories
  • Cameras, LCDs, sensors, cases
Introduction to the Raspberry Pi
What do you need to get started?

- HDMI monitor or TV
- HDMI cable
- USB keyboard and mouse
- 8GB+ micro SD card
- SD card reader (your laptop may have one built in)
- OS image
- Software to write to the SD card
  - https://etcher.io/
- Micro USB power adapter
  - Many cell phone chargers will work
  - Make sure it is at least rated for 2.5A output
  - If in doubt, buy a UL rated one for a Raspberry Pi 3
Raspbian OS - Desktop
Raspbian OS – Programming Menu
Raspbian OS – Terminal

Terminal is command line interface (CLI) to Linux

Let’s you give OS commands via text

Many documents prepend CLI commands with a ‘$’:

$ ls

ls is the command to list the files in the current directory
Useful Linux CLI commands

- `ls`: List files in the current directory
- `cd`: Change to a new directory
- `mv`: Move/rename a file
- `rm`: Remove/delete a file
- `man`: Read manual pages about commands
- `nano`: CLI text editor
- `sudo`: Run commands with elevated privileges (run as root)

- Commands, their options, and their arguments are case sensitive
Linux Software

• There is a lot of freely available Linux software
• Many of the popular software packages are available in Raspbian
  • apt-cache: Search for packages
    • $ apt-cache search apt-file
  • apt-get: Install, update, delete packages
    • $ sudo apt-get install apt-file
    • $ sudo apt-get update
    • $ sudo apt-get upgrade
  • apt-file: Find what package provides a file
    • $ apt-file search pip
  • dpkg: Show information about packages
    • $ dpkg -l
    • $ dpkg -L python-pip
Raspbian OS – raspi-config

$ sudo raspi-config

Change the Locale under Localisation Options from en.GB.UTF-8 UTF-8 to en.US.UTF-8 UTF-8
Raspbian OS – Update Keyboard Layout
Raspbian OS – Update Keyboard Layout
Raspbian OS – Update Keyboard Layout
Programming 101
What is a computer?

A computer is a general purpose device that can be *programmed* to carry out a set of *arithmetical* or *logical* operations automatically.

Computers have:
- Input
- Output
- Storage
- Processing
What is a program?

A recipe of arithmetic and logical operations in specific order.

This recipe is called an **algorithm**.

We’ll explore arithmetic and logical operations later.

---

Scrambled Eggs:
Mixture = 2 eggs + salt + pepper

IF Mixture is hot AND mixture is cooked THEN eat
Human languages

- Let's write "Good Morning!" in 3 different languages: English, Spanish, Japanese
- English: Good Morning!
- Spanish: ¡Buenos Días!
- Japanese: おはよう!
Why computer languages?

• International scientists communicate with each other in English

• Similarly, if we want computers to understand what we want them to do, we must write our programs using a *computer language*

• A *computer language* allows humans to communicate with computers in a meaningful way
Many computer languages

• Just like there are many human languages, there are many computer languages as well

• Let's write a "Hello, world!" program in 4 different languages: Pseudo code, C, Python, JavaScript
Pseudo Code

Display “Hello, world!”
C

#include <stdio.h>

int main() {
    printf("Hello, world!");
    return(0);
}

Javascript

document.write(“Hello, world!”);
Python

```python
print("Hello, world!")
```
An example Python program

```python
import random
num = random.randint(1, 100)

while True:
    print('Guess a number between 1 and 100')
    guess = input()
    i = int(guess)
    if i == num:
        print('You guessed right')
        break
    elif i < num:
        print('Try higher')
    elif i > num:
        print('Try lower')
```
Let’s talk to the computer

Open IDLE for Python 3 and write a “Hello, world!” program.
Let’s talk to the computer

Python shell

Prompt: >>>
Let’s talk to the computer

• Make sure to include double quotes
• Press enter when done

```python
>>> print("Hello, World!")
Hello, World!
>>> 
```
Let’s talk to the computer

1. From Menu: File -> New File
2. Type in the previous code
3. File -> Save Name it hello.py
4. Run -> Run module
Binary

• On/Off, High/Low, Open/Closed, True/False, 1/0
• True/False values also referred to as Booleans
• Everyone knows how to count to 12 in decimal

  1

• Let’s count to 12 in binary

  0 0 0 0
Why Python?

- Python is a language that was designed to be easy to read and use fewer symbols (!#$*)

**Python**

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

**C++**

```cpp
int main()
{
    std::cout << "Hello, world!" << std::endl;
    return(0);
}
```
English Grammar

• Let’s eat Grandma!
  vs.
• Let’s eat, Grandma!

• In english how do you end a question.
Programming Language Syntax

• Just like grammar, it helps you read a sentence.
• Language has a syntax that lets the computer read your program.
• If you get the syntax wrong, the computer will have an error running your program.
• Python is case sensitive.
Working with data in Python
Arithmetic Operators

- +, -, *, /, (, )
  - 1+1
  - 1+2*4
  - (1+2)*4
  - 1+
  - 2/0

Order of operations
- Computers do exactly what you tell them in the order you tell them.
- Be explicit.
Logical Operators

• $>, <, \geq, \leq, ==, !=$
  • $2 > 1$
  • $2 > 4$
  • $2 \geq 2$
  • $2 == 2$
  • $2 == 1$
  • $2 != 1$

Booleans

• Just like arithmetic operators manipulate numbers, logical operators manipulate Booleans.

• Logical operations return a Boolean for the answer, instead of a number
Logical Operators

• and, or, not
  • True and True
  • True and False
  • True or False
  • not True
  • 2<4 and 1<=2

Booleans

• Just like arithmetic operators manipulate numbers, logical operators manipulate Booleans.
• Logical operations return a Boolean for the answer, instead of a number
Variables

- Variables are like containers
- Examples

```python
>>> fred = 100
>>> print(fred)
100
```

Variable Names

- Letters
- Numbers
- _ (underscore)
Using Variables

• Let’s try the following code

```python
>>> found_coins = 20
>>> magic_coins = 10
>>> print(found_coins)
```

• What should be the output value?

• Let’s now add the following line. What would be the output?

```python
>>> print(found_coins + magic_coins)
```
What are Data Types?

Examples:

- **String**: “I am 1 string”, “What’s up?”, “cheese”
- **Integer**: 10, -42, 12000, 9
- **Float**: 3.2, 0.00001, -10900.999, 123.456
- **Boolean**: True, False
What are Data Types? (and why are they important?)

- Remember when we talked about binary?
- Computer only understands 0 and 1
- We need to tell it how to interpret them

Example

- 30 : 11110
- “30” : 00110011 00110000
Controlling program flow
Counting Numbers

Let’s count natural numbers:

- 1 2 3 4 5 6 ....

Now let’s try using Python sentences:

```python
print(1)
print(2)
print(3)
print(4)
... 
```

and so on
Counting Numbers with a Loop

There is a common construct in almost all computer languages called *for loop*

```python
for x in range(1, 6):
    print(x)
```

Result:

1
2
3
4
5
Making Choices with Conditionals

- Remember logical operators?
  - >, <, >=, <=, ==, !=
- Use these to make comparisons:

```python
if (some condition is True):
    do something
    do another thing
else:
    do something different
    do more stuff
```
Checking Numbers

Let’s combine loops and conditionals

```python
for x in range(1, 7):
    if (x<3):
        print(“Not close”)
    elif (x<=5)
        print(“Almost there”)
    else:
        print(x)
```

Result:

Not close
Not close
Almost there
Almost there
Almost there
6
Code Reuse
Let’s Recycle (Programmer’s are lazy)

- Reuse common code
  - Only need to figure out how to do something once
  - Don’t have to type the same code over and over
  - Makes your code cleaner and shorter (easier to read)
  - Reduces errors
- Multiple ways in Python (and other languages)
  - Functions
  - Modules or Libraries
Functions and Modules

- **Functions** are like tools that you can use again and again

- **Modules** are like toolboxes, used to hold related tools
Functions

- We used them a bunch already
  - `print()`
  - `range()`

Defining a New Function

```
def newfunc(fname, lname):
    print("Hello, %s %s" % (fname, lname))
```
Functions

- A function is often used to return a value, using a `return` statement. For example, you could write a function to calculate how much money you were saving:

```python
def savings(pocket_money, paper_route, spending):
    return pocket_money + paper_route - spending

print( savings(10, 10, 5) )
```

Result: 15
Variables and Scope

Scope:
Scope determines when a variable is “visible” or valid

```python
def variable_test():
    first_variable = 10
    second_variable = 20
    return first_variable * second_variable

print(first_variable)
```

NameError: name 'first_var' is not defined

Why?
The variable `first_variable` is only defined within the scope of the function `variable_test()`. It does not exist outside the function.
Different Scopes

- If a variable is defined outside the function, it has a different scope

```python
another_variable = 100
def variable_test():
    first_variable = 10
    second_variable = 20
    return first_variable * second_variable

print(variable_test())
print(another_variable)
print(first_variable)
```

Result:

```
200
100
NameError: name 'first_variable' is not defined
```
Python Modules

• Just like Linux software there are a lot of available Python modules
• Many of the popular Python modules are available in Raspbian
  • NumPy: $ apt-get install python3-numpy
  • SciPy: $ apt-get install python3-scipy
  • pigpio: $ apt-get install python3-pigpio

• Discover what modules are available in Python
  >>> help()
  help> modules

• Use Python Packaging Index (PyPI) - https://pypi.org/
  • pip search numpy
  • pip install numpy
Using Modules

- **Use the `import` directive**

```python
import datetime as dt

print(dt.date.today())
```

**Result:**

2018-04-29
Specific Importing

- Use the `from` directive

```python
from datetime import date

print(date.today())
```

Result:
2018-04-29
Data Structures
Lists
Also Called Arrays

• Create a list of strings
  • Declare using []

```python
>>> food=['hotdog buns', 'coffee', 'eggs', 'orange juice']
>>> print(food[1])
```

• List index starts at 0

• Change value of entry

```python
>>> food[3]='milk'
>>> print(food)
```

Result:
```
coffee
['hotdog buns', 'coffee', 'eggs', 'milk']
```
Lists of Lists
Also Called Multi-dimensional Arrays

>>> numbers=[12, 74, 8, 506]
>>> mixed_list=[‘We’, ‘have’, 2, ‘wait’, 4, ‘dinner’]
>>> mylist=[mixed_list, numbers]
>>> print(mylist)

Result:
[[‘We’, ‘have’, 2, ‘wait’, 4, ‘dinner’],
[12, 74, 8, 506]]
Tuples

• Like lists but declare with () instead of []

```python
>>> lakes=('Huron', 'Ontario', 'Michigan', 'Erie', 'Superior')
>>> lakes[1]
```

• Cannot change values once declared

```python
>>> lakes.append('Crystal')
>>> lakes[1]='Crystal'
```

Result:

`'Ontario'`

AttributeError: 'tuple' object has no attribute 'append'

TypeError: 'tuple' object does not support item assignment
Maps
Also called Dicts or Dictionaries

- Collection of things, similar to lists and tuples
- Key : Value pairs instead of index

```python
>>> bdays={'Jim': 'May 4', 'Sue': 'April 20', 'Ed': 'July 17'}
>>> print(bdays['Sue'])
April 20
>>> bdays.update({'Sam': 'Dec 8'})
>>> print(bdays)
{'Jim': 'May 4', 'Sue': 'April 20', 'Ed': 'July 17', 'Sam': 'Dec 8'}
>>> del bdays['Ed']
>>> bdays['Sam'] = 'dec 10'
>>> print(bdays)
{'Jim': 'May 4', 'Sue': 'April 20', 'Sam': 'Dec 10'}
```
Indexing and Slicing Strings

Hint: Strings are Lists of Characters

>>> fred="Hello, Fred"

>>> print (fred[4])

• Index starts at 0

>>> print (fred[2:8])

• From index 2 up to, but not including, index 8

>>> print(fred[8:])
>>> print(fred[:5])

Result:

Hello
Loops with Lists

wizard_list = ['spider legs', 'toe of frog', 'snail tongue', 'bat wing', 'slug butter', 'bear burp']
for i in wizard_list:
    print(i)

“For each item in wizard_list, store the value in the variable i, and then print the contents of that variable”

Result:
spider legs
toe of frog
snail tongue
bat wing
slug butter
bear burp
Working with a Team
Mojang (Minecraft)

- Markus “Notch” Persson
- Has anyone heard of Minecraft?
- Worth over $1.3 Billion
Mojang Jobs

• Developer
• Artist
• Architect
• Designer
• Project Manager
• Customer Support
• DevOps Engineer
Facebook

• Mark Zuckerberg, CEO
• Worth $66.4 Billion
Facebook Employees

Data Credit: Statista
https://bit.ly/2Kn8FrW
Teams

- So what do all these people do?
- Why can’t just one person do everything?
- So how do you work on a team?
  - Agree to and document standards
  - Version Control – git, svn, mercurial, etc.
  - Comment your code
  - Communicate

https://github.com/
https://git-scm.com/doc
Debugging

• You will make mistakes
• Your team will make mistakes
• Finding and fixing problems in your program
• As simple as adding `print` statements
• As sophisticated as an interactive debugger like `pdb`
• Effective debugging is as much an art form as a skill
Performance

• There is almost always more than one way to solve a problem
• Some ways are better in certain situations than others and worse in other situations
• You might have a correct algorithm that isn’t the right algorithm for the job
  ▪ http://www.sorting-algorithms.com/
Acknowledgement

This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy, Office of Science, under contract number DE-AC02-06CH11357.