Last Class

things we've studied

- operator
- variable
- types
- expression
- code block
- control statement
- conditional
- loops
- function
- module
- class
- exceptions

Operators

+- unary, say you have -10 + 4, the - in front of the 10 is calculated first

** power

() parentheses

* multiply

/ divide

// floor division

+ add

- subtract

Variables

Variables are labels or tags or names that you can assign to a number or to a math operation. They let you store a number or result to use later in your program.

```
>>> some_number = 4
>>> radius = 5
>>> pi = 3.14159
>>> area = pi * radius**2
```

Types

Integer : any whole number
1,2,3

Float : any number with a decimal
2.1, 3.2, 4.3

Boolean : Only two states
True , False

String : collection of characters
‘Bob’ or “Bob” or “Bob’s” or ‘Bob\’s’

List : Collection of other types or objects
[1, 2, ‘Nancy’]

Dictionary : Collection of key value pairs
{‘name’: ‘Jan’, ‘age’: 12}

Expressions

An expression is anything the that is calculated in your program and produces a result that you can store in a variable or used elsewhere in your program.

```
num = 4
sum = 3+num
pow = sum ** 8

first_name = ‘Steve’
last_name = ‘Jones’
name = first_name + ‘ ‘ + last_name

some_num = randint(1, 10)
```

Code Blocks

Python is an indented language. What this means is that sections of code that we run begin and end with an indentation. Python code blocks are typically indented with 4 spaces. Blocks of code can be found in functions, classes, if, while and for loops.

Consider this code.

```
    print("I'm out of a code block")
    if True:
        print("I'm in a block")
        number = 4
        print(number * 3)
    print("I'm out of a code block")
```

There is a code block after the if statement.

Control Statements

What are control statements?

Conditionals

Think of a boolean as True or False, yes or no.

Booleans expressions can be identified using comparison operators.

== != < > <= >=
in not in
and or

>>> small = 3

>>> small == 3
True

>>> small == 4
False

>>> ‘c’ in ‘cat’
True

>>> large = 5

>>> small == large
False

>>> small > large
False

>>> small < large
True

>>> small <= large
True

>>> small >= large
False

>>> small != large
True

>>> small == 3 and small == large
False

>>> small == 3 and small < large
True

>>> small == 3 and
(small > large or large == 5)
True

if, elif, else

```
if c == 'z':
    robot.left_rot()
elif c == 'c':
    robot.right_rot()
else:
    print('unknown command')
```

```
if item_name == item['name'] and
        (school_map[player.location[0]
        [player.location[1]].monster['health'] > 0 or
        item['attack'] < 0):
    print("\n%s shall feel the wrath of your %s.\n" % (
```

Loops

While Statement

If you need repeatedly run a code block use a loop. Like the if statement the while loop uses a comparison. If the comparison is True the code block will continue to run.

```
some_number = 0
while some_number < 100:
    some_number = some_number + 10
    print('some_number is less than 100', some_number)
print('some_number is now', some_number)
```

For Statement

Another type of loop is the for loop. The for loop does not use a comparison like the while loop. The for loop is useful when you would like to use a counter in your code block. Also useful if you would like to pull individual values from a list or collection into your code block. This is called iteration. You can use the range() function to count numbers.

```
sum = 0
for number in range(10):
    sum = sum + number
    print('number',number,'sum',sum)
```

Functions

Functions are sections of code that we can reuse throughout our program.
Functions usually are defined to do one thing really well.

```
def say_hello():
    print('Hello')
```

To run or call a function just type the name and the parantheses.

```
>>> def hello():
...     name = input('Name? ')
...     print('Hello', name)
...
>>> hello()
Name? Jack
Hello Jack
```

Parameters are values that you pass into a function from outside a function.

```
>>> def hello(name):
...     print('Hello',name)
...
>>> hello('John')
Hello John
```

Functions may return results. These results can be displayed on the screen or used in other calculations.

```
>>> def power(num, pow):
...     return num**pow
...
>>> power(2, 3)
8
```

Modules

A Python module is file that contains Python code. Modules help you group together variables, functions and classes that are related. For example you might have a module called animals that contains a list of animals. This module might also contain functions that let you do animal related things.

The import statement is used to add the code from one module to another.

You could have a module called animals.py that looks like this.

```
    animals_dict = {
        'cow': 'four legged bovine, vegan, sometimes spotted',
        'cat': 'four legged feline, omnivore, usually furry, meows alot',
        'bird': 'winged creature, noisy, tweets constantly'
    }

    def sorted_animals():
        return sorted(animals)
```

Then in my_code.py you could import specific objects from the animals.py module.

```
    from animals import animals_dict

    first = animals_dict['cat']
    print('cat -', first)
```

Class

Classes describe how objects look and behave

```
class Daredevil:
    def __init__(self):
        self.name = 'daredevil'
        self.pointy_hat = True
        self.baton = True

    def attack(self):
        print('i\'m coming')


dd = Daredevil()

print(dd.pointy_hat) dd.defend() dd.use_senses()
True
stand back
ping ping ping
```

Exceptions

Using try, except blocks We can deal with these exceptions and let our program continue.

```
# exceptions
try:
    x = 4/0
except:
    print('exception happened')
```

```
try:
    x = 4/0
except ZeroDivisionError:
    print('wat? divide by zero')
except Exception:
    print('some exception happened')
```

```
try:
    x.do_something()
except ZeroDivisionError:
    print('wat? divide by zero')
except Exception:
    print('some exception happened')
```

```
try:
    x.do_something()
except Exception as e:
    print('some exception happened')
    print(str(e))
```

class work

dictionaries

```
CIPHER_ENCRYPT = {"A": "X", "B": "Y", "C": "Z", "D
                  "K": "H", "L": "I", "M": "J", "N
                  "U": "R", "V": "S", "W": "T", "X

CIPHER_DECRYPT = {"X": "A", "Y": "B", "Z": "C", "A
                  "H": "K", "I": "L", "J": "M", "K
                  "R": "U", "S": "V", "T": "W", "U


def encrypt(message):
    new_message = ''
    for c in message:
        if c in CIPHER_ENCRYPT.keys():
            new_message += CIPHER_ENCRYPT[c]
        else:
            new_message += c
    return new_message


def decrypt(message):
    new_message = ''
    for c in message:
        if c in CIPHER_DECRYPT.keys():
            new_message += CIPHER_DECRYPT[c]
        else:
            new_message += c
    return new_message
```

Roman Cipher

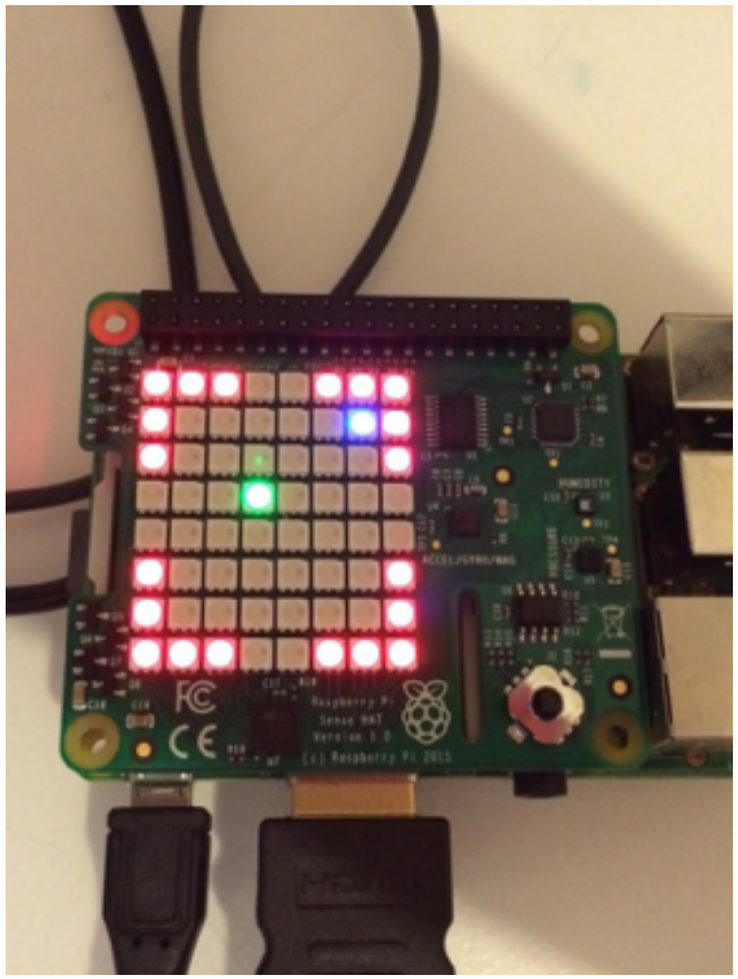
To Encrypt			To Decrypt		
A	=	X	X	=	A
B	=	Y	Y	=	B
C	=	Z	Z	=	C
D	=	A	A	=	D
E	=	B	B	=	E
F	=	C	C	=	F
G	=	D	D	=	G
H	=	E	E	=	H
I	=	F	F	=	I
J	=	G	G	=	J
K	=	H	H	=	K
L	=	I	I	=	L
M	=	J	J	=	M
N	=	K	K	=	N
O	=	L	L	=	O
P	=	M	M	=	P
Q	=	N	N	=	Q
R	=	O	O	=	R
S	=	P	P	=	S
T	=	Q	Q	=	T
U	=	R	R	=	U
V	=	S	S	=	V
W	=	T	T	=	W
X	=	U	U	=	X
Y	=	V	V	=	Y
Z	=	W	W	=	Z

```
GREETINGS EARTHLINGS
DOBBQFKDP BXOQEIFKDP
GREETINGS EARTHLINGS
```

search

binary search

```
1  from guess import guess
2
3  MIN = 0
4  MAX = 9999
5  MESSAGE = '!'
6
7
8  def binary_search():
9      minimum = MIN
10     maximum = MAX
11     tries = 0
12     while minimum <= maximum:
13         tries += 1
14         number = minimum + (maximum - minimum) // 2
15         response = guess(number, MESSAGE)
16         print(tries, maximum, minimum, number, response)
17         if response == 'correct':
18             break
19         elif response == 'too low':
20             minimum = number + 1
21         elif response == 'too high':
22             maximum = number - 1
23     print(tries, number)
24
```



sequential search

```
1  from guess import guess
2
3  for i in range(999):
4      response = guess(i, 'cookies')
5      print(i)
6      if response == 'correct':
7          print('code is: ', i)
8          break
9
```

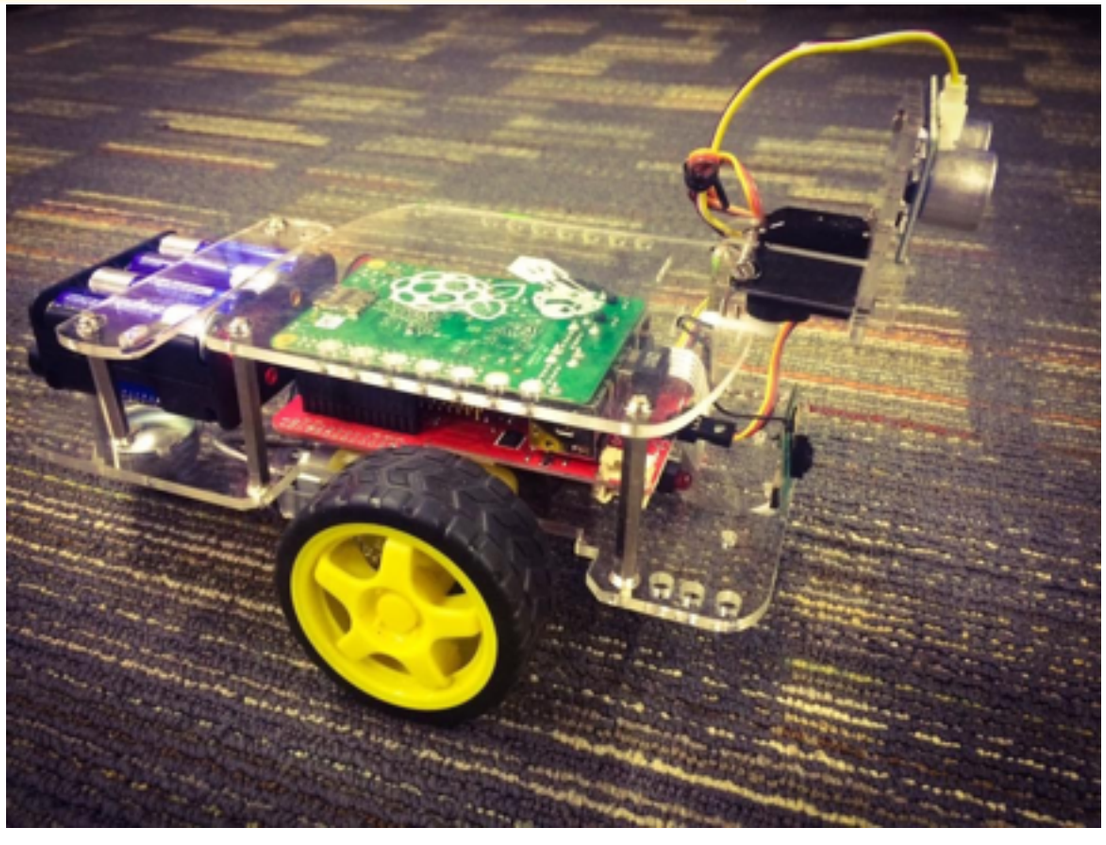
file processing

```
1  # Read one line of a text file.
2  f = open('sample.txt','r')
3  first_line = f.readline()
4  print('first line:', first_line)
5  f.close()
6
7  # Iterate through all lines of a text file.
8  f = open('sample.txt','r')
9  for line in f:
10     print('line:', line)
11 f.close()
12
13 # Read the entire file in a single text call.
14 f = open('sample.txt','r')
15 entire_file = f.read()
16 print('entire file:', entire_file)
17 f.close()
18
19 # Read using with
20 with open('sample.txt','r') as f:
21     for line in f:
22         print('with line:',line)
```

using modules

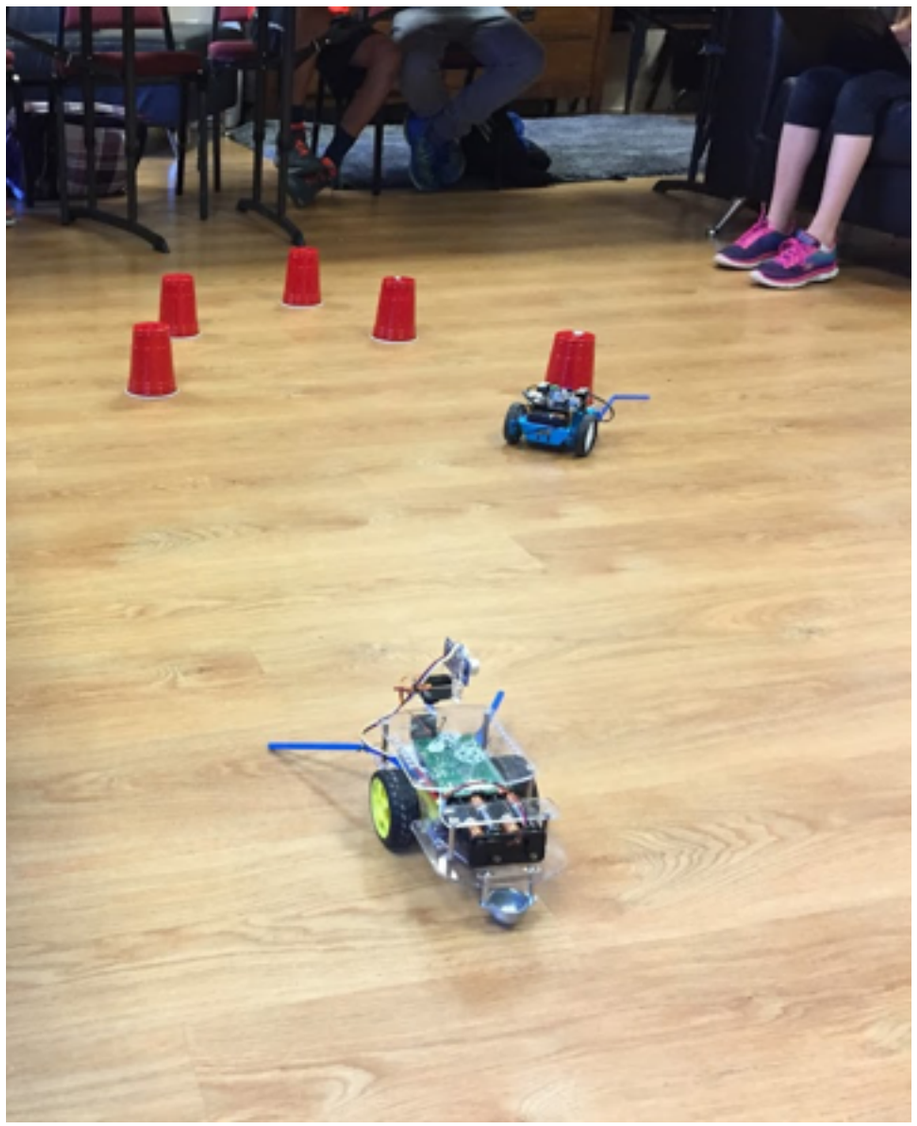
```
1  import robot
2  '''
3  We import robot. This lets us call all of the functions in robot.py.
4
5  We set robot_debug to True to test our code without sending it to the robot.
6  Set robot_debug to False when you are ready to run your code on the robot.
7  '''
8  robot.robot_debug = False
9
10 robot.robot_speed = 200
11
12 robot.forward(5.0)
13 robot.forward(1.0)
14 robot.left(1.5)
15 robot.right(1.0)
16 robot.blink(1.0)
17 robot.forward(4.5)
```

```
These are all the functions you can call in robot.

robot.backward(1.0)
robot.forward(1.0)
robot.left(1.0)
robot.right(1.0)
robot.left_rot(1.0)
robot.right_rot(1.0)
robot.stop(1.0)
robot.blink()
d = robot.distance()
v = robot.volt()
```



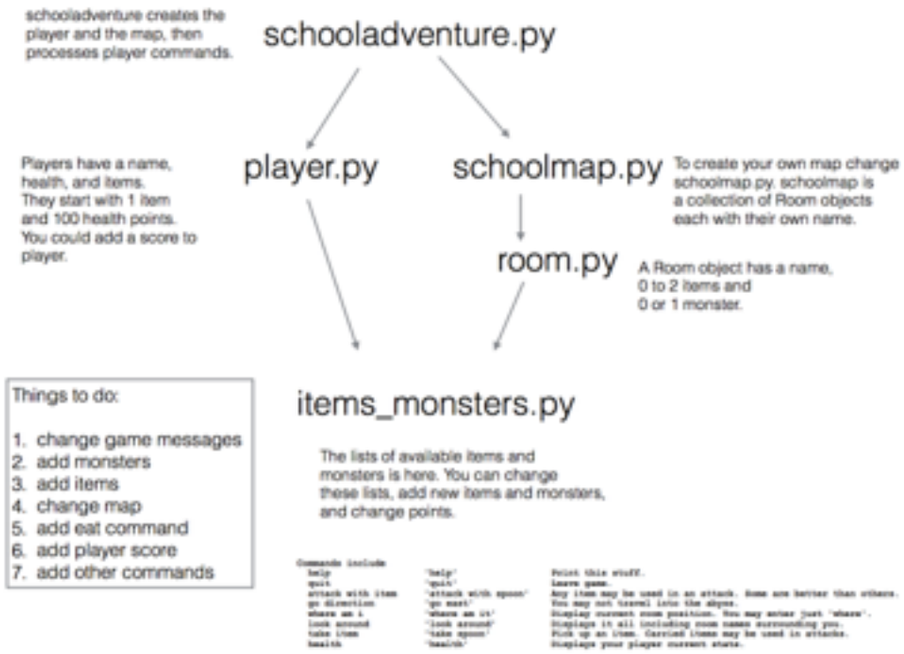
processing input

```python
import dragonbot as robot
#import gopibot as robot

robot.robot_debug = False

while True:
    print("\n'w'=forward, 's'=backward, 'a'=left, 'd'=right, ")
    print("space=stop, 'z'=rotate left, 'c'=rotate right, 'q'=quit ")
    c = input("command: ")
    print(c)
    if c == 'q':
        break
    elif c == 'w':
        robot.forward()
    elif c == 's':
        robot.backward()
    elif c == 'a':
        robot.left()
    elif c == 'd':
        robot.right()
    elif c == ' ':
        robot.stop()
    elif c == 'z':
        robot.left_rot()
    elif c == 'c':
        robot.right_rot()
    else:
        print('unknown command')

print('stop')

```



put it all together

```
15  """
16  STARTING_ITEM = 0
17
18  ITEMS = [
19      {'name': 'spoon', 'attack': -5, 'health': -1},
20      {'name': 'sword', 'attack': 8, 'health': -100},
21      {'name': 'chair', 'attack': 5, 'health': 1},
22      {'name': 'thimble', 'attack': 1, 'health': 0},
23      {'name': 'branch', 'attack': 6, 'health': 1},
24      {'name': 'book', 'attack': 4, 'health': 2},
25      {'name': 'marker', 'attack': 2, 'health': -1},
26      {'name': 'teacher', 'attack': 4, 'health': 2},
27      {'name': 'soda', 'attack': 1, 'health': 6},
28      {'name': 'skittles', 'attack': 0, 'health': 4},
29      {'name': 'pen', 'attack': 23, 'health': -1},
30      {'name': 'table', 'attack': 30, 'health': -10},
31      {'name': 'pizza', 'attack': 2, 'health': 100},
32      {'name': 'unicorn', 'attack': 20, 'health': 30},
33      {'name': 'chicken', 'attack': 0, 'health': 10}
34  ]
```




```
Welcome to school !!! OoooohhhoooooOOOhhh! (act scared)

What is your name? Scott

You are in the Library.

There's some junk in here. It looks like a...  a teacher  and a thimble

                NORTH
                the abyss

WEST   the abyss            Cafeteria   EAST

                Stem Lab
                SOUTH

Hey Scott, what do you want to do?
```

```
Hey Scott, what do you want to do? attack with teacher

Peanutbot shall feel the wrath of your teacher.


You know, teacher is not a very effective weapon.

ATTACKING Peanutbot WITH YOUR teacher!
ATTACKING Peanutbot WITH YOUR teacher!
ATTACKING Peanutbot WITH YOUR teacher!

Peanutbot's health is now 8.

Peanutbot counter attacks NOW!

ATTACKING!
ATTACKING!
ATTACKING!

You lose 5 health points. Your health is now 95.

Attack complete.
```

classes and exceptions

```
class CaptainAmerica:
    def __init__(self):
        self.name = ‘cap’
        self.pointy_hat = True
        self.shield = True

    def attack(self):
        print('i\'m coming')

    def defend(self):
        print('stand back')

    def lift_heavy_stuff(self):
        print('uuurrggg')
```

Captain America

```
cap = CaptainAmerica()

print(cap.pointy_hat)
cap.attack()
cap.lift_heavy_stuff()
```

```
True
i'm coming
uuurrggg
```

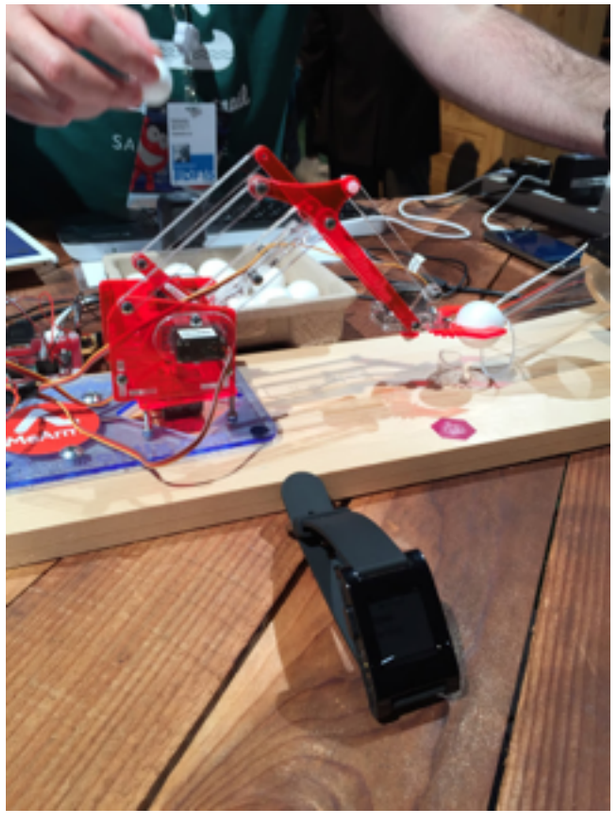
```
try:
    cap.use_senses()
except AttributeError:
    print('%s, doesn\'t …’ % cap.name)
```

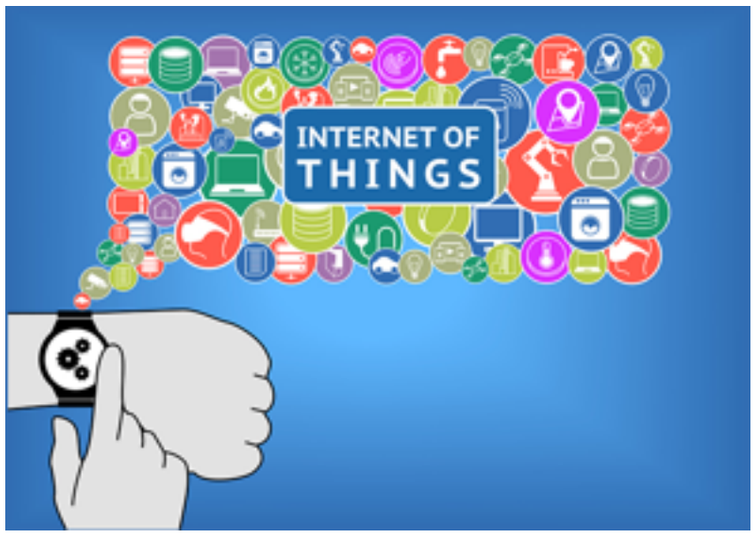
```
cap, doesn't have this power
```

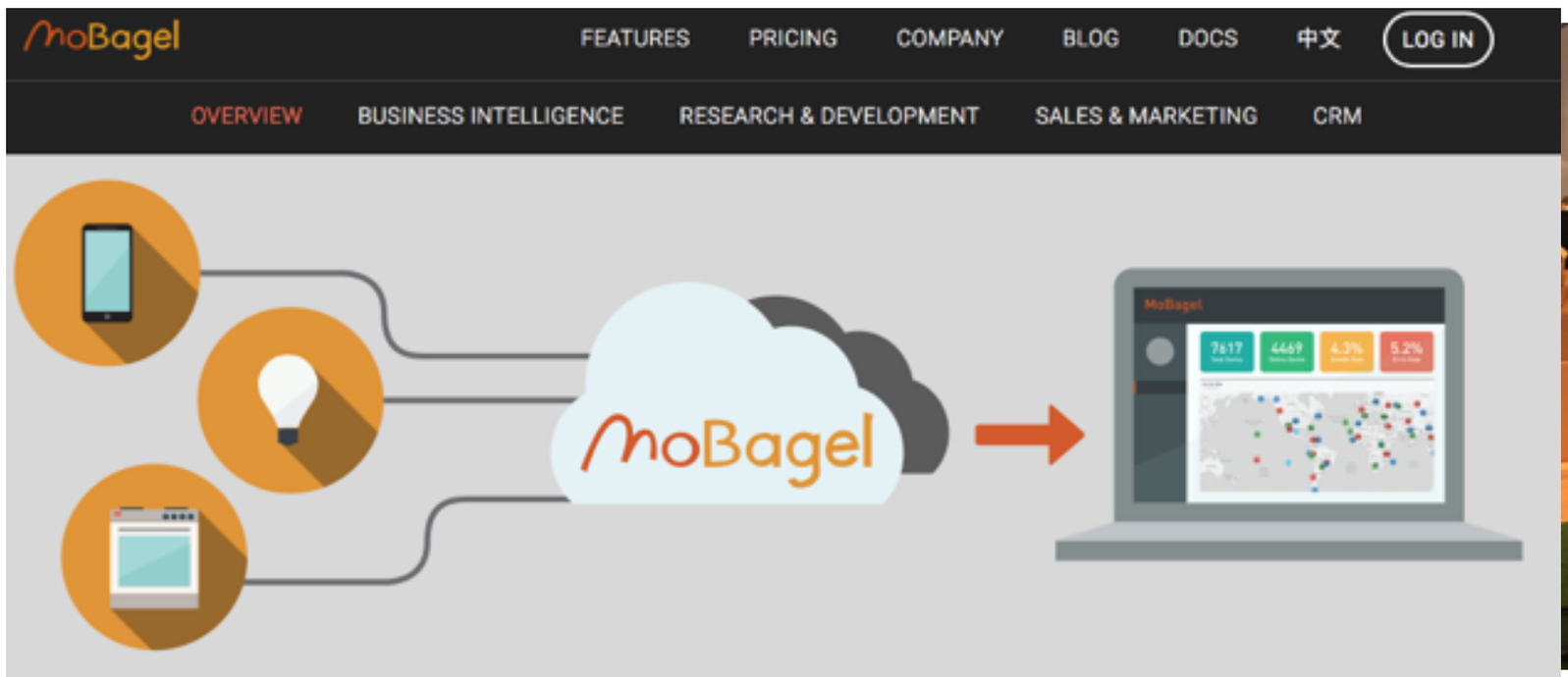
Connecting the dots

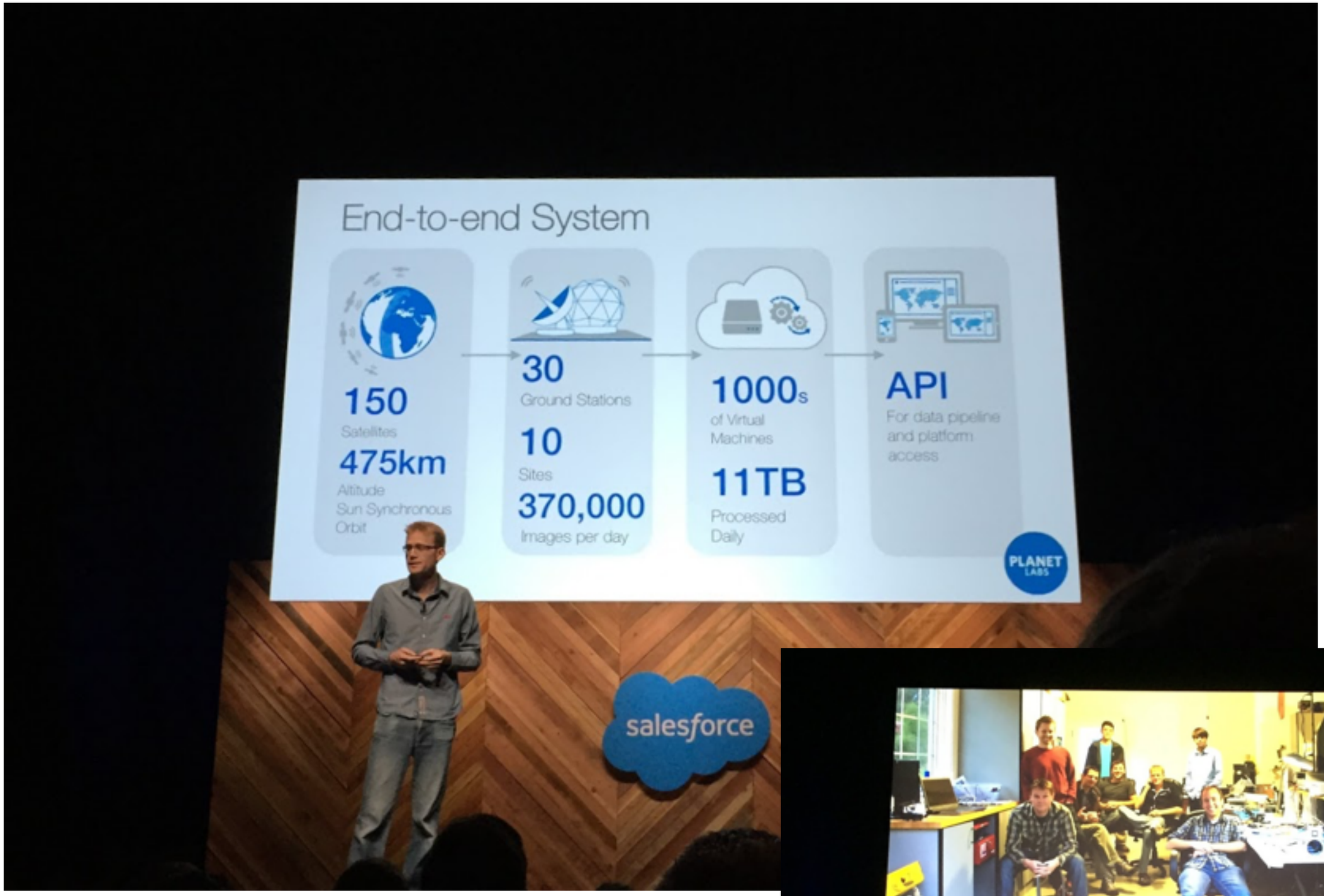
Create don’t consume

Stay up on current trends







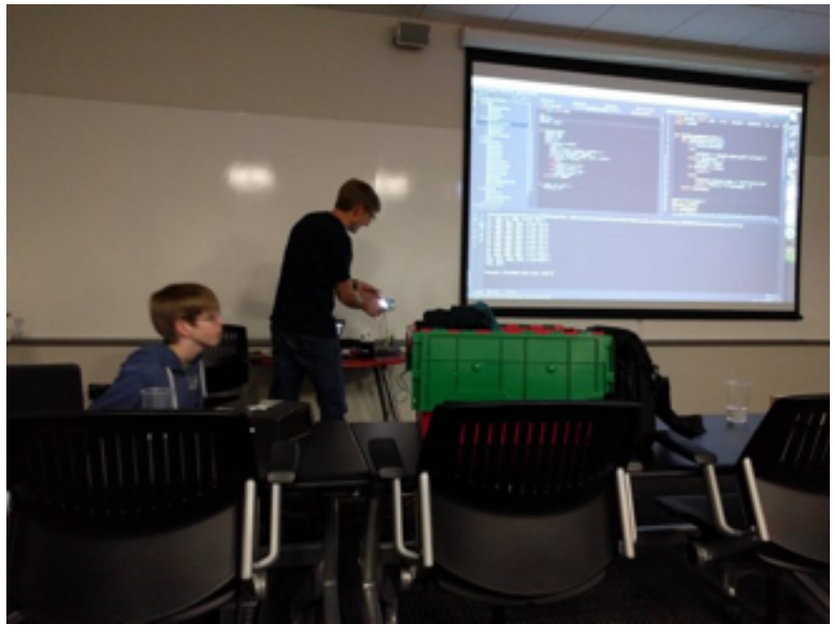

End-to-end System
150
Satellites
475km
Altitude
Sun Synchronous
Orbit
30
Ground Stations
10
Sites
370,000
Images per day
1000s
of Virtual
Machines
11TB
Processed
Daily
API
For data pipeline
and platform
access
PLANET
LABS
salesforce

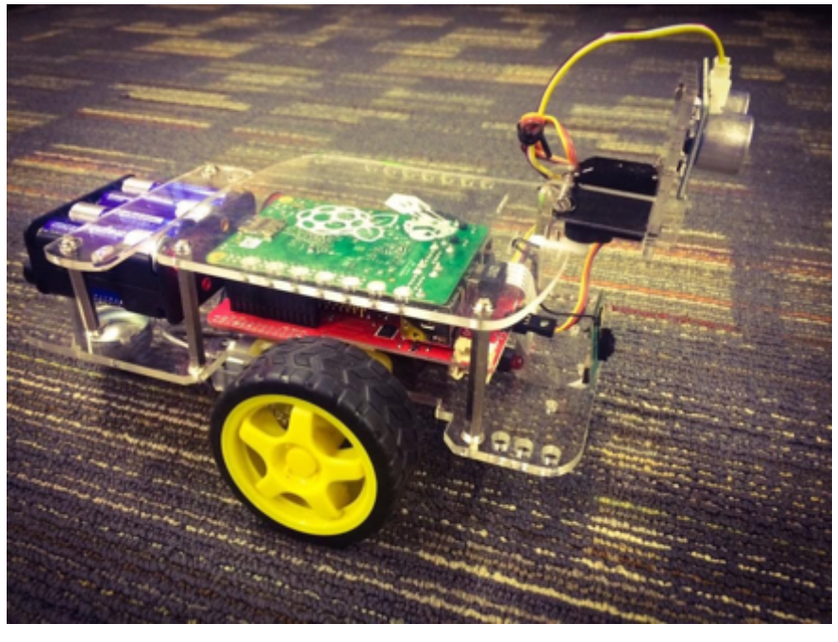

salesforce

Planet Labs

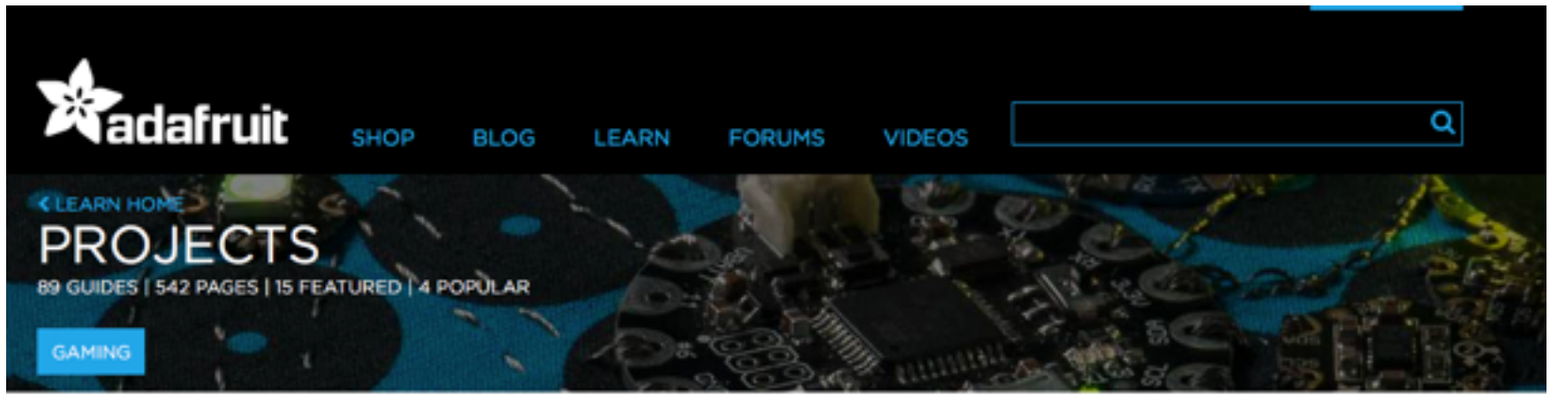
Savioke

Get involved






EXIT
WORK
MEMPHIS
I CODE



Look for ideas

Constantly build

Fair Weather Friend: Internet-Connected Migraine or Allergies Detector

"Web scraping" provides an alternative to restrictive programming APIs

by Phillip Burgess

Sandblaster - 3D printed sand buggy

3D printed sand buggy!

by Rick Winscot

Sandblaster is a variation on

Press Your Button for Raspberry Pi

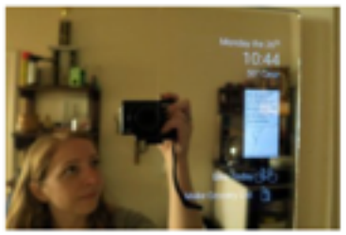
Press Your Button is a game of chance for all ages.

by Roberto Marquez

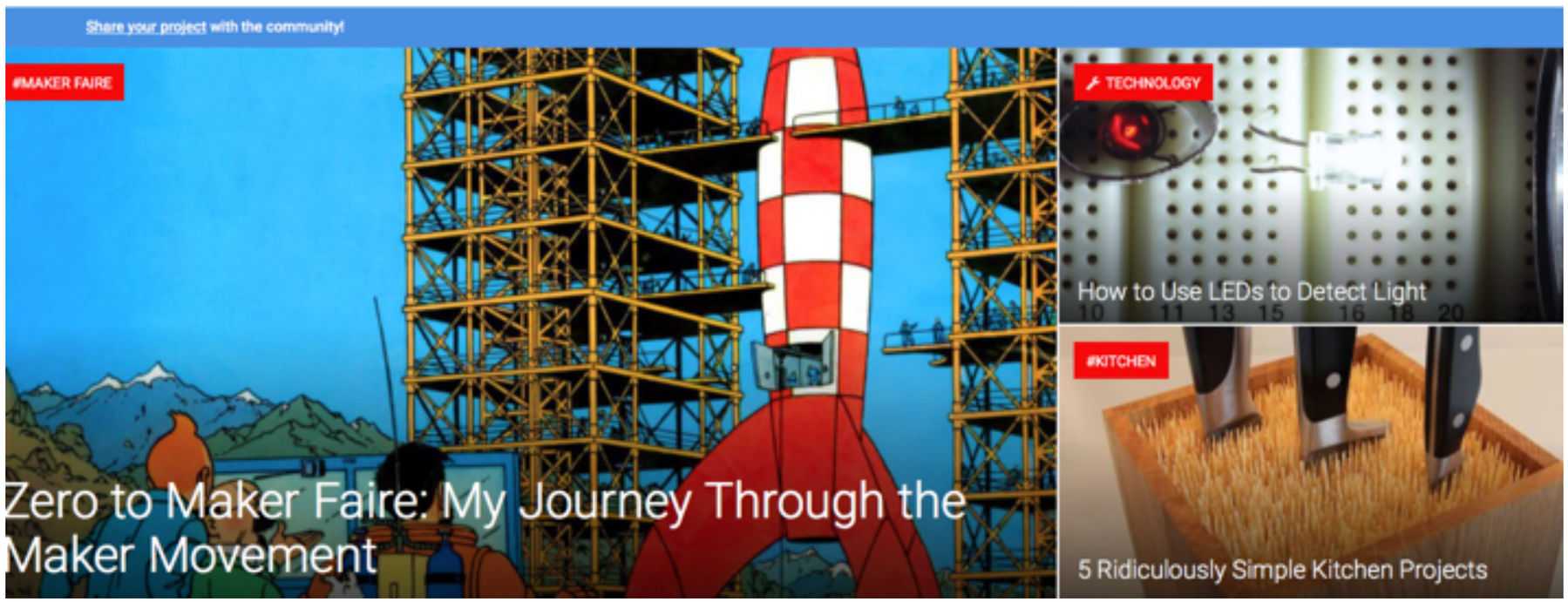
Android Smart Home Mirror

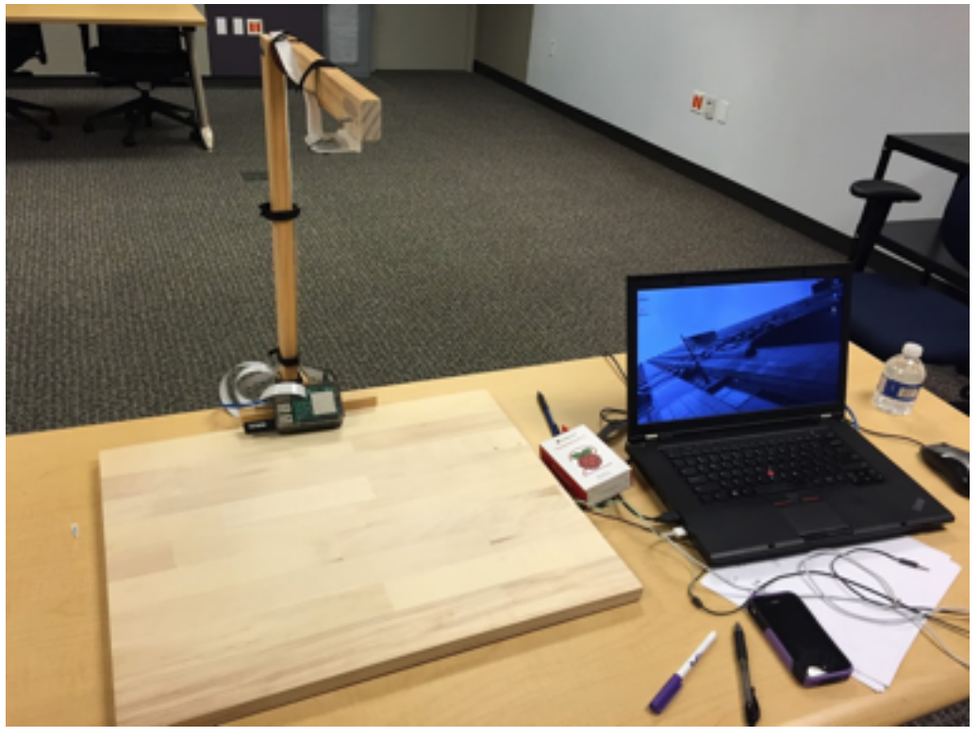
Turn an old tablet into an entryway display

by Becky Stern

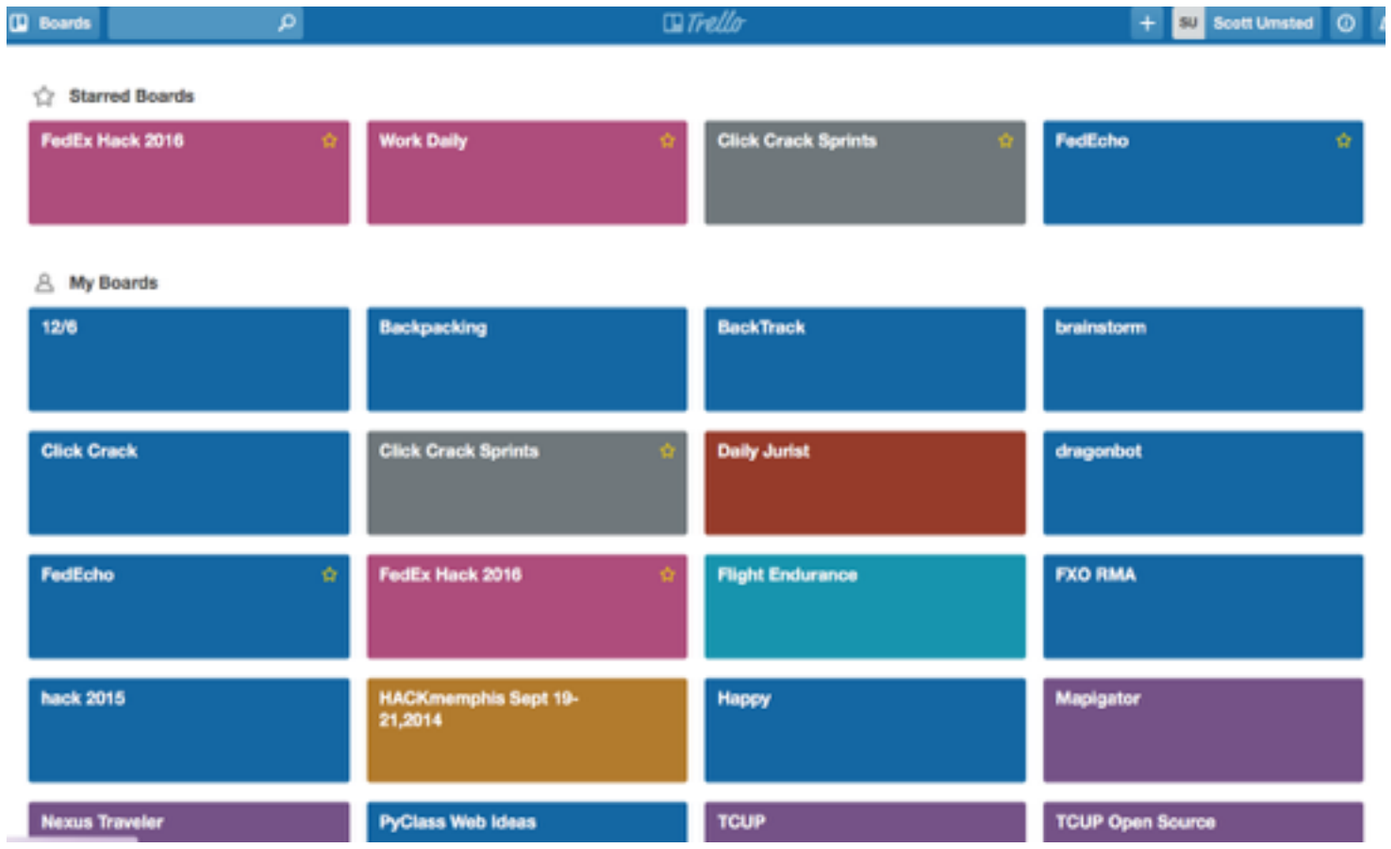


Flex your Android muscles by





Take on as many
side projects as you can

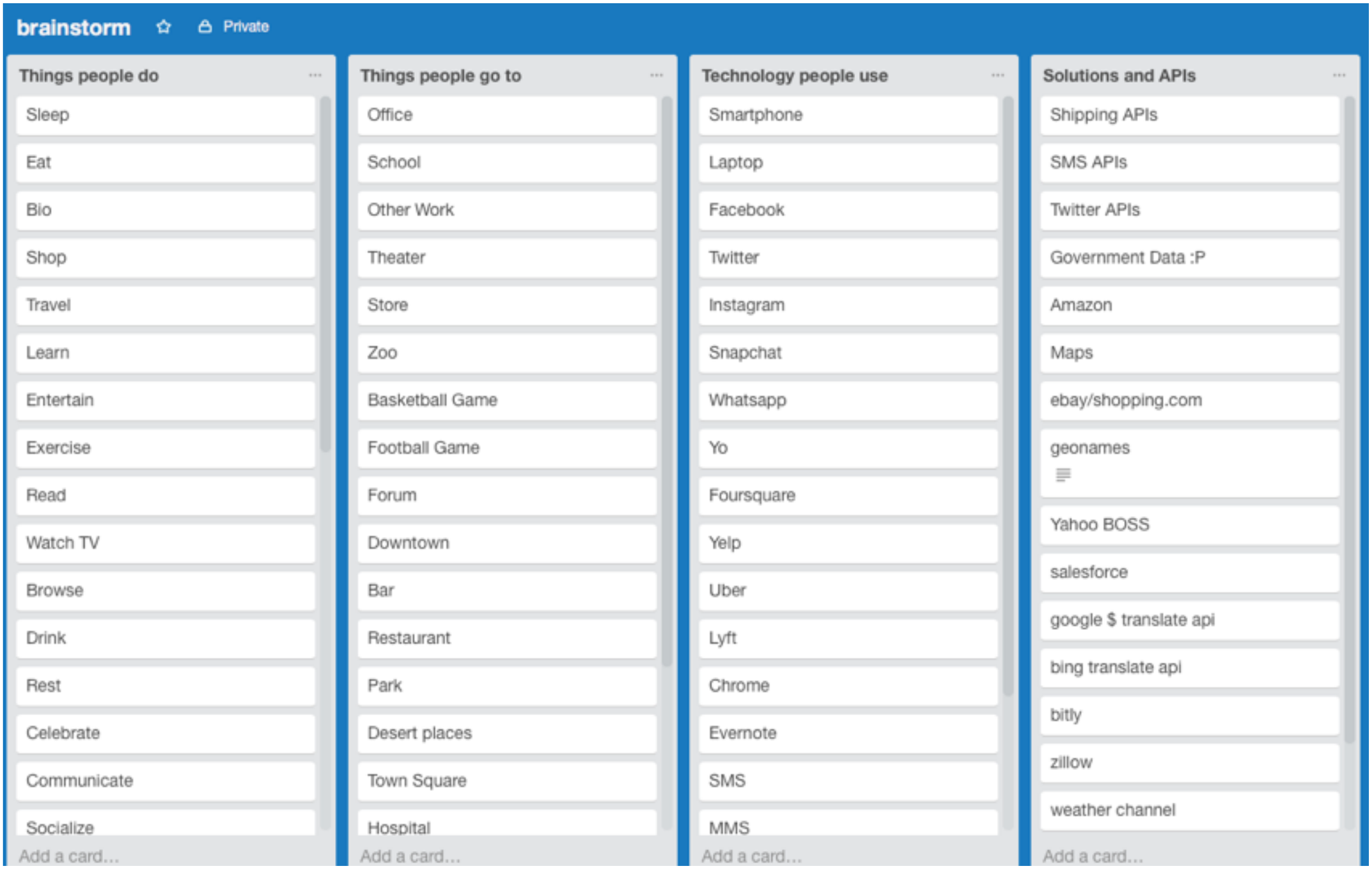


Some you will love

Others you'll drop

It's easier when it's your idea

Connect the dots



Rex

Jill

Nevel

Carly

1. Create a dictionary stored in a variable named hair_colors. The key should be name and value should be hair color.

2. Create a list of names stored in variable kids.

3. Using a loop, iterate through kids and print the name.

4. Show how you might append an new name to kids.

5. Show how you might add a new name & hair color to hair_colors.

1. Ask the user to enter a temperature in Fahrenheit, store in variable f.

2. Convert variable f to an integer, using int().

3. Create a function called get_celsius. It should accept a parameter f and return celsius. In this function subtract 32 from f and divide the result by 1.8, store in variable c. Return c.

4. Store the result of the function in variable c and print out the converted temperature, so that it looks like

   # degrees fahrenheit = # degrees celsius

5. If c is less than or equal to 0 also print

   Burr it's cold!