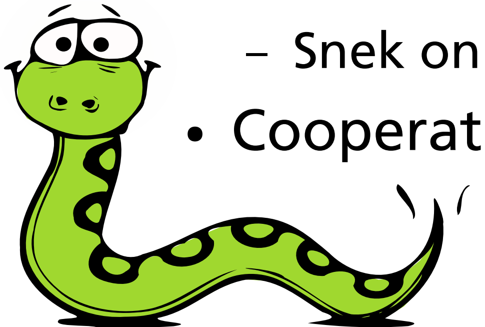
Snek: Baby Python

Keith Packard
Principal Engineer
SiFive
keithp@keithp.com
@keith_x11

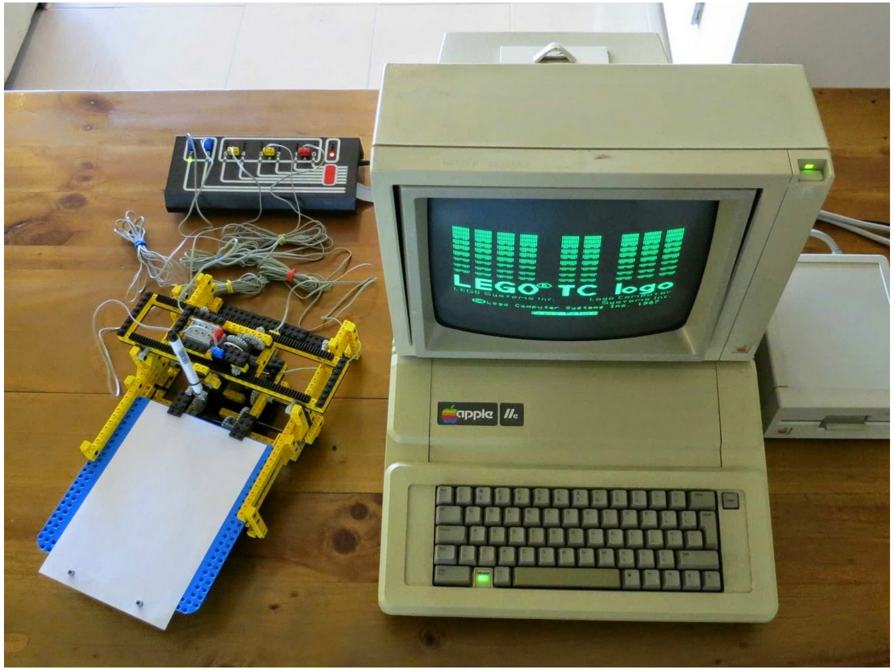
Lego Robotics Class

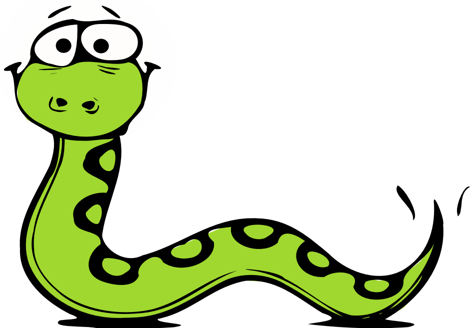
- 10 students ages 10-12
  - 3-4 years of Lego-based instruction already
- 1-2 TAs ages 13-18
  - Former students
- 2 teachers + 2 adult volunteers
- 3 languages
  - Logo on Macintosh with Lego control panel
  - Robolab on Lego RCX
  - Snek on Arduino and Snekboard
- Cooperative Environment



My First Lego Computer

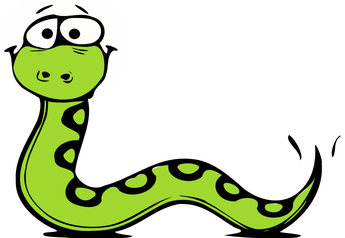
- Apple][(6502!)
- Floppy Disk
- Plug-in controller box
- Lego, so many Lego
- Logo Language



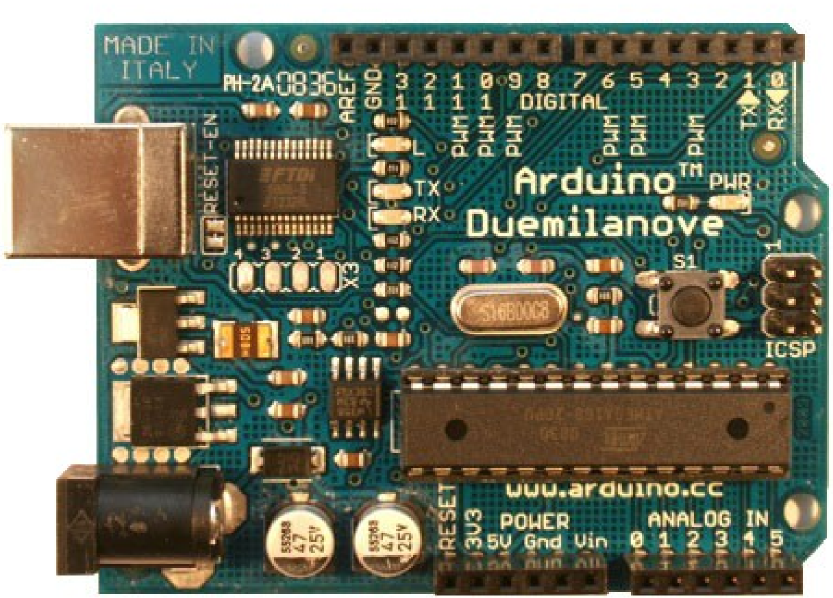


Logo Code

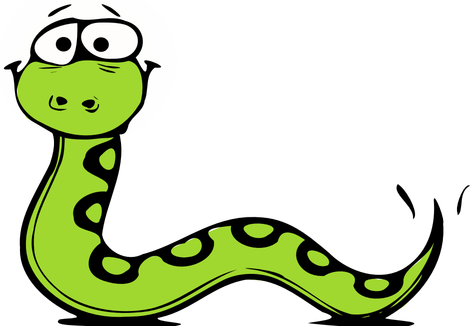
```
forever [
  talkto 1
  onfor 10
  sleep 10
]
```



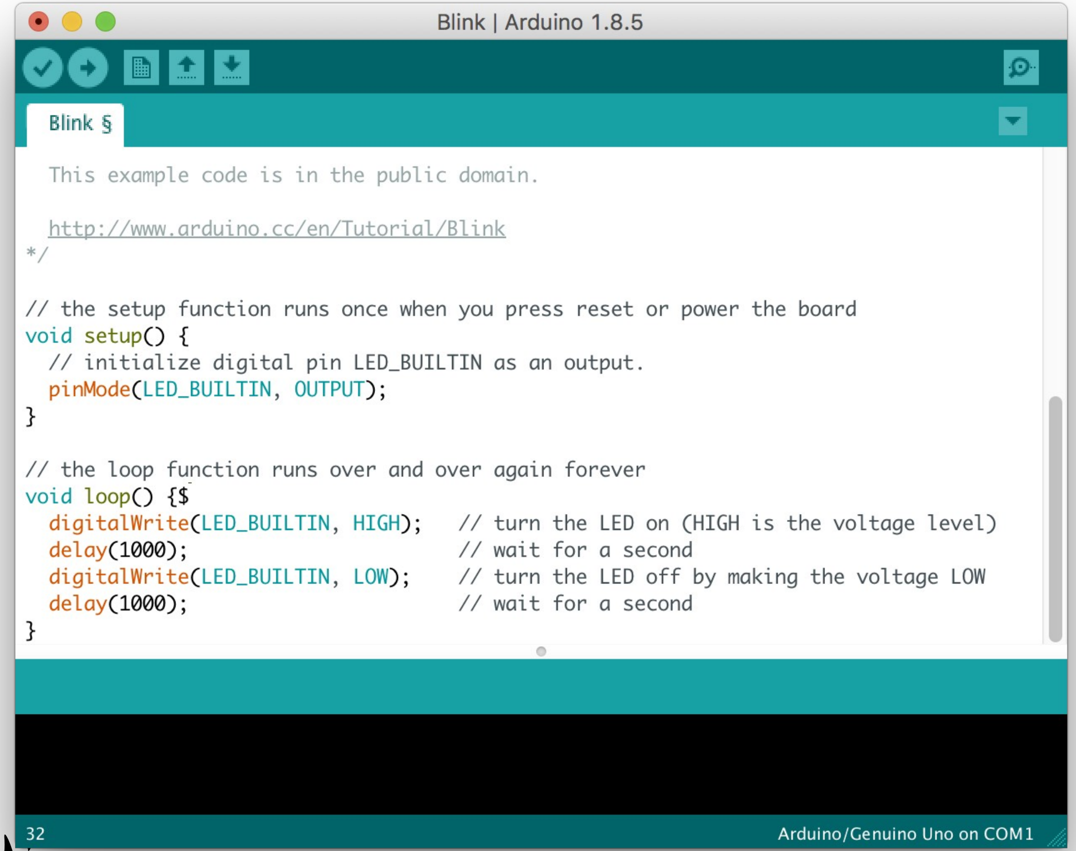
Arduino



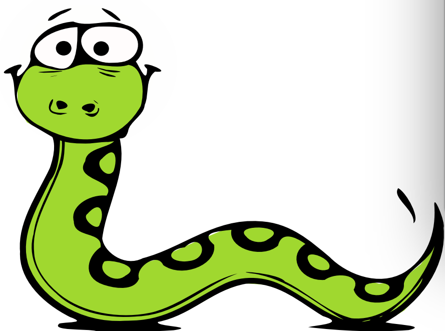
- 8-bit Atmega 328
- C++ language
- IDE on desktop



Arduino Code

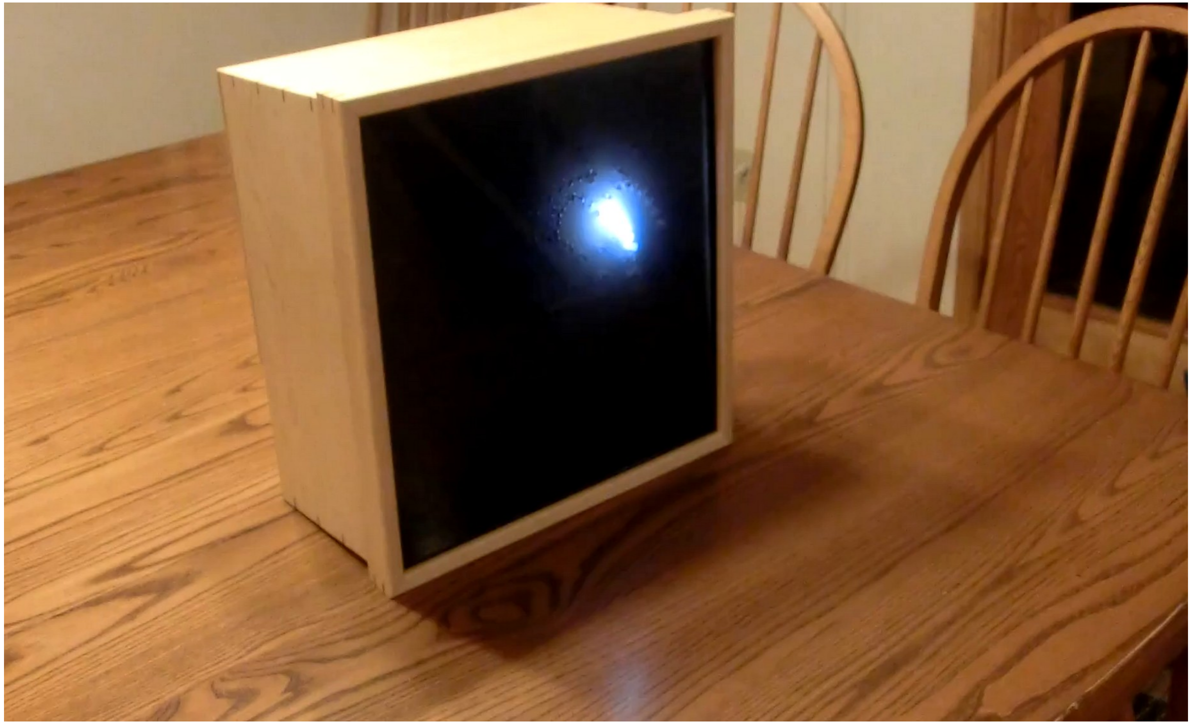


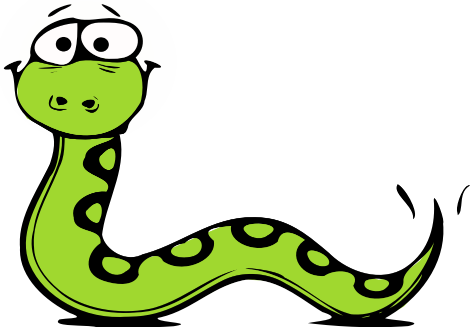
```
  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {$
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```



Fireworks

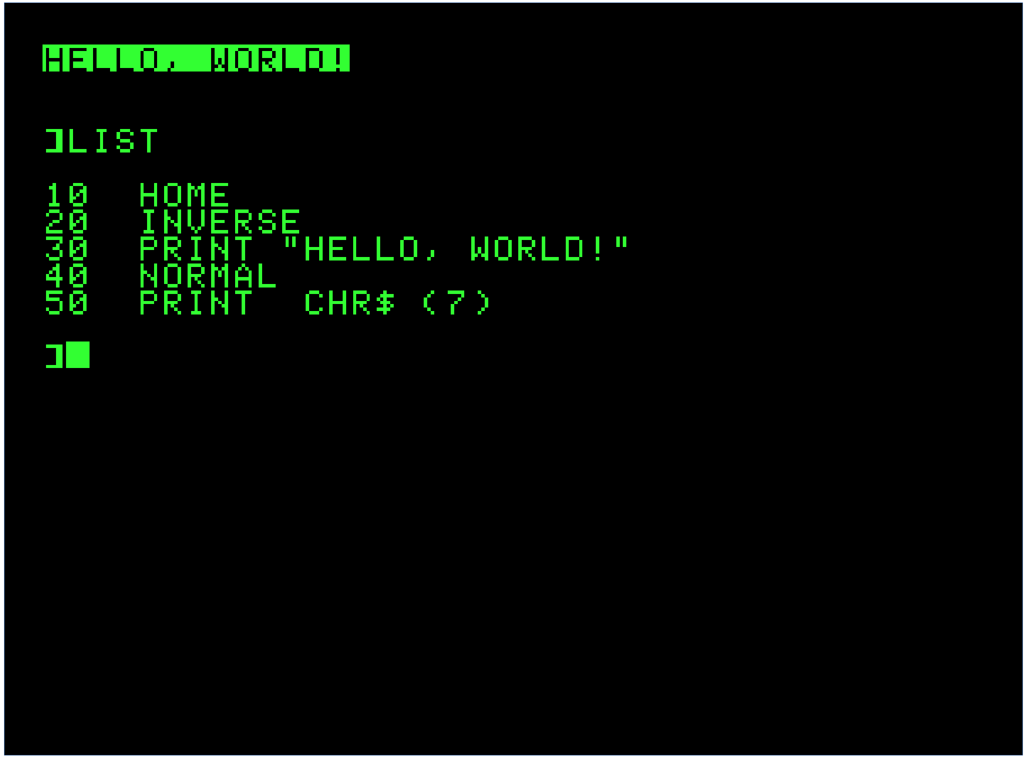
- Duemilanove
- SPI-based LED driver
- PWM hundreds of LEDs





Project Goals

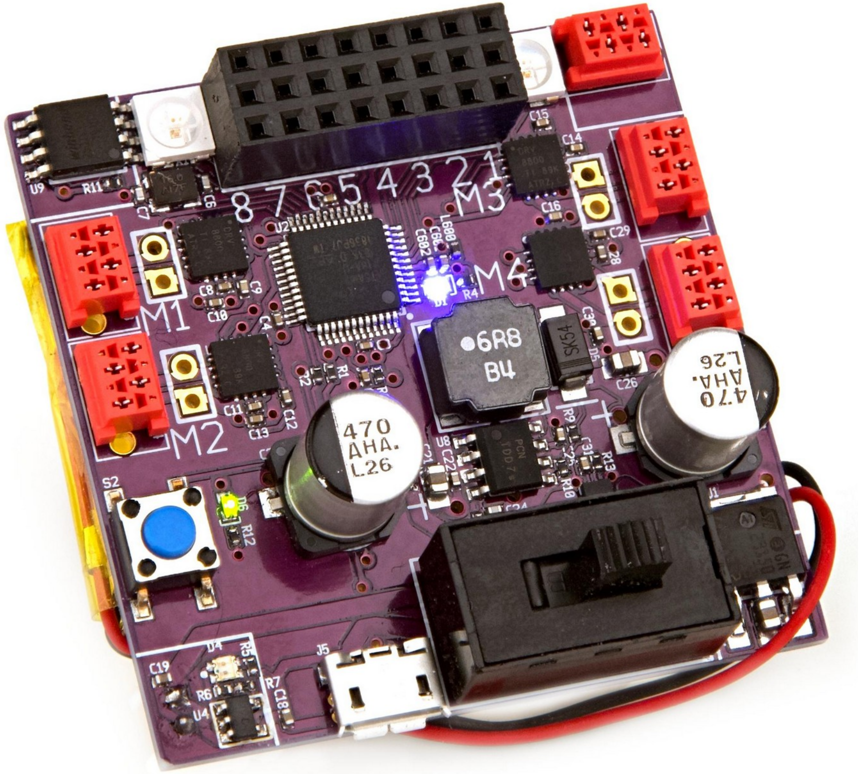
- Python-inspired syntax, BASIC-inspired scale
- Run on a Duemilanove
- Small enough to explore in a few hours

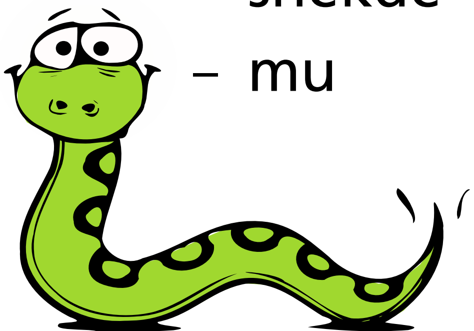


```
HELLO, WORLD!


]LIST

10  HOME
20  INVERSE
30  PRINT "HELLO, WORLD!"
40  NORMAL
50  PRINT  CHR$ (7)

]
```

Snek

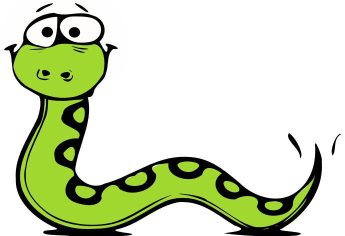
- Small SoC
  - ATmega 328P
  - RISC-V
  - ARM Cortex-M
- Python-compatible
- IDE on desktop
  - snekde
  - mu





Snek Line Bug

```
while True:
  talkto(M1)
  setright()
  on()
  while read(A1) > 0.35:
    pass
  off()
  talkto(M3)
  setleft()
  on()
  while read(A1) < 0.35:
    pass
  off()
```

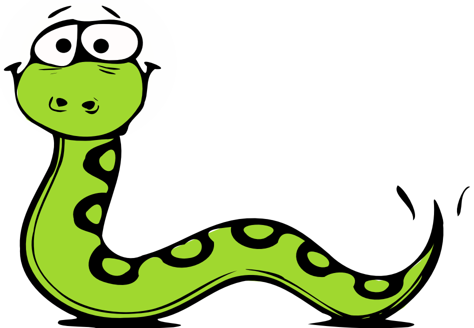


Circuit Python

- Based on Micro Python
- Larger language (objects, etc)

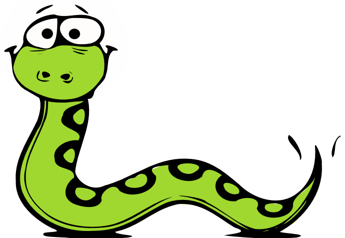
```
   text    data     bss     dec     hex filename
  64440      48   21080   85568   14e40 snek.elf
 246912     964    7124  255000   3e418 circuit-python.elf
```

- More sophisticated GPIO usage



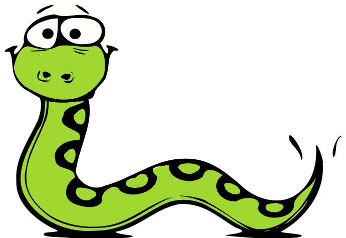
Circuit Python Example

```
import time
import board
from digitalio import DigitalInOut, Direction, Pull

led = DigitalInOut(board.D13)
led.direction = Direction.OUTPUT

switch = DigitalInOut(board.D2)
switch.direction = Direction.INPUT
switch.pull = Pull.UP

while True:
    if switch.value:
        led.value = False
    else:
        led.value = True

    time.sleep(0.01)  # debounce delay
```



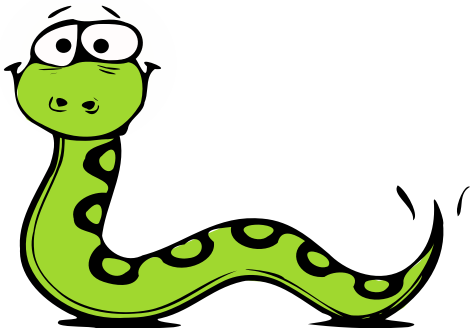
Snek Example

```
talkto(M1)
while True:
    if read(A1):
      on()
    else
      off()
```

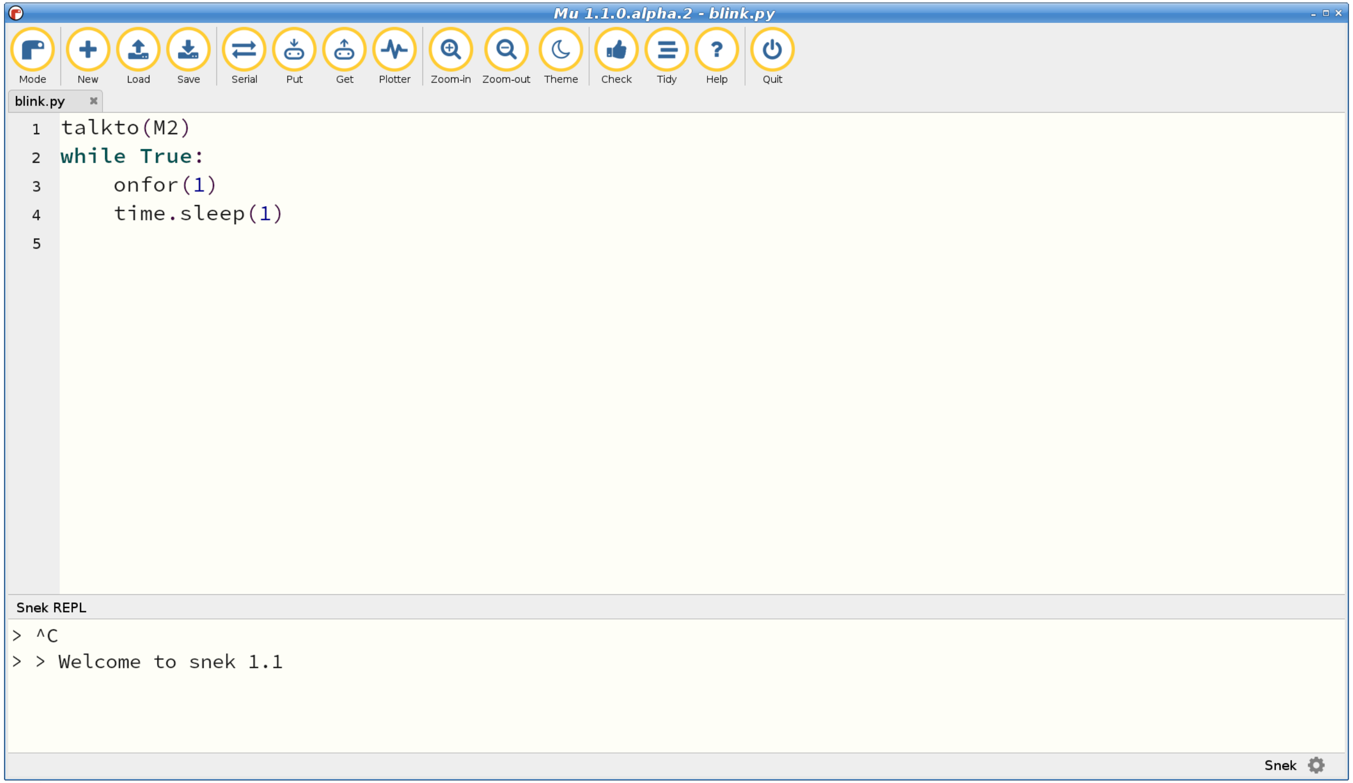


mu – a Python IDE

- An IDE for writing Python, written in Python.
- Support for MicroPython devices
- Easily extended to support Snek
- Merge Request pending upstream

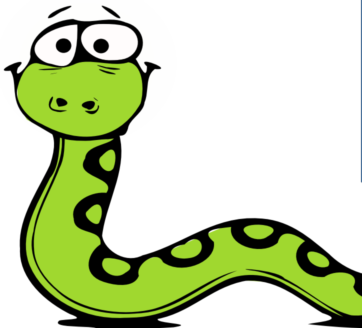


Editing Snek with Mu

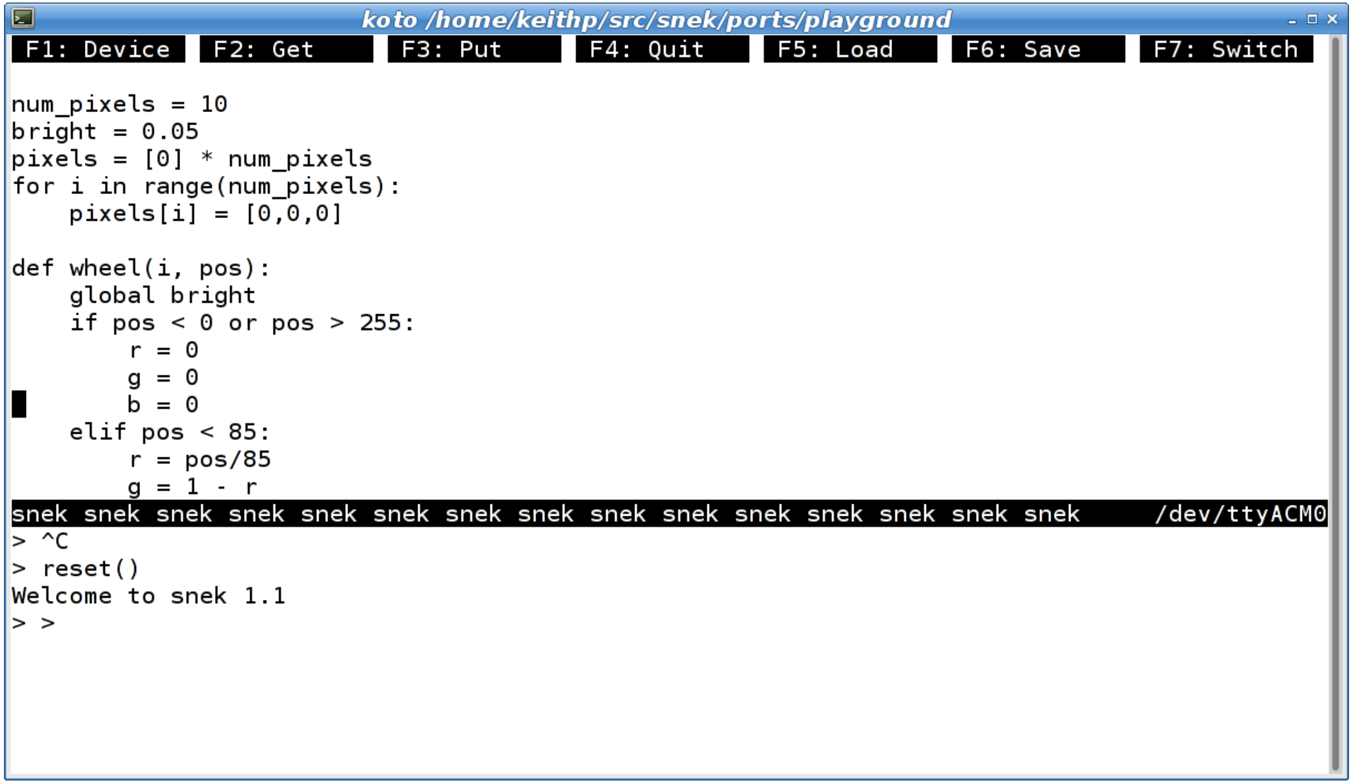


```
talkto(M2)
while True:
    onfor(1)
    time.sleep(1)

```

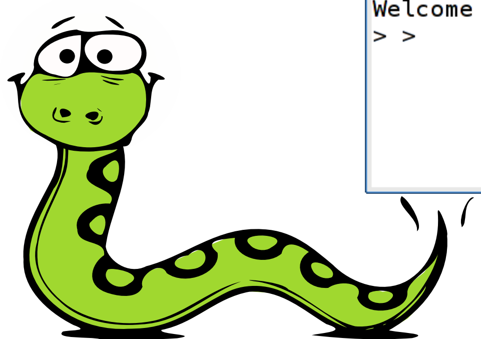
```
> ^C
> > Welcome to snek 1.1
```



Editing Snek with Snekde

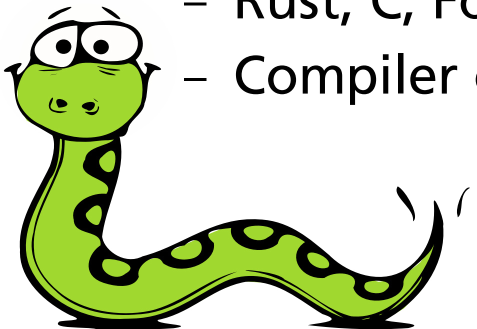


```
num_pixels = 10
bright = 0.05
pixels = [0] * num_pixels
for i in range(num_pixels):
    pixels[i] = [0,0,0]

def wheel(i, pos):
    global bright
    if pos < 0 or pos > 255:
        r = 0
        g = 0
        b = 0
    elif pos < 85:
        r = pos/85
        g = 1 - r
snek snek snek snek snek snek snek snek snek snek snek snek snek snek snek     /dev/ttyACM0
> ^C
> reset()
Welcome to snek 1.1
> >
```



Language Implementations

- Direct interpretation
  - BASIC, LISP
  - Source code **is** the executable
- Bytecode
  - Python, Snek, Perl, Ruby, Javascript, Java, …
  - Compiler output runs on a virtual machine
- Compiled
  - Rust, C, Fortran
  - Compiler output runs directly on the CPU



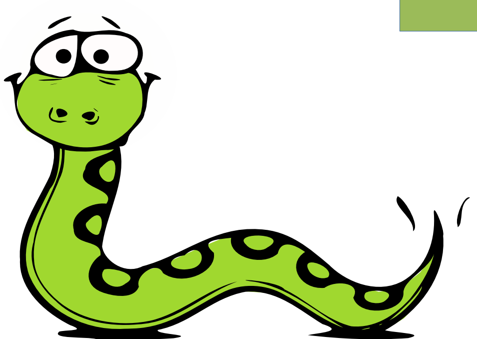
Snek Implementation

Compiler

Virtual Machine

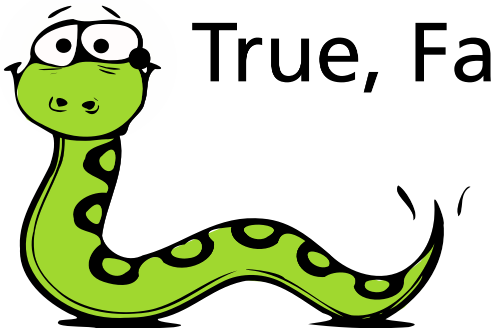
HW Support

Memory Manager

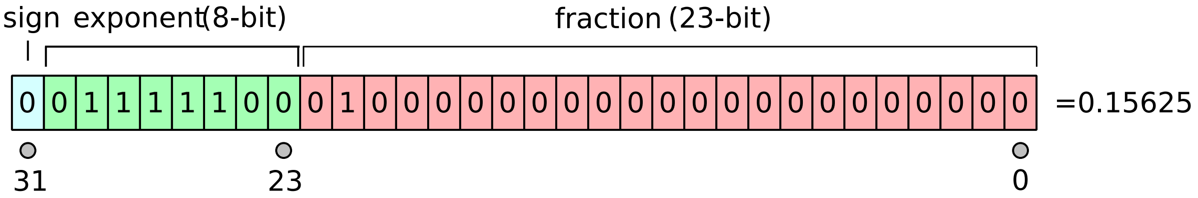


Snek Values

- 32-bit IEEE float
- Tuple/List/Dictionary
- String
- Compiled Function
- Builtin Function
- True, False, None



NaN-o-Rama

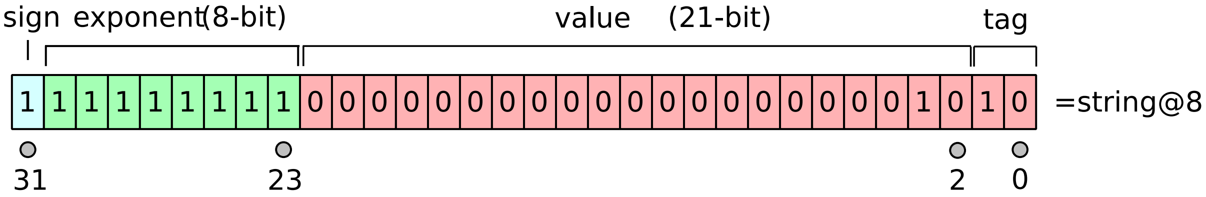


sign exponent(8-bit) fraction (23-bit)

0 | 0 1 1 1 1 1 0 0 | 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 =0.15625

31 23 0

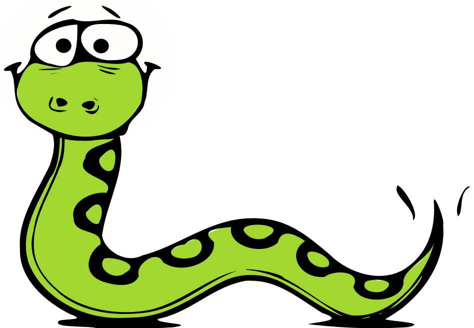
- Exponent == 0xff → NaN
  - 16 **million** NaNs!



sign exponent(8-bit) value (21-bit) tag

1 | 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 | 1 0 =string@8

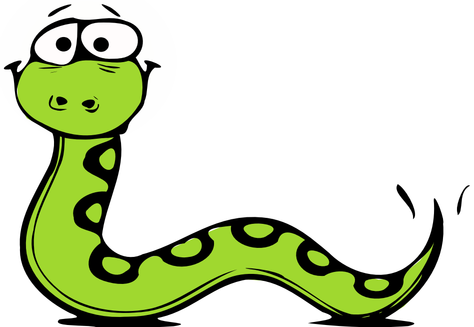
31 23 2 0



Codekaizen [CC BY 3.0 (https://creativecommons.org/licenses/by/3.0)]

Snek Compiler

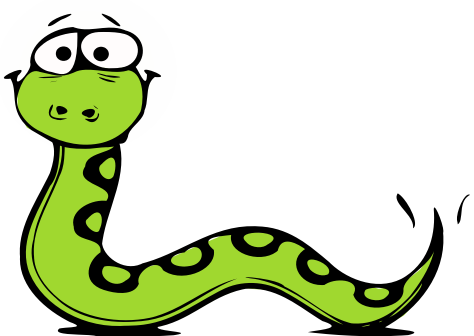
- LL(1) table-driven parser using Lola
- Hand-written lexer
- Old-school direct code generation
- About 1500 lines of code



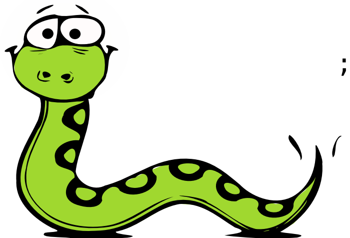
Lola

- Parser generator (written in Python)
- Small parser
  - Lola: 3600 code + 1200 data
  - Bison: 4400 code + 6000 data

```
formal : NAME
           @{
               snek_parse_formals[snek_parse_nformal++] =
                                      snek_token_val.id;
           }@
         opt-named-p
       ;
```

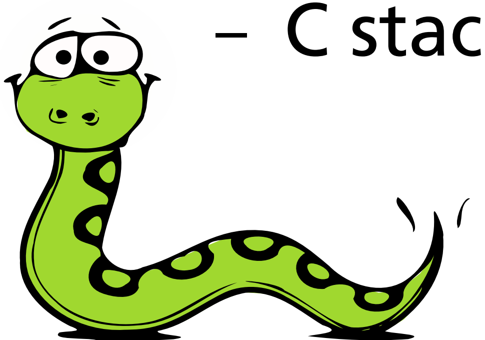


```
while-stat      :
                        @{
                                /* push 0 - top_off */
                                value_push_offset(snek_code_current());
                        }@
                  WHILE expr COLON
                        @{
                                snek_code_add_op_offset(snek_op_branch_false, 0);
                                /* push 1 - while_off */
                                value_push_offset(snek_compile_prev);
                        }@
                  suite
                        @{
                                /* push 2 - loop_end_off */
                                snek_code_add_op_offset(snek_op_branch, 0);
                                value_push_offset(snek_compile_prev);
                                /* push 3 - while_else_stat_off */
                                value_push_offset(snek_code_current());
                        }@
                  while-else-stat
                        @{
                        patch_loop:;
                                snek_offset_t while_else_stat_off = value_pop().offset;
                                snek_offset_t loop_end_off = value_pop().offset;
                                snek_offset_t while_off = value_pop().offset;
                                snek_offset_t top_off = value_pop().offset;

                                snek_code_patch_branch(while_off, while_else_stat_off);
                                snek_code_patch_branch(loop_end_off, top_off);
                                snek_code_patch_forward(while_off, loop_end_off, snek_forward_continue, top_off);
                                snek_code_patch_forward(while_off, loop_end_off, snek_forward_break, snek_code_current());
                        }@
                ;
```



Snek VM

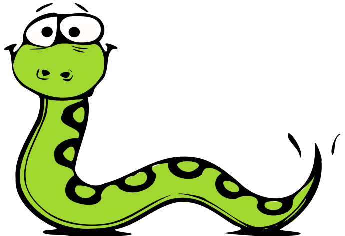
- Stack with Accumulator
- 61 opcodes
  - 39 expression operators (+, -, =, …)
  - Python-specific operators like “slice”
- Non-recursive implementation
  - C stack stays within known bounds



Snek Bytecode

```
> for i in range(10):
+  print("hello %d" % i)
+
```

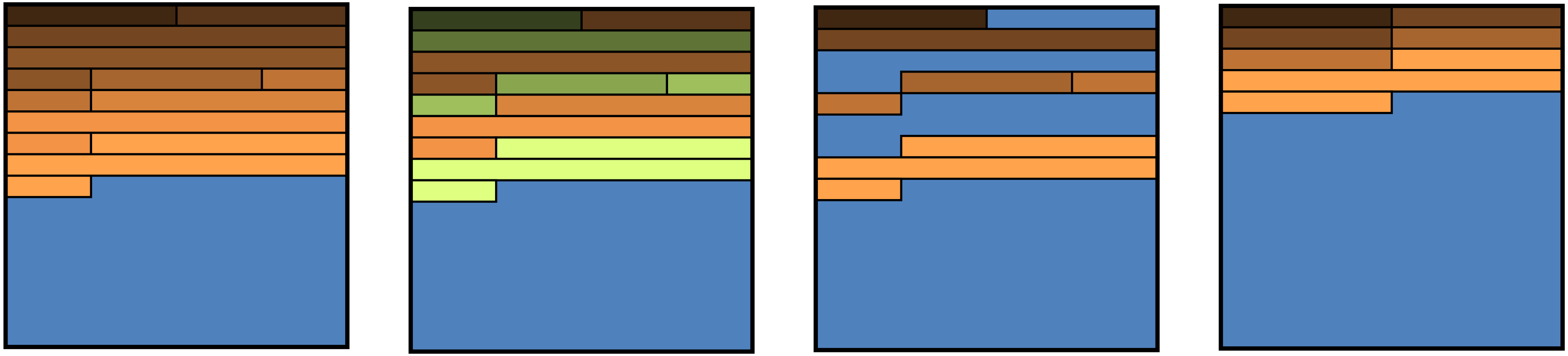
```
 0:  line             1
 3:  int            ^ 10
 5:  range_start      1 0 i
11:  range_step       36 0 i
17:  line             2
20:  id             ^ (   60) print
23:  string         ^ hello %d
26:  id               (   85) i
29:  mod            ^
30:  call             1 position 0 named
33:  branch           11
```



Snek Memory Manager

- Mark & Sweep Garbage Collector
  - uses only one heap
- Compacting
  - smooshes allocated objects all together
- Incremental
  - frees unused objects in chunks
- Bounded Recursion
  - finite stack usage during collect

Memory Management

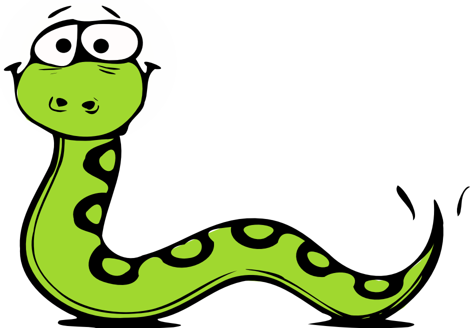


allocate

mark

sweep

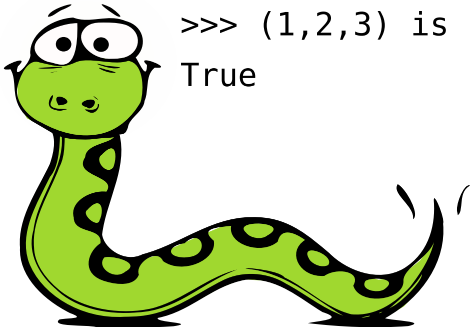
compact



Tricky Python Bits

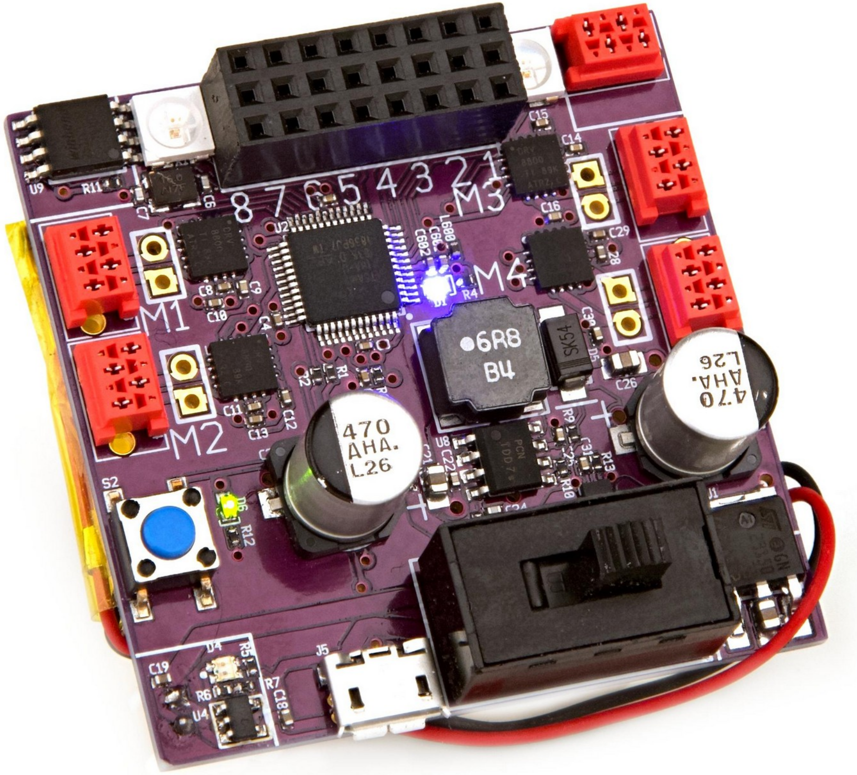
- Lexical white space
  - differs between REPL and files
  - error recovery is tricky
- Function Parameters
  - named vs positional
  - required vs optional
- Optimizing dictionaries
- Shared constants

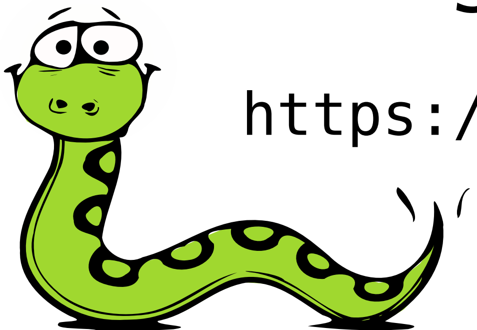
```
>>> a = (1,2,3)
>>> b = (1,2,3)
>>> a is b
False
>>> (1,2,3) is (1,2,3)
True
```



SnekBoard

- Drives Lego Power Functions motors
- Runs Snek and CircuitPython
- Crowd Supply campaign starting soon!





https://www.crowdsupply.com/keith-packard/snekboard

Demo