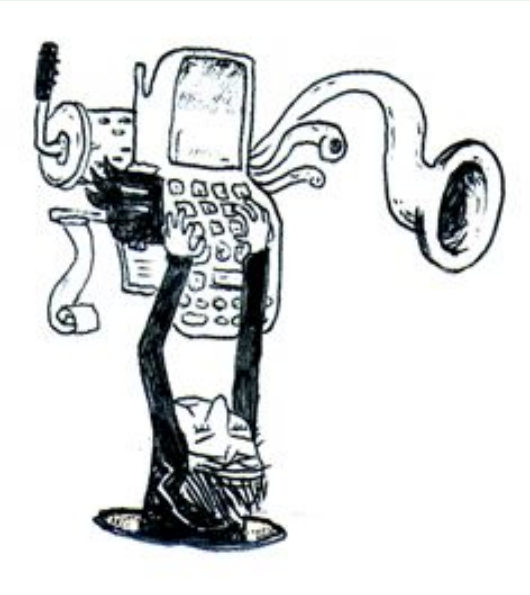
Noisy Programs



Hackety org

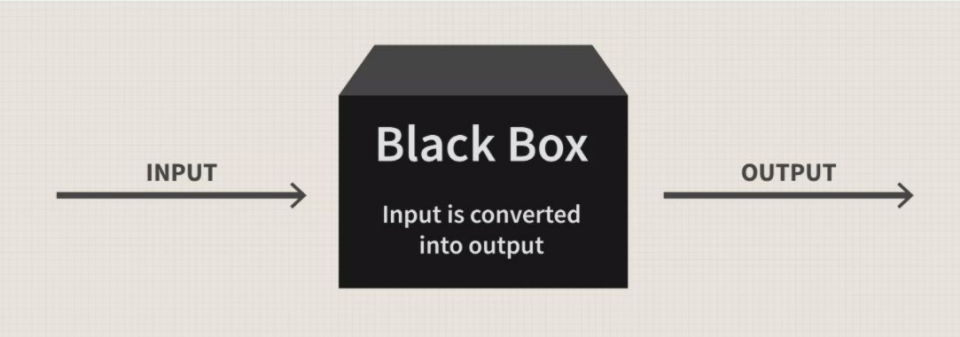
Feedback (in real life)

- In the real world, feedback is everywhere.
- Example: Cars
  - Your actions have an immediate effect, which you can feel.
  - They're noisy, giving you a sense of what goes on inside.
  - Continuous feedback is very satisfying (just ask hedgehog).

Hedgehog Driving by A-Pancake (DeviantArt)

Feedback (in computers)

- Computers are silent and motionless. (Electricity is weird)
- They're almost *too* fast—`$ echo hello` and `$ grep todo huge_file.txt` both *feel* like the same amount of work for the computer.
- Giving the user feedback is a hard problem and often faked to some degree.
- Exception: spinning hard drives

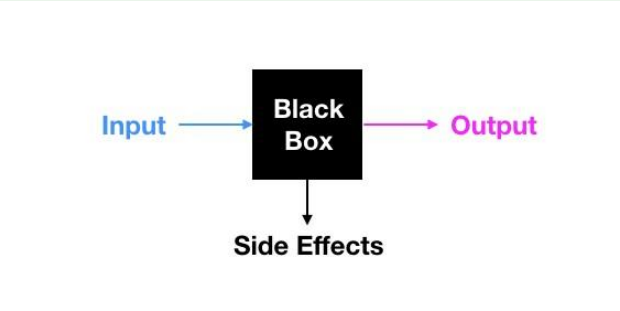


working hard…
almost there…
i promise...

i'm trying to launch this app but it keeps coming back down...

Print statements

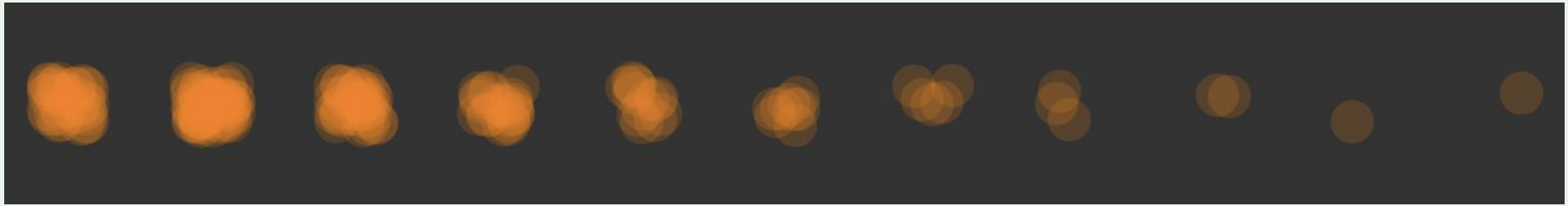
- Programs can have **side effects**.
- “Judiciously-placed print statements” can tell the *story* of how your code executed.




```
function fib(int $n) {
  print $n;
  if ($n ≤ 1) {
    return 1;
  } else {
    return fib($n - 1) + fib($n - 2);
  }
}

fib(5);
// Output: 5 4 3 2 1 0 1 2 1 0 3 2 1 0 1
```

Print statements you can FEEL

```
function fib(int $n) {
  feels("
    fill(255, 127, 0, 50);                          /* Orange, 50%-transparent */
    noStroke();
    circle(
      map($n, 0, 10, 30, width-30) + random(-10, 10), /* $n mapped to X-position */
      100 + random(-10, 10),                        /* Y-position */
      20                                            /* Size */
    );"
  );
  if ($n ≤ 1) {
    return 1;
  } else {
    return fib($n - 1) + fib($n - 2);
  }
}

fib(10);
```



Domain events, in the domain of code

```
function fib(int $n) {
  feels('fib_enter', $n);
  if ($n ≤ 1) {
    return 1;
  } else {
    return fib($n - 1) + fib($n - 2);
  }
}

fib(10);
```

```
feels_listen('fib_enter', (n) => {
  fill(255, 127, 0, 75);
  noStroke();
  circle(
    map($n, 0, 10, 30, width-30) + random(-10, 10),
    100 + random(-10, 10),
    20
  );
})
```

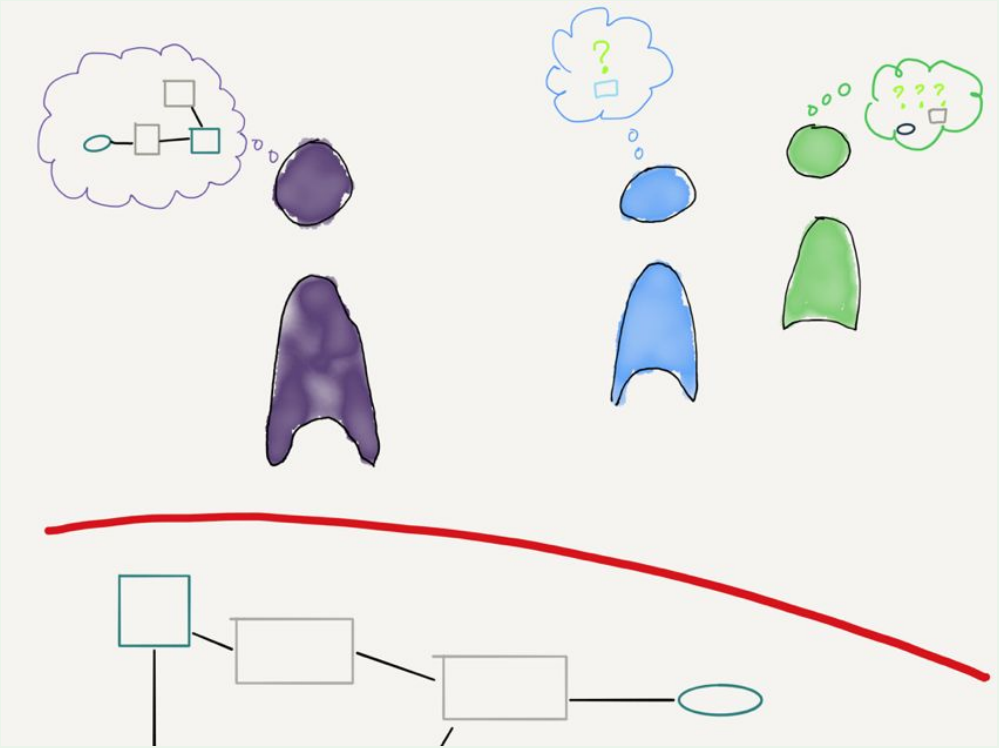
Lua + `feels`

- Lua is a programming language similar to JavaScript.
- Compiles your code into bytecode, executed by a virtual machine (i.e. a giant `switch` statement inside an infinite loop).
- What if every bytecode instruction makes a sound?
- Demo time!

```
void luaV_execute (lua_State *L, CallInfo *ci) {
  const Instruction *pc;
  EM_ASM({ lua_event('enter'); });
  /* main loop of interpreter */
  for (;;) {
    Instruction i;  /* instruction being executed */
    i = *(pc++);

    emscripten_sleep(50);

    OpCode opcode = GET_OPCODE(i);
    EM_ASM({
      lua_event('opcode ' + $0);
    }, (int32_t)opcode);

    switch (opcode) {
      case OP_MOVE:
        setobjs2s(L, ra, RB(i));
        break;
      case OP_LOADI:
        lua_Integer b = GETARG_sBx(i);
        setivalue(s2v(ra), b);
        break;
      case OP_LOADF:
        int b = GETARG_sBx(i);
        setfltvalue(s2v(ra), cast_num(b));
```

Why feedback is important

- Without feedback, our mental model of the code naturally slips away.
- Glitches in our understanding of the code is where bugs *come from*.
- We need a tight feedback loop to keep our mental models grounded in reality.
- Ideally, run your code after every small change.



@jessitron

The end

- Visualization / creative coding
  - https://explorabl.es/
  - https://p5js.org/
  - https://d3js.org/
  - https://sonic-pi.net/
  - https://natureofcode.com/
- Debugging
  - Bryan Cantrill's "Debugging Under Fire" talk
- Programming language implementation
  - http://craftinginterpreters.com/
  - The Implementation of Lua 5.0
- **`feels`**
  - https://github.com/paileyq/feels
  - https://github.com/paileyq/lua-feels

Follow @ntsutae and @Hau_kun for inspiration!