Amazon scraps secret AI recruiting tool that showed bias against women

“[Amazon’s] machine-learning specialists uncovered a big problem: their new recruiting engine did not like women. The team had been building computer programs since 2014 to review job applicants’ resumes with the aim of mechanizing the search for top talent.”

“But by 2015, the company realized its new system was not rating candidates for software developer jobs and other technical posts in a gender-neutral way. That is because Amazon’s computer models were trained to vet applicants by observing patterns in resumes submitted to the company over a 10-year period. Most came from men, a reflection of male dominance across the industry.”

Administrivia

- Assignments:
  - Lego Family [submit] due tonight (1/23)
  - Website Setup [checkoff] due tomorrow (1/24)
  - Reading Check 3 due tomorrow @ 3:30 pm (1/24)

- Editing your portfolio from home
  - Download and install Cyberduck & VS Code
  - Re-do Steps 3 & 4 from the website setup

- Make sure to take advantage of office hours and Piazza!
Functions (So Far)

- Used for **abstraction**
  - *Detail Removal*: subtasks with intuitive names
  - *Generalization*: don’t repeat code

**Lightbot:**

**Processing:**
- run setup()
- run draw()
- line(), rect(), ...
- min(), max()
Program Execution with Functions

- Functions “break” the normal sequential execution model
  - When function is called, begin execution of function code
  - When end of function is reached, jump back to where function was called from
    - The keyword associated with this is `return`

- Analogy: Song lyrics with a repeated chorus
  - Example: Survivor – Eye of the Tiger
    - Verse 1, Verse 2, Chorus, Verse 3, CHORUS, Verse 4, CHORUS, Outro
Data Passing with Functions

- It takes in zero or more inputs, completes some task(s), and then returns a value
  - Functions can do more in Processing than in Lightbot!

- **Analogy: An Oven**

- **Analogy: Song lyrics with that change slightly**
  - *Parameterized Example: Old MacDonald*
    - Chorus(cow,moo), Chorus(pig,oink), Chorus(duck,quack)
House-Drawing Function

```java
void setup() {
    size(500, 500);
}

void draw() {
    drawHouse(70, 10);
}

void drawHouse(int x, int y) {
    triangle(x, y, x-40, y+40, x+40, y+40);   // roof
    rect(x-40, y+40, 80, 80);                  // walls
    rect(x+10, y+80, 20, 40);                 // door
    return;
}
```
Return Type

What the function sends back to whoever called it

- Can be any of the datatypes: `int`, `float`, `color`, etc.
- If not returning anything, then we use `void`

```c
void drawHouse(int x, int y) {
    triangle(x, y, x-40, y+40, x+40, y+40); // roof
    rect(x-40, y+40, 80, 80);               // walls
    rect(x+10, y+80, 20, 40);              // door
    return;
}
```
Function Name

- Function name does not matter to computer, but does to humans.
  - Should describe what the function does.

- Subject to same naming constraints as variables.

- No two functions (or variables) can have the same name.
  - Confusing for the computer.
Parameters

- Required part of every function definition
  - Must be surrounded by parentheses
  - If no parameters, parentheses are left empty
- Datatype and name for every parameter must be specified
  - You are essentially declaring variables
  - Separate parameters with commas

```c
void drawHouse(int x, int y) {
    triangle(x, y, x-40, y+40, x+40, y+40); // roof
    rect(x-40, y+40, 80, 80);                // walls
    rect(x+10, y+80, 20, 40);                // door
    return;
}
```
Function Body

- Body is enclosed in curly braces `{ }`
  - Parameters are variables that are used inside the body

- Body of a function is **indented** for better readability
  - Processing uses two spaces by default
  - Can use Edit → “Auto Format” to clean yours up automatically
Lightbot Functions

- Lightbot functions had a different syntax, but similar parts:

```python
F.turn_around() { Right, Right }
```
Functions Worksheet

```cpp
void setup() {
    size(500, 500);
}

void draw() {
    drawHouse(70, 10);
}

void drawHouse(int x, int y) {
    triangle(x, y, x-40, y+40, x+40, y+40); // roof
    rect(x-40, y+40, 80, 80); // walls
    rect(x+10, y+80, 20, 40); // door
    return;
}
```

- Make sure you *explain* why you see what you see!
Donatello as a Function

```cpp
void drawDon() {
    fill(0, 100, 0);  // dark green
    rect(x_pos, 182, 40, 15);  // top of head

    fill(88, 44, 141);  // purple
    rect(x_pos, 197, 40, 6);  // bandana mask

    fill(0, 100, 0);  // dark green
    rect(x_pos, 203, 40, 20);  // bottom of head

    fill(219, 136, 0);  // dark yellow
    rect(x_pos, 223, 40, 50);  // shell

    fill(0, 100, 0);  // dark green
    rect(x_pos, 273, 40, 45);  // lower body
}
```
Donatello Function *Parameterized*

- Can now call `drawDon()` function with different arguments (stored in parameter `x_don`):

```cpp
// draw Donatello
void drawDon(int x_don) {
    fill(0, 100, 0);  // dark green
    rect(x_don, 182, 40, 15);  // top of head
    ...
}
```

```cpp
void draw() {
    background(255, 245, 220);
    drawDon(200);
    drawDon(400);
}
```

- We can also add parameter `color mask` to draw the other Teenage Mutant Ninja Turtles!
Parameters vs. Arguments

void draw() {
    background(255, 245, 220);
    drawTurtle(200, color(88, 44, 141)); // donatello
    drawTurtle(400, color(255, 0, 0));   // raphael
}

void drawTurtle(int x, color mask) {
    fill(0, 100, 0);       // dark green
    rect(x, 182, 40, 15);  // top of head
    fill(mask);            // apply mask color
    rect(x, 197, 40, 6);   // bandana mask
    ...

- Implicit parameter-variable initialization with argument values
Parameters vs. Arguments

- When you define a function, you specify the parameters
  - Parameters are *internal variables* / boxes for functions
  - Use parameters for values that you want to be different on different calls to this function

- When you call a function, you pass arguments
  - The order of the arguments must match the order of the parameters

- We define a function once, but can call it as many times as we want (and in different ways)!
Solving Problems

- Understand the problem
  - What is the problem description?
  - What is specified and what is unspecified?
  - What has been given to you (e.g. starter code)?

- Break the task down into less complex subtasks

- Example: Make a function that draws a row of five mice with their ears touching/overlapping. The mice should all be the same color except for the middle one, which should be red.

  main subtask: draw a mouse
  something like: void mouse(int x-pos, color c)
Parameter Example

```c
// draw mouse at position (x,y) in color c
void mouse() {
    noStroke();
    fill(color(255,0,255));  // magenta color
    ellipse(50, 50, 50, 50);  // head
    ellipse(25, 30, 30, 30);  // right ear (left on screen)
    ellipse(75, 30, 30, 30);  // left ear (right on screen)
    fill(0);                  // black color
    ellipse(40, 44, 10, 10);  // right eye (left on screen)
    ellipse(60, 44, 10, 10);  // left eye (right on screen)
    stroke(0);                // black color
    line(20, 50, 48, 60);     // upper-right whisker
    line(80, 50, 52, 60);     // upper-left whisker
    line(25, 70, 48, 60);     // lower-right whisker
    line(75, 70, 52, 60);     // lower-left whisker
}
```
Parameter Example

```c
void draw() {
    mouse(0, 0, color(255, 0, 0));
    mouse(100, 0, color(0, 255, 0));
    mouse(200, 0, color(0, 0, 255));
}

// draw mouse at position (x,y) in color c
void mouse(int x, int y, color c) {
    noStroke();
    fill(c); // argument color
    ellipse(50+x, 50+y, 50, 50); // head
    ellipse(25+x, 30+y, 30, 30); // right ear (left on screen)
    ellipse(75+x, 30+y, 30, 30); // left ear (right on screen)

    fill(0); // always black
    ellipse(40+x, 44+y, 10, 10); // right eye (left on screen)
    ellipse(60+x, 44+y, 10, 10); // left eye (right on screen)

    stroke(0); // always black
    line(20+x, 50+y, 48+x, 60+y); // upper-right whisker
    line(80+x, 50+y, 52+x, 60+y); // upper-left whisker
    line(25+x, 70+y, 48+x, 60+y); // lower-right whisker
    line(75+x, 70+y, 52+x, 60+y); // lower-left whisker
}
```
Looking Forward

- **Portfolio**
  - Don’t forget to add Taijitu, Logo Design, and Lego Family!

- **Animal Functions**
  - Start in lab on Thursday, due Monday (1/28)
  - Design your own animal (like the frog shown here)

Example from CSE120 Wi18 student