

Variables & Datatypes

CSE 120 Spring 2017

Instructor:

Justin Hsia

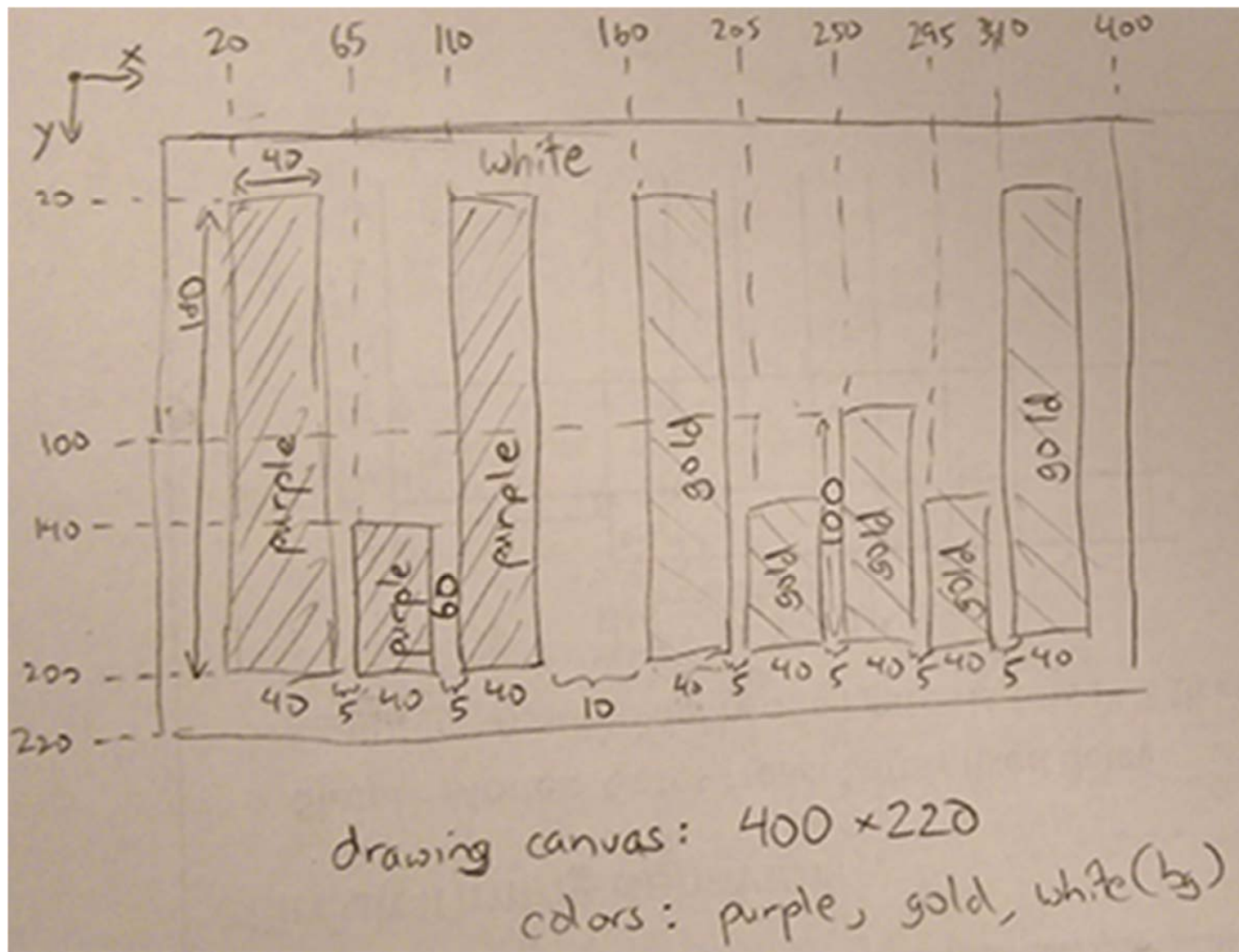
Teaching Assistants:

Anupam Gupta, Braydon Hall, Eugene Oh, Savanna Yee

Administrivia

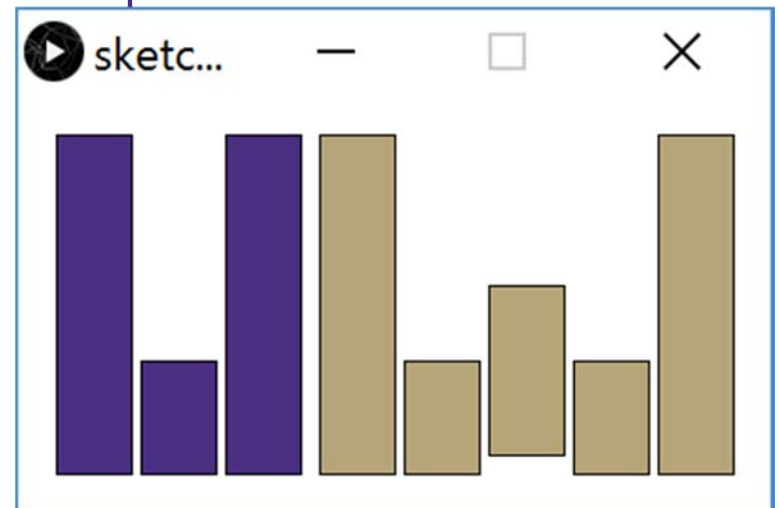
- ❖ Assignments:
 - Taijitu due today (4/5)
 - Reading Check 2 due tomorrow (4/6)
 - Custom Logo due Friday (4/7)
- ❖ No “big ideas” lecture this week
 - More time on programming

Lab: Custom Logo



Lab: Custom Logo

```
uw_logo ▾
1 /* uw_logo.pde
2   Created by Justin Hsia
3
4   UW logo made out of rectangles in school colors.
5 */
6
7 size(400,220); // drawing canvas of 400x220
8 background(255); // white background
9
10 // The letter 'U' in purple
11 fill(75, 47, 131); // purple fill
12 rect(20, 20, 40, 180); // left side of U
13 rect(65, 140, 40, 60); // middle base of U
14 rect(110, 20, 40, 180); // right side of U
15
16 // The letter 'W' in gold
17 fill(183, 165, 122); // gold fill
18 rect(160, 20, 40, 180); // left segment of W
19 rect(205, 140, 40, 60); // left base of W
20 rect(250, 100, 40, 90); // middle segment of W
21 rect(295, 140, 40, 60); // right base of W
22 rect(340, 20, 40, 180); // right segment of W
```



Drawing a Square

❖ [See Demo on Panopto]

Variables

- ❖ Piece of your program that holds the value of something
 - Every variable must be given a *name* and a *datatype*
- ❖ The values of these **variables** can change (*i.e.* vary) during the execution of your program
 - Warning: Not like a variable in Algebra (*i.e.* an unknown)
- ❖ **Assignment**: give a variable a specific value
 - *e.g.* $x = 12;$
- ❖ **Read**: use the current value of a variable
 - *e.g.* $y = x + 1;$

Datatypes

- ❖ `int`: integers
- ❖ `float`: decimal/real numbers
- ❖ `color`: a triple of numbers representing RGB
- ❖ `boolean`: true or false

- ❖ Many more exist and can be found in the Processing Reference:

Primitive

`boolean`

`byte`

`char`

`color`

`double`

`float`

`int`

`long`

Declarations

- ❖ We **declare** a variable by telling Processing the variable's datatype, followed by the variable's name:

```
1 int x;  
2 float half;  
3 color yellow;
```

- ❖ You can also give a variable a starting value (**initialization**) in the same line as the declaration:

```
1 int x = 4;  
2 float half = 0.5;  
3 color yellow = color(255, 255, 0);
```


Drawing a Square with Variables

❖ [See Demo on Panopto]

Variable Rules & Guidelines

- ❖ Variables are case-sensitive
 - e.g. `leftside` is not the same as `leftSide`
- ❖ Variable names are meaningless to computers, but meaningful to humans
 - Choosing informative names improves readability and reduces confusion
- ❖ In this class, most of our variables will be declared and initialized at the very top of our programs

Variable Manipulation

- ❖ Executed sequentially, just like other statements
- ❖ For variable assignments, compute right-hand side *first*, then store result in variable

❖ Example: `int x = 4;`
 `x = x + 1;`

- 1) Read the current value of `x` (4) for the right-hand side
- 2) Add 1 to the current value of `x`
- 3) Store the result (5) back into `x`

Variable Practice

```
1) int x = 1;
   int y = 2;
   int z = 3;
```

```
x = x + 1;
y = y - 1;
z = z + 2;
```

x	y	z

```
2) int x = 7;
   int y = 2;
   int z = 0;
```

```
x = x + 3;
y = y - 2;
z = x + y;
```

x	y	z

```
3) int x = -1;
   int y = 0;
   int z = 5;
```

```
x = x + z;
y = y - x;
z = x + z;
```

x	y	z

TMNT: Donatello

```
donatello
1 size(500,500);
2 noStroke();
3 background(255,245,220);
4
5 // Donatello
6 fill(0,100,0); // dark green
7 rect(230,182,40,15); // top of head
8
9 fill(88,44,141); // purple
10 rect(230,197,40,6); // bandana mask
11
12 fill(0,100,0); // dark green
13 rect(230,203,40,20); // bottom of head
14
15 fill(219,136,0); // dark yellow
16 rect(230,223,40,50); // shell
17
18 fill(0,100,0); // dark green
19 rect(230,273,40,45); // lower body
```

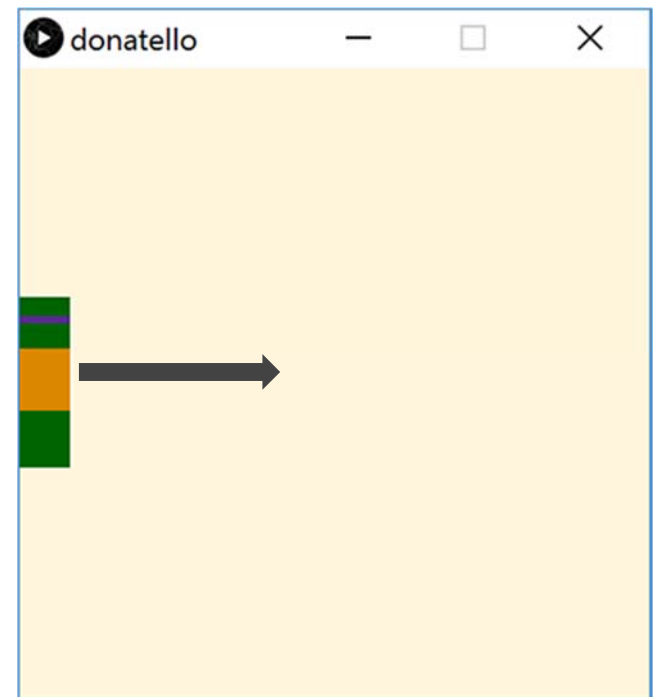
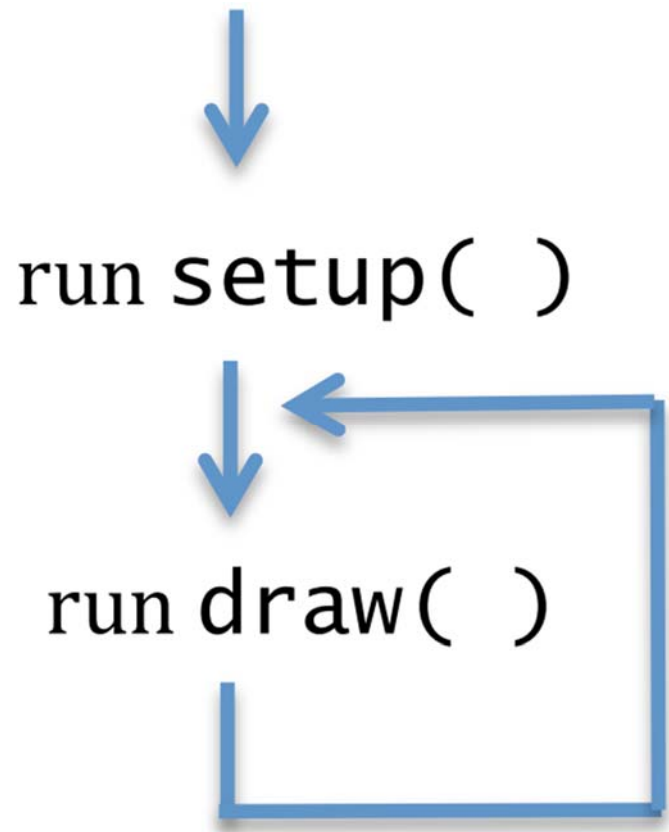


Donatello with a Variable

```
donatello
1 int x_pos = 100;           // x-position
2
3 size(500,500);
4 noStroke();
5 background(255,245,220);
6
7 // Donatello
8 fill(0,100,0);           // dark green
9 rect(x_pos,182,40,15);   // top of head
10
11 fill(88,44,141);        // purple
12 rect(x_pos,197,40,6);   // bandana mask
13
14 fill(0,100,0);           // dark green
15 rect(x_pos,203,40,20);  // bottom of head
16
17 fill(219,136,0);        // dark yellow
18 rect(x_pos,223,40,50);  // shell
19
20 fill(0,100,0);           // dark green
21 rect(x_pos,273,40,45);  // lower body
```



Donatello with Motion



Stopping Motion

- ❖ Stop Donatello from running off the right side of the screen:

```
x_pos = min(x_pos + 1, 460);
```

- ❖ Stop Donatello from running off the left side of the screen:

```
x_pos = max(x_pos - 1, 0);
```


Falling Into Place

- ❖ Introduce variables for each body segment:

```
3 int head_pos = 0; // head position
4 float mask_pos = 15; // mask position
5 int face_pos = 21; // face position
6 float body_pos = 41; // body position
7 int leg_pos = 91; // leg position
```

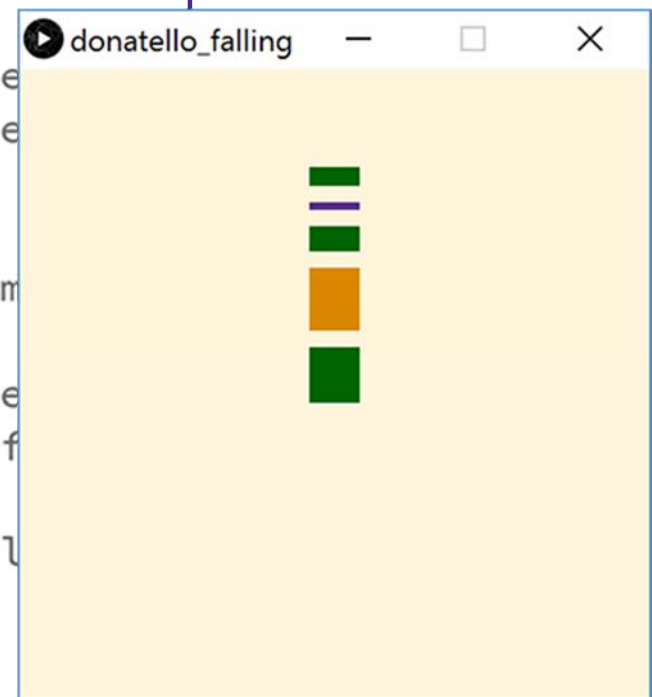
- ❖ Update each variable at different speeds:

```
33 head_pos = min(head_pos + 3, 364);
34 mask_pos = min(mask_pos + 3.5, 379);
35 face_pos = min(face_pos + 4, 385);
36 body_pos = min(body_pos + 4.5, 405);
37 leg_pos = min(leg_pos + 5, 455);
```

Falling Into Place

- ❖ Update y-positions of drawing based on new variables:

```
17 // Donatello
18 fill(0,100,0); // dark green
19 rect(x_pos,head_pos,40,15); // top of head
20
21 fill(88,44,141); // purple
22 rect(x_pos,mask_pos,40,6); // bandana mask
23
24 fill(0,100,0); // dark green
25 rect(x_pos,face_pos,40,20); // bottom of face
26
27 fill(219,136,0); // dark yellow
28 rect(x_pos,body_pos,40,50); // shell
29
30 fill(0,100,0); // dark green
31 rect(x_pos,leg_pos,40,45); // lower body
```



Summary

- ❖ Variables are named quantities that can vary during the execution of a program
- ❖ Datatypes specific different forms of data
 - e.g. int, float, color, Boolean
- ❖ Variable *declarations* specify a variable datatype and name to the program
 - Generally occurs at top of program
- ❖ `min()` and `max()` functions can be used to limit or stop change in a variable value