

Computer Creativity

# *Conditionals*



Okanagan

Slides courtesy of Dr. Abdallah Mohamed.

# *Announcements*

---

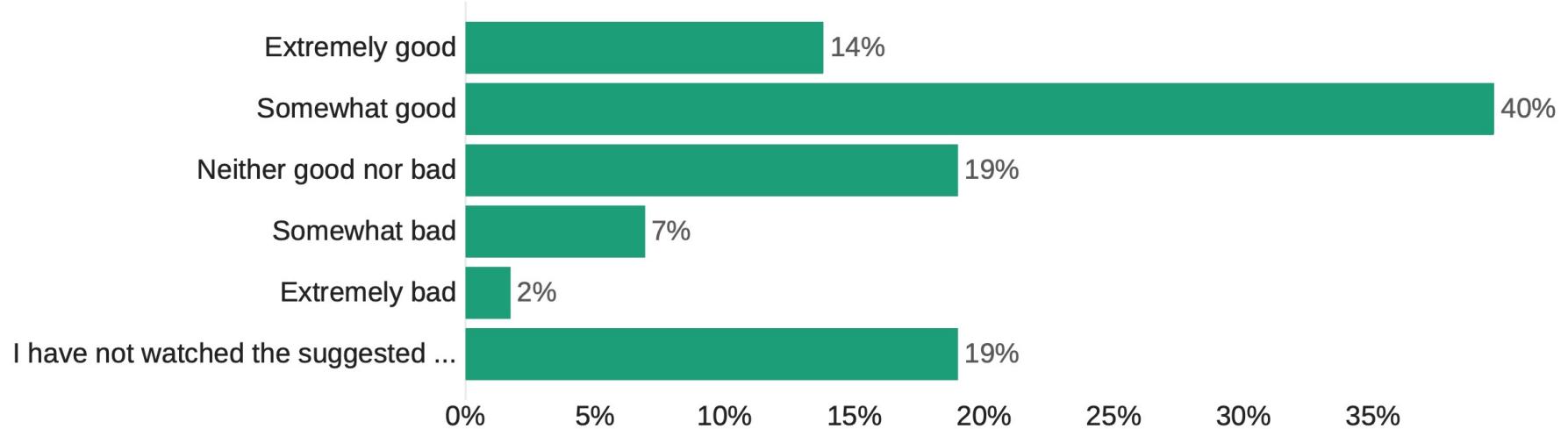
- No Tests this week!
- Classes, Labs, Student hours all resume this week.
- Reminder to keep up on labs and activities!
  - According to the suggested schedule, you should be done Week 6 Labs and Activities
  - This week Week 8 Lab and Week 8 Activity have been released
- Results from the mid-course survey are next

# *Results of the Survey*

---

What do you think of the Course YouTube videos so far?

58 Responses

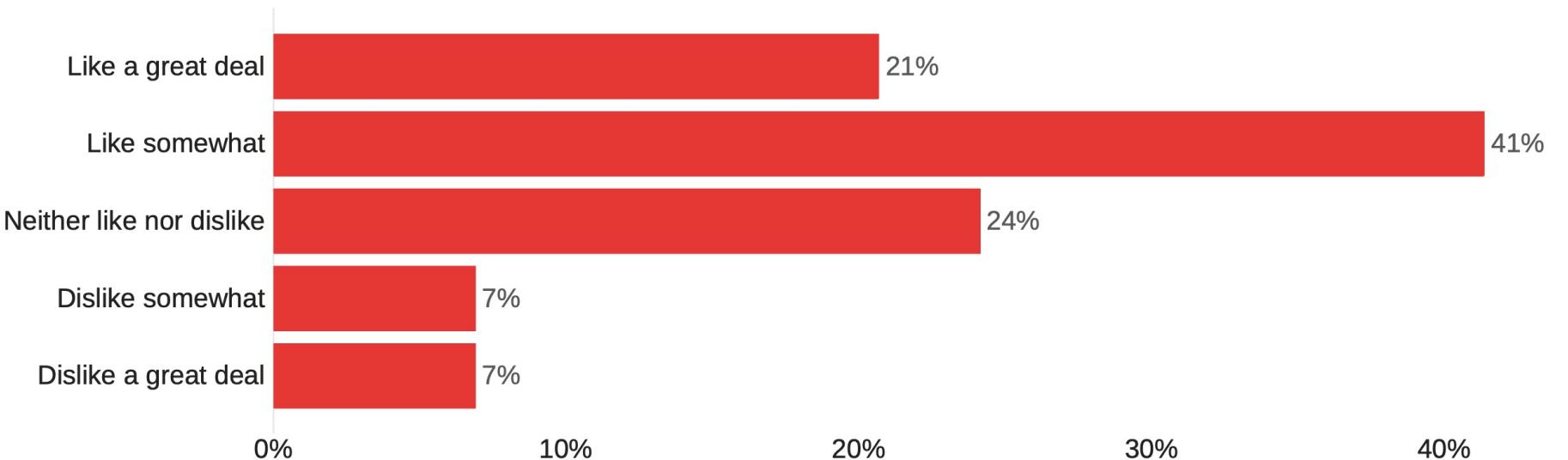


# *Results of the Survey*

---

Q3 - What do you think about the labs so far?

58 Responses

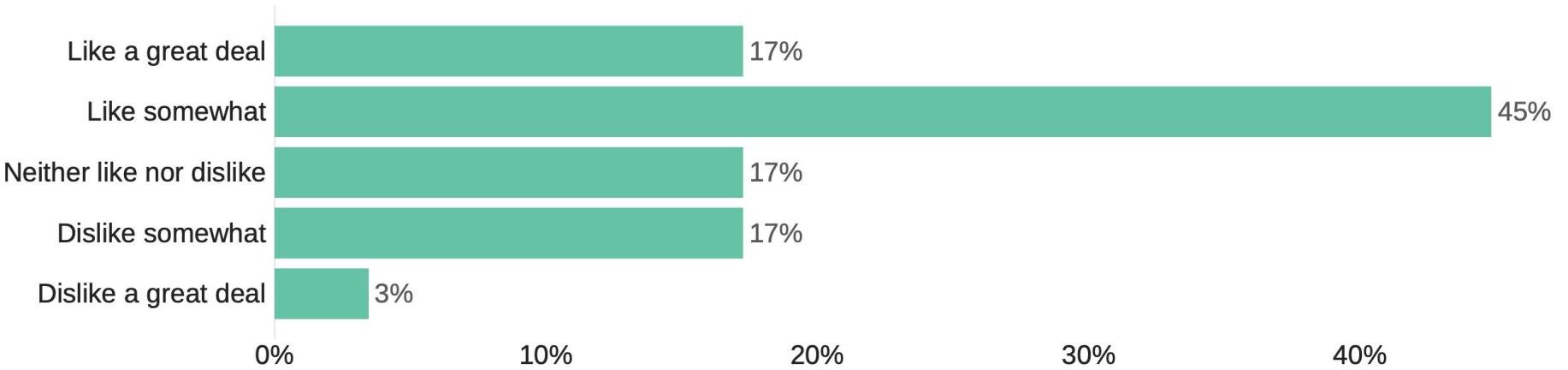


# *Results of the Survey*

---

What do you think of the Lecture Activities so far?

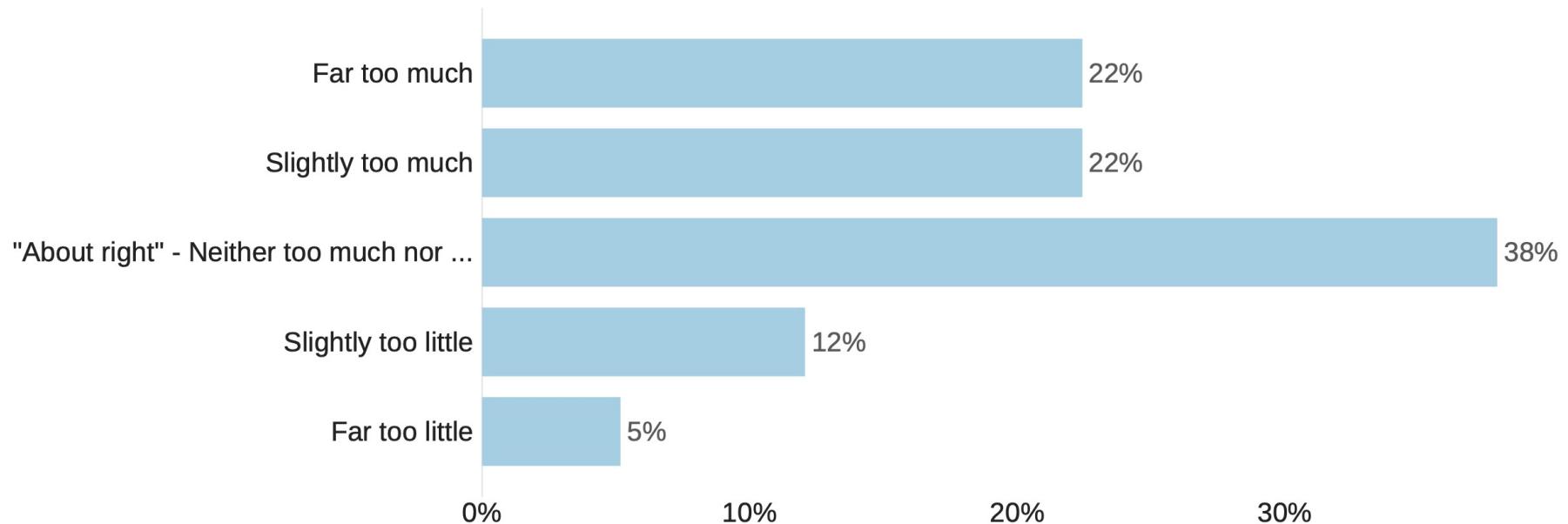
58 Responses



# *Results of the Survey*

On a weekly basis, how much time are you spending in this course?

58 Responses

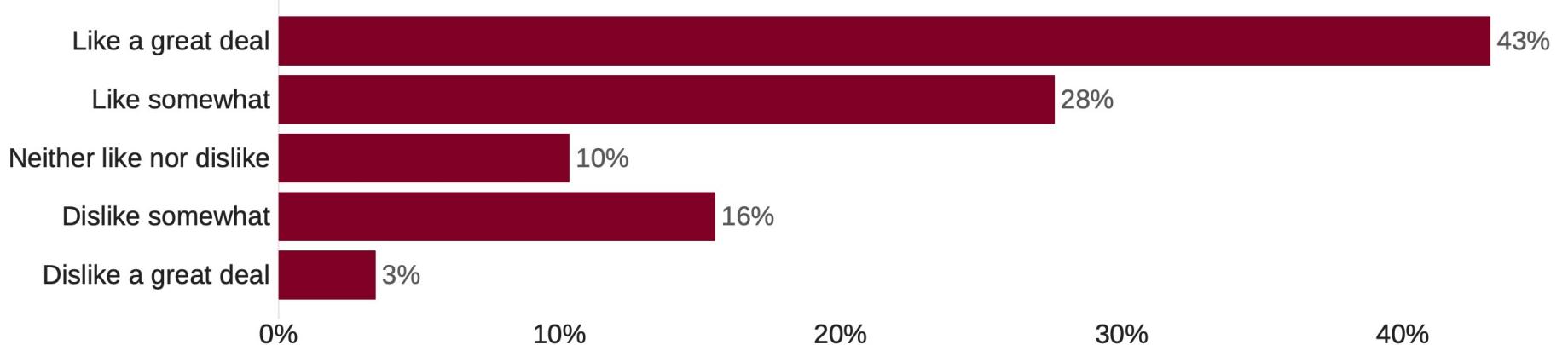


# *Results of the Survey*

---

What do you think of the Specifications-Grading system in the course? Specifically the E,G,R,I system & the unlimited submissions with no deadlines

58 Responses

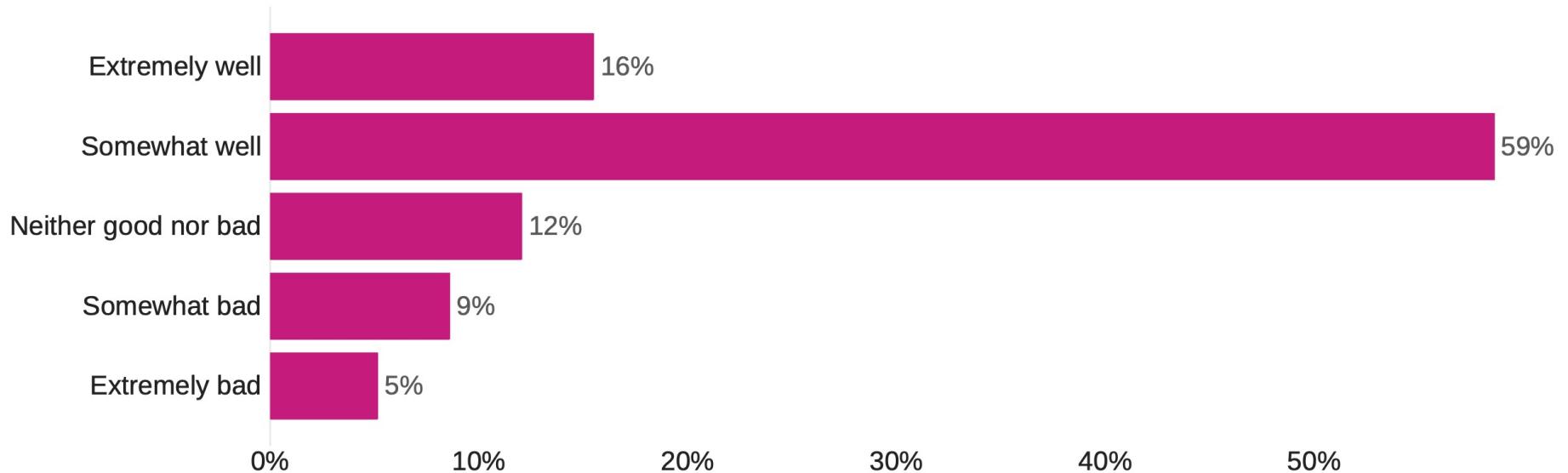


# *Results of the Survey*

---

How do you think the course is going so far (overall) ?

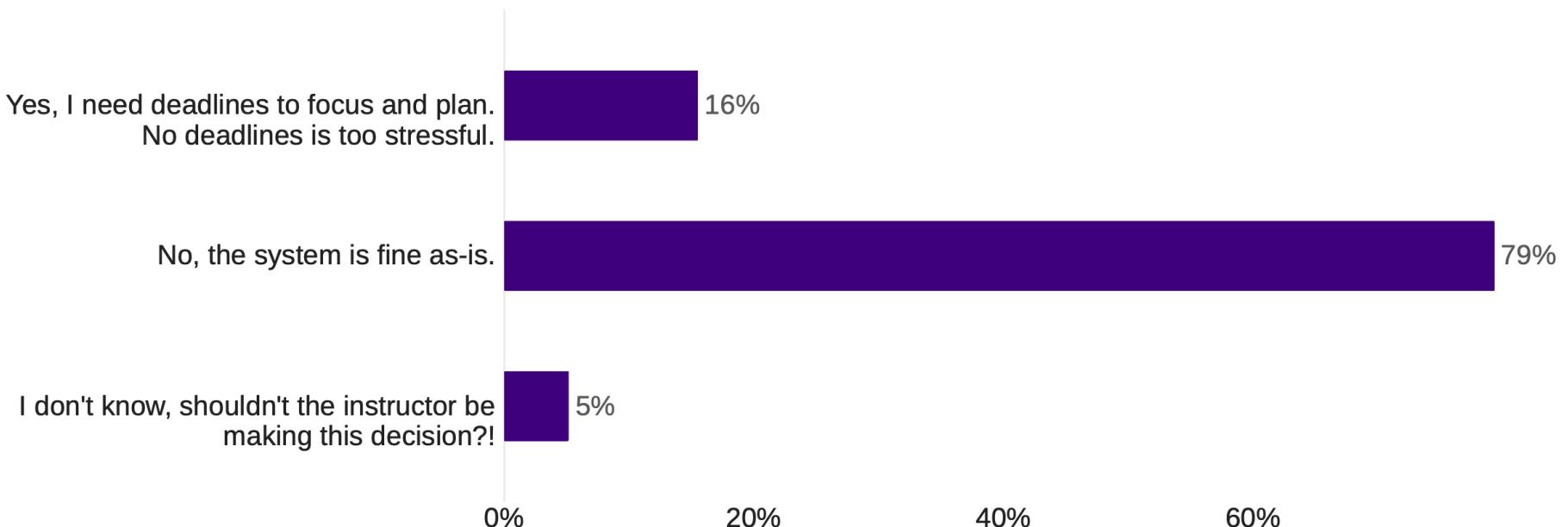
58 Responses



# *Results of the Survey*

As I mentioned, one of the problems with having no deadlines is that students may not have any motivation to keep up and with other pressures may give you TOO MUCH freedom. Would it help you if I try to do something to take back some of the freedom given to you (for e.g., by adding deadlines, limiting resubmissions, and only accepting resubmissions from those that submit by the deadline. etc...)?

58 Responses



# *Results of the Survey*

---

# **Results of Stop / Start / Continue**

Things the instructor should STOP doing	My Response
Putting in so many tasks in Lecture Activities and having one every week.	<i>There has to be one every week. I will review them, but if I reduce # of tasks in Lecture Activities, you will all struggle a LOT more in labs!</i>
Forgetting to upload recordings after class	<i>I am sorry and I am annoyed by this too. It takes 1-3 hrs after class for the recording to be available, and by the time it happens I forget. Feel free to remind me on Ed Discussion!</i>
Using so many tools in the course	<i>This is an unfortunate circumstance of every tool only being good at very few things. I will try in the future to have fewer tools.</i>
Pronouncing gif as "GIF" instead of "JIFF", who cares what the creator says it's pronounced.	<i>No :-P it's "Jiff" like Giraffe</i>
Nothing!	<i>Good to hear so many of you are happy with how things are going!</i>

# *Results of Stop / Start / Continue*

Things the instructor should START doing	My Response
Giving more hints on labs and lecture activities on how to start them.	<i>Starting this week, I will ask the TAs to start giving more hints at the start of every lab when they do an overview.</i>
Giving us an overview of our grades	<i>Okay, I will send you a progress report at some point this week!</i>
Making deadlines at midnight instead of 6 PM	<i>Unfortunately not, making midnight deadlines is bad for student health, well-being, sleep, stress, anxiety. 6 PM deadlines are intentional.</i>
Referencing the documentation for processing in lessons. I don't feel like every student is aware of how documentation works and it is very helpful.	<i>This is a VERY good point, I should do this more often and will make it a point to start now.</i>

# *Results of Stop / Start / Continue*

<b>Things the instructor should CONTINUE doing</b>	<b>My Response</b>
EGRI system - "I like not stressing about percentages"	<i>Excellent, I am very happy to hear that! I think that's the one thing that has definitely worked in the course.</i>
Examples and demos in class.	<i>Good to hear you like those! I'll try and do more live coding and demos in class.</i>
Music before/during class	<i>Always!</i>

# *Final Decision on March 4<sup>th</sup> Deadline*



DrMoosvi (Firas) INSTRUCTOR  
a day ago in Instructor Announcements

UNPIN

STAR

WATCHING

143  
VIEWS



Hello everyone!

Hope you all had a wonderful reading break, and got some rest - it's hard to keep track of days, but I think there were at least a few days of sun.

I'll discuss this more during Tuesday's class, but I wanted to let everyone know that after a lot of consideration and reflection:

**I have decided to make Friday March 4th at 6 PM the absolute latest cut-off for Activities 1-6 and Labs 1-6 (basically all the content before reading week).**

In keeping with the specs-grading system, you will still be able to make corrections and re-submit, but you will need to submit a ***solid and decent effort at all tasks*** in the labs and activities by Friday March 4th at 6 PM to earn the right to submit a resubmission and have it regraded.

**To be clear, this deadline is only applies to Activities 1-6 and Labs 1-6.** For the remaining labs and activities, we'll keep the last day of the class as the final date to submit stuff. Keep in mind though, that the later you submit your work, the less likely it is you'll get feedback in time to do a resubmissions.

**Resubmissions are not permitted past the last day of the class.**

# *Final Decision on March 4<sup>th</sup> Deadline*



DrMoosvi (Firas) INSTRUCTOR  
a day ago in Instructor Announcements

UNPIN

STAR

WATCHING

143  
VIEWS



Hello everyone!

Hope you all had a wonderful reading break, and got some rest - it's hard to keep track of days, but I think there were at least a few days of sun.

I'll discuss this more during Tuesday's class, but I wanted to let everyone know that after a lot of consideration and reflection:

**I have decided to make Friday March 4th at 6 PM the absolute latest cut-off for Activities 1-6 and Labs 1-6 (basically all the content before reading week).**

In keeping with the specs-grading system, you will still be able to make corrections and re-submit, but you will need to submit a ***solid and decent effort at all tasks*** in the labs and activities by Friday March 4th at 6 PM to earn the right to submit a resubmission and have it regraded.

**To be clear, this deadline is only applies to Activities 1-6 and Labs 1-6.** For the remaining labs and activities, we'll keep the last day of the class as the final date to submit stuff. Keep in mind though, that the later you submit your work, the less likely it is you'll get feedback in time to do a resubmissions.

**Resubmissions are not permitted past the last day of the class.**

**Friday March 4<sup>th</sup> at 6 PM will have a 48-hr grace period (if you need it)!**

# **If this is a problem...come see me during student hours for an exception...**

DrMoosvi (Firas) **INSTRUCTOR** 2h

Hello, thanks for sharing your perspective on this. I understand that predicament that you're in, and if you feel that the deadline policy causes you undue hardship, you (and anyone else) is more than welcome to come see me during my Student Hours after class and we can work out an alternate arrangement.

My goal in setting this deadline is to get students caught up with the course and is for your own benefit. I am happy to work out alternate arrangements on an individual case-by-case basis as is outlined [in the syllabus](#):

Let's talk, send me a private message and we can discuss it. I do not necessarily need to know all the personal details, just a high-level summary of your situation and what you think an ideal solution would be.

I'm sure we will come to some agreement, generally the earlier you let me know of any special circumstances or accommodation, the more I'll be able to do for you!

As long as you can explain to me what your plan is to complete the course content in a timely manner and get feedback, I will likely grant you an extension (on a case-by-case basis). Remember my goal is to help you succeed and I am legitimately worried for students that have not yet submitted any of the course material by this time - regardless of whether you're in 4th year or 1st year.

as well as the general deadline policy:

If you miss a deadline by more than the grace period, the general course policy is that you will not be able to get full credit for it, and in many cases, may even get a 0 for it. In the cases of Tests, it is not possible to get partial credit, or complete it at times other than within the scheduled windows. In some cases, I reserve the right to grant an extension or make alternate accommodations as needed.

# Reminder

---

## Passing requirements

- All students must satisfy ALL conditions to pass the course:
  1. Obtain an average grade of at least 50% on the Labs,
  2. Obtain an average grade of at least 50% on the Test and Exam components together,
  3. Obtain an average grade of at least 40% on the Final Exam,
  4. Obtain a grade of at least 50% on the whole course.

If students do not satisfy the appropriate requirements, the student will be assigned the **lower** of their earned course grade or, a maximum overall grade of 45% in the course.

# Announcements

---

- No Tests this week!
- Classes, Labs, Student hours all resume this week.
- Reminder to keep up on labs and activities!
  - According to the suggested schedule, you should be done Week 6 Labs and Activities
  - This week Week 8 Lab and Week 8 Activity have been released
- Results from the mid-course survey are next
- Labs 1-6 and Activities 1-6 must be submitted by Friday March 4<sup>th</sup> at 6 PM + 48 hour grace period (if you need it)

# Objectives

- After going through these, you should be able to:
  - Write and evaluate conditions. This includes using:
    - relational operators (==, >=, etc.)
    - AND, OR, and NOT operators (to write complex conditions).
    - nested conditional statements.
  - Make decisions in your program using if-else statement
  - Understand and use the “IDEAS” presented in this set of notes.
- *Today is a revision on conditionals + useful ideas for your animation*
  - *This reading is basically a revision on topics discussed in COSC 111, 122*
  - *The only addition is the application of these topics in the context of our course (Processing).*



# Making Decisions

- **Decisions** are used to allow the program to perform different actions in certain conditions.
  - For example, if a person applies for a driver's license and is not 17, then the computer should not give them a license.
- To make a decision in a program we must do several things:
  - 1) Determine the **condition** used to make the decision.
    - E.g. in a website for applying for a driving license, what is the appropriate age to for an applicant to be eligible?
  - 2) Tell the computer what to do if the condition is true or false.
    - A decision always has a Boolean value or true/false answer.
    - E.g. in the above example, if age > 16, allow user to apply.
- There are several keywords that can be used to make a decision such as the **if** statement.

# *Making Decisions: Comparisons*

- **Relational operators** compare two items called operands.
  - Syntax: operand1 operator operand2
- Comparison operators:
  - > - Greater than
  - >= - Greater than or equal
  - < - Less than
  - <= - Less than or equal
  - == - Equal (Note: not "=" which is used for assignment)
  - != - Not equal
- The result of a comparison is a **Boolean value** which is either **true** or **false**.

# Using Boolean Variables

```
int j=25, k = 45;
double d = 2.5, e=2.51;
boolean result;

result = (j == k);           // false
result = (j <= k);          // true
result = (d != e);          // true
result = (k >= 25);         // true
result = (25 == j);          // true
result = (j > k);           // false
result = (e < d);           // false
j = k;
result = (j == k);          // true
```

# Making Decisions: *if* Statement

- To make decisions with conditions, we use the **if** statement.
  - If the condition is **true**, the statement(s) after **if** are executed otherwise they are skipped.
  - If there is an **else** clause, statements after **else** are executed if the condition is **false**.

- Syntax: **if (condition)  
statement;**

OR

```
if (condition)  
statement;  
else  
    statement;
```

- Example

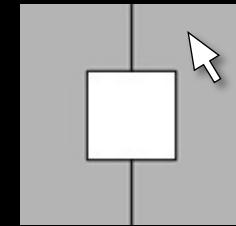
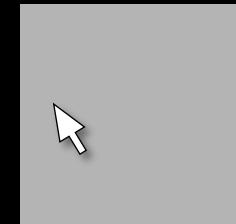
```
//draw rectangle only when  
//mouse is in right half of window  
  
void draw() {  
    background(180);  
    if(mouseX > width/2)  
        rect(30, 30, 40, 40);  
}
```

```
//draw rectangle or ellipse based  
//on where the mouse is  
  
void draw() {  
    background(180);  
    if(mouseX > width/2)  
        rect(30, 30, 40, 40);  
    else  
        ellipse(50, 50, 40, 40); }
```

# *Making Decisions: Block Syntax*

- If there are multiple statements in the if or else parts, we must use the block syntax. A block starts with “{“ and ends with “}”.
  - All statements inside the brackets are grouped together.
- Example:

```
//draw a rectangle and a line only when  
//mouse is in right half of window  
void draw() {  
    background(180);  
  
    if(mouseX > width/2) {  
        line(width/2, 0, width/2, height);  
        rect(30, 30, 40, 40);  
    }  
}
```

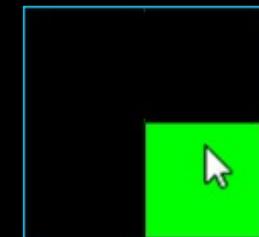
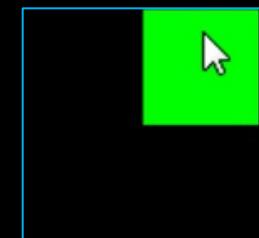
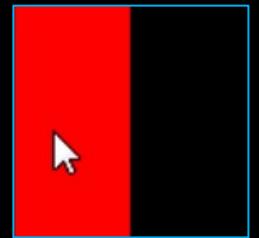


- We will use block statements in many other situations as well.

# Nested *if* Statement

- We nest if statements for more complicated decisions.

```
void draw() {  
    background(0);  
    if (mouseX < width/2) {  
        fill(255,0,0);  
        rect(0,0,width/2,height);  
    }else{  
        fill(0,255,0);  
        if (mouseY < height/2)  
            rect(width/2, 0, width/2, height/2);  
        else  
            rect(width/2, height/2, width/2, height/2);  
    }  
}
```



# Dangling else Problem

- The dangling else problem occurs when a programmer mistakes an else clause to belong to a different if statement than it really does.
- You can use blocks (brackets) to determine which statements are grouped together
- Example:** we want `else` to belong to first if.

Wrong

```
if (grade >= 50)
    if (grade >= 95)
        state = "With Honours";
else // Belongs to 2nd if
    state = "Fail";
```

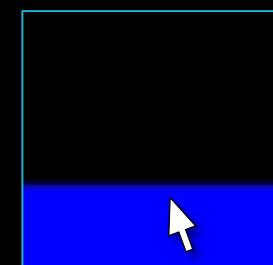
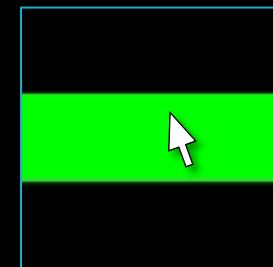
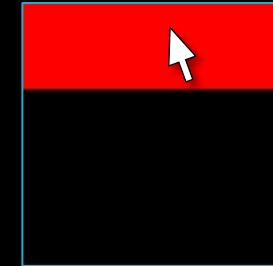
Correct

```
if (grade >= 50){
    if (grade >= 95){
        state = "With Honours";
    }
} else { // Belongs to 1st if
    state = "Fail";
}
```

# *Multi-part if statement*

- You can test multiple conditions using “**else if**” as in the example below

```
void draw() {  
    background(0);  
    noStroke();  
    if (mouseY < height/3) {  
        fill(255,0,0);  
        rect(0,0,width,height/3);  
    } else if (mouseY < 2*height/3){  
        fill(0,255,0);  
        rect(0,height/3,width,height/3);  
    } else {  
        fill(0,0,255);  
        rect(0,2*height/3,width,height/3);  
    }  
}
```



# *Boolean Expressions*

- A Boolean expression is a sequence of conditions combined using AND (`&&`), OR (`||`), and NOT (`!`).
- Allows you to test more complex conditions
- Group subexpressions using parentheses
- Syntax:
  - `(expr1) && (expr2)` - expr1 AND expr2
  - `(expr1) || (expr2)` - expr1 OR expr2
  - `!(expr1)` - NOT expr1
- Examples:

```
boolean b;  
b = (x > 10) && !(x < 50);  
b = (month == 1) || (month == 2) || (month == 3);  
if (day == 28 && month == 2);  
if !(num1 == 1 && num2 == 3);  
b = ((10 > 5 || 5 > 10) && ((10>5 && 5>10)); // False
```



## *IDEA1: Deciding based on System Variables*

# Draw Only When Mouse is Pressed

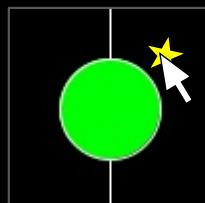
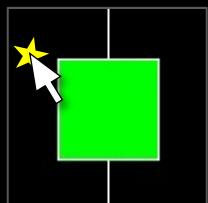
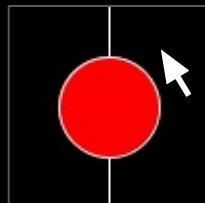
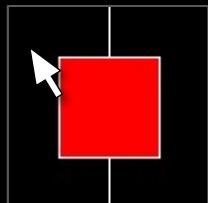
- You previously wrote a program to draw a continuous line using the *current* and *previous* mouse coordinates.
- The previous program can be modified so that the mouse draws only **IF** the mouse is pressed (using the system variable **mousePressed**).

```
void setup() {  
    size(400,400); background(255);  
}  
void draw() {  
    if(mousePressed)  
        line(pmouseX,pmouseY,mouseX,mouseY);  
}
```



# Changing Color Only when Mouse is Pressed

- The system variable **mousePressed** is used to determine the fill color.
  - "if mouse is pressed, color is red otherwise color is blue"*
- mouseX** is used to determine which shape to draw
  - "if mouse is on right, draw a circle otherwise draw a rect"*

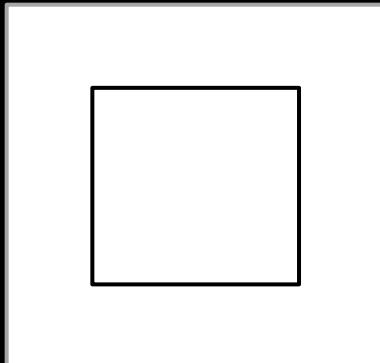


```
void setup() {  
    rectMode(CENTER);  
    stroke(255);  
}  
  
void draw() {  
    background(0);  
    line(width/2, 0, width/2, height);  
  
    //Set Color based on whether mouse is pressed  
    if (mousePressed)  
        fill(0, 255, 0);  
    else  
        fill(255, 0, 0);  
  
    //Draw Shape based on mouse position  
    if (mouseX > width/2)  
        ellipse(width/2,height/2,50,50);  
    else  
        rect(width/2,height/2,50,50);  
}
```

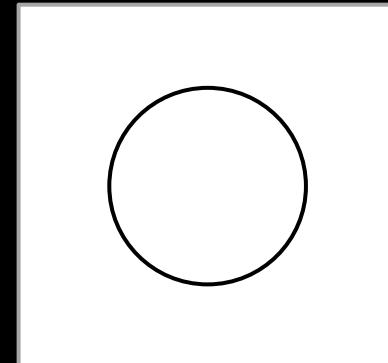
# Changing Shape if Key is Pressed

- If any key is pressed, draw a rectangle.
- Otherwise, draw a circle

```
void draw() {  
    background(255);  
    if (keyPressed)  
        rect(20, 20, 60, 60);  
    else  
        ellipse(50, 50, 80, 80);  
}
```



keyPressed = true



keyPressed = false

## *IDEA2: Moving Objects with Arrow Keys*

# Moving Objects with Arrow Keys Version1

**The key idea is as follows:**

- Draw an item at (x,y).  
Update x & y with speedX  
and speedY in every  
frame.
- When an arrow key is  
pressed:
  - Set speedX/Y to non-  
zero values.
- When key is released
  - Set speedX/Y to 0

```
float x=50, y=50, speedX=0, speedY=0;  
void draw() {  
    background(0);  
    ellipse(x,y,30,30);  
    x += speedX;  
    y += speedY;  
}  
void keyPressed(){  
    if (keyCode == LEFT) speedX = -5;  
    if (keyCode == RIGHT) speedX = 5;  
    if (keyCode == UP) speedY = -5;  
    if (keyCode == DOWN) speedY = 5;  
}  
void keyReleased(){  
    if(keyCode==LEFT | | keyCode==RIGHT) speedX = 0;  
    if(keyCode==DOWN | | keyCode==UP) speedY = 0;  
}
```

**Try this code →  
on your computer!!**



# Moving Objects with Arrow Keys Version2

This is the same code as the previous example except we use the `constrain` function to keep the object within the boundaries of the sketch.

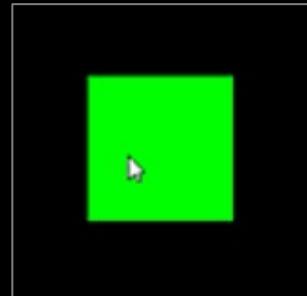
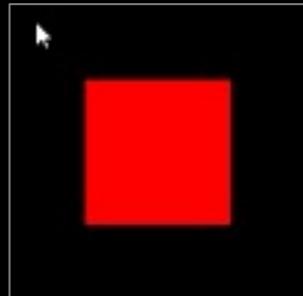
```
float x=50, y=50, speedX=0, speedY=0;  
void draw() {  
    background(0);  
    ellipse(x,y,30,30);  
    x = constrain(x+speedX,0,width);  
    y = constrain(y+speedY,0,height);  
}  
void keyPressed(){  
    if (keyCode == LEFT) speedX = -5;  
    if (keyCode == RIGHT) speedX = 5;  
    if (keyCode == UP) speedY = -5;  
    if (keyCode == DOWN) speedY = 5;  
}  
void keyReleased(){  
    if(keyCode==LEFT||keyCode==RIGHT) speedX = 0;  
    if(keyCode==DOWN||keyCode==UP) speedY = 0;  
}
```



## *IDEA 3: Detecting Mouse Movement Over Objects*

# Detecting Mouse Over a Rectangle

- The example shows how to detect the mouse moving over a shape (rectangle).
- When the mouse moves over the shape, the color changes to green. When the mouse is off the shape, the color changes back to red.



```
float x = 50, y = 50, w = 100, h = 100;  
void setup(){  
    size(200,200);  
}  
void draw(){  
    background(0);  
    //set color based on mouse position  
    if(mouseX > x && mouseX < x+w &&  
        mouseY > y && mouseY < y+h)  
        fill(0,255,0);  
    else  
        fill(255,0,0);  
    //draw rectangle  
    rect(x,y,w,h);  
}
```

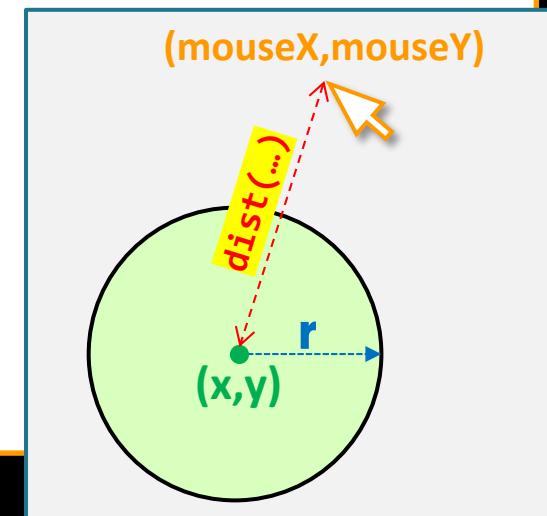
## Example

# Detecting Mouse Over a Circle

- This code is similar to previous code except that it detects the mouse presence over a *circular* area.
- The `dist()` function is used to check the distance between the mouse location and the circle center (i.e. to check if mouse is within the circumference of the circle.)



```
float x = 100, y = 100, r = 50;  
void setup(){  
    size(200,200);  
}  
void draw(){  
    background(0);  
    //set color based on mouse position  
    if(dist(mouseX,mouseY,x,y) < r)  
        fill(0,255,0);  
    else  
        fill(255,0,0);  
    //draw circle  
    ellipse(x,y,2*r,2*r);  
}
```



## Example

# Multiple Rollover

- In this code, we use if statement to highlight the one quarter of the window that is under the mouse.

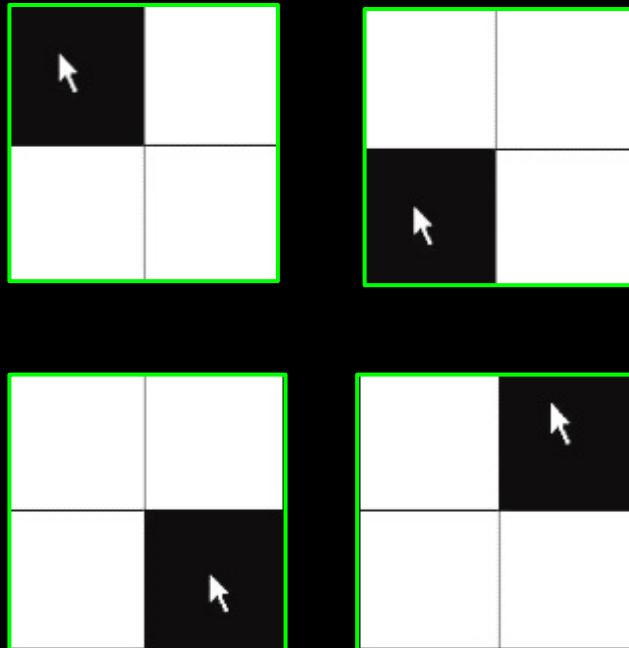


Figure 5.7

```
void setup() {
    size(200, 200);
}
void draw() {
    background(255);
    stroke(0);
    line(100, 0, 100, 200);
    line(0, 100, 200, 100);
    // Fill a black color
    noStroke();
    fill(0);
    if (mouseX < 100 && mouseY < 100)
        rect(0, 0, 100, 100);
    else if (mouseX > 100 && mouseY < 100)
        rect(100, 0, 100, 100);
    else if (mouseX < 100 && mouseY > 100)
        rect(0, 100, 100, 100);
    else if (mouseX > 100 && mouseY > 100)
        rect(100, 100, 100, 100);
}
```

Source: Shiffman

## *IDEA 4: Clickable Buttons*

# Designing a Clickable Button *version1*

- This code is similar to previous example except that the circle (button) turns green if two conditions are satisfied:
  - Mouse is over the button
  - Mouse is pressed
- This simulates the visuals of a button being clicked



```
float x = 100, y = 100, r = 50;  
  
void setup(){  
    size(200,200);  
}  
  
void draw(){  
    background(0);  
    //set color based on mouse position  
    if(dist(mouseX,mouseY,x,y)<r && mousePressed)  
        fill(0,255,0);  
    else  
        fill(255,0,0);  
    //draw circle  
    ellipse(x,y,2*r,2*r);  
}
```

## Example

# Designing a Clickable Button version2

- This code is the same as the previous example except with additional visual components:
  - ON/OFF text on the button
  - Button gets smaller when clicked.

*see orange code on the right*



```
float x = 100, y = 100, r = 50; String s = "OFF";
void setup(){
    size(200,200);
    textSize(28); textAlign(CENTER,CENTER);
}

void draw(){
    background(0);
    //set color based on mouse position & if clicked
    if(dist(mouseX,mouseY,x,y)<r && mousePressed){
        fill(0,255,0);
        s = "ON";
        r = 45;
    }else{
        fill(255,0,0);
        s = "OFF";
        r = 50;
    }
    //draw button
    ellipse(x,y,2*r,2*r);
    fill(255);
    text(s,x,y);
}
```

## Example

# Designing a Clickable Button version3

- This is the same as version2 of this example except a **Boolean variable** is used to “remember” whether the button is clicked.



```
float x = 100, y = 100, r = 50; String s = "OFF";
boolean clicked = false;
void setup(){
    size(200,200);
    textSize(28); textAlign(CENTER,CENTER);
}
void draw(){
    background(0);
    //set clicked
    if(dist(mouseX,mouseY,x,y)<r && mousePressed)
        clicked=true;
    else
        clicked=false;
    //set button attributes
    if(clicked){
        fill(0,255,0); s = "ON"; r = 45;
    }else{
        fill(255,0,0); s = "OFF"; r = 50;
    }
    //draw button
    ellipse(x,y,2*r,2*r);
    fill(255);
    text(s,x,y);
}
```

# Designing a Clickable Button version4

- Here is another way that produces the same output as the previous example, but using **boolean variable** that is set in two methods: **mousePressed** and **mouseReleased**.
- The **benefit of using a Boolean variable** is that its can be accessed in different methods.



```
float x = 100, y = 100, r = 50; String s = "OFF";
boolean clicked = false;
void setup(){
    size(200,200);
    textSize(28); textAlign(CENTER,CENTER);
}
void draw(){
    background(0);
    if(clicked){
        fill(0,255,0); s = "ON"; r = 45;
    }else{
        fill(255,0,0); s = "OFF"; r = 50;
    }
    ellipse(x,y,2*r,2*r);
    fill(255);
    text(s,x,y);
}
void mousePressed(){
    if(dist(mouseX,mouseY,x,y)<r && mousePressed)
        clicked=true;
    else
        clicked=false;
}
void mouseReleased(){
    clicked=false;
}
```

## *IDEA 5: Toggle Buttons*

## Example

# Designing a Toggle Button

- A toggle button holds one of two states. Clicking the button alternates the state.
- The state of the button can be stored in a **Boolean variable**, and the looks of the button as well as other decisions can be determined based on that variable.

```
boolean buttonActive = false; // initial button state
int x = 50, y = 50, r = 40; // button attributes
void setup() {
    size(200, 200);   strokeWeight(5);
    textSize(25);    textAlign(CENTER,CENTER);
}
void draw() {
    background(0);
    if(buttonActive) { // make decisions based on button state
        fill(0, 200, 0); stroke(0,255,0); ellipse(x,y,2*r,2*r);
        fill(200,255,200);text("ON",x,y);
    }else{
        fill(180, 0, 0); stroke(255,0,0); ellipse(x,y,2*r,2*r);
        fill(100,0,0); text("OFF",x,y);
    }
}
void mousePressed() {
    if(dist(mouseX,mouseY,x,y)<r)
        buttonActive = ! buttonActive; // change button state
}
```





## *IDEA 6: Bouncing “Attributes”*

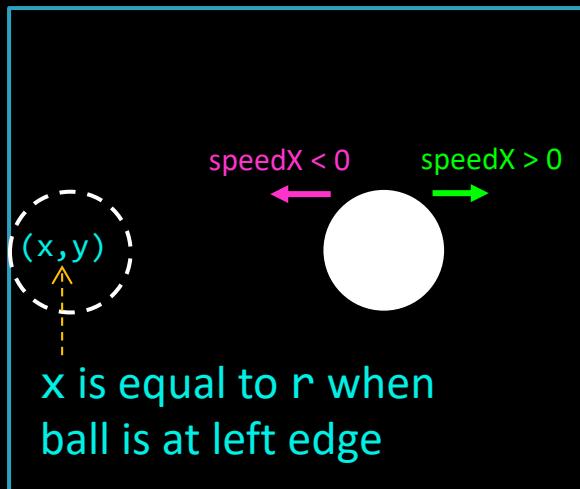
## Example

# Bouncing Ball version 1

The code shows a ball bouncing off the edges. For simplicity, the ball moves only horizontally.

### IDEA:

- increment the `x` position by `speedX`.
- IF the ball reaches the right OR left edge, reverse the motion direction (by changing the sign of `speedX`).

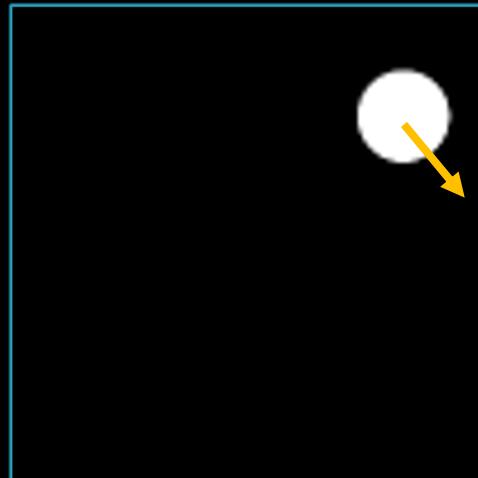


```
float speedX = 3;  
float x=20, y=100, r = 20;  
void setup(){  
    size(200,200);  
}  
void draw(){  
    background(0);  
    ellipse(x,y,2*r,2*r);  
    //increment x  
    x += speedX;  
    //if ball at edge, reverse direction  
    if( x > width-r || x < r )  
        speedX = -speedX;  
}
```

# Bouncing Ball version 2

This is the same as previous example except that the ball moves in all directions.

- i.e. y and speedY also change.



```
float speedX = 1, speedY = 2;  
float x=20, y=100, r = 20;  
void setup(){  
    size(200,200);  
}  
void draw(){  
    background(0);  
    ellipse(x,y,2*r,2*r);  
    //increment x,y  
    x += speedX;  
    y += speedY;  
    //if ball at edge, reverse direction  
    if( x > width-r || x < r )  
        speedX = -speedX;  
    if(y > height-r || y < r)  
        speedY = -speedY;  
}
```

## Example

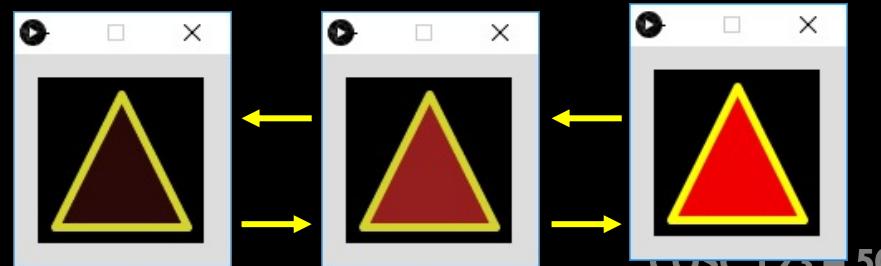
# Bouncing Other Attributes

The logic used in the Bouncing Ball example can be applied to bounce any attribute once it reaches its limits.

### Example: Flashing Color

- In this example, we **bounce the fill color** once its value reaches some limits (50 to 245). The output is a flashing triangle that looks like an alert signal.

```
float c = 100;           //color attribute
float speedC=10;         //speed of color change
void setup() {
    size(100, 100);
    strokeWeight(5);
    stroke(255,255,0);
    strokeJoin(ROUND);
}
void draw() {
    background(0);
    fill(c,0,0);
    triangle(50,10,10,90,90,90);
    c += speedC;
    if(c<50 || c>245)
        speedC = - speedC;
}
```



# *Using Conditionals in Sketches*

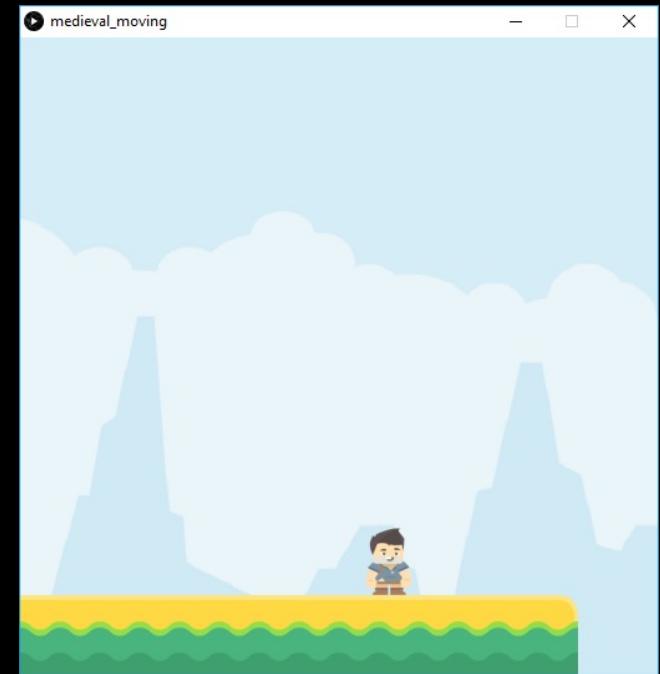
## Move Your Player

Previously, you created the game platform. For today, do the following:

- 1) Use your solution from the previous exercise
- 2) Update the code so that the player ***moves left or right with the arrow keys.***

The player should be moving as long as an arrow key is pressed and should stop moving if the key is released.

***Hint:*** use *IDEA2*



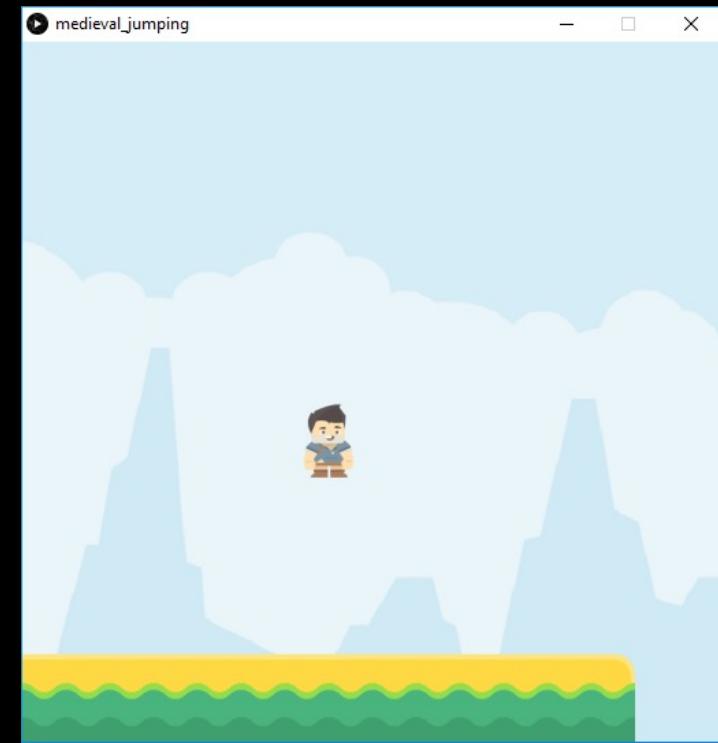
Tiles from kenney.nl

# Jump.. Jump.. High Up in the Sky!!

Previously, you created the game platform and added code to ***move the player left or right with the arrow keys.***

1) Open your solution from the previous exercise

2) ***Add more code to make your player jump when SPACE is pressed. Note that a player cannot jump if already jumping***



Tiles from kenney.nl

*End of Tuesday's Class*

Computer Creativity

# *Conditionals*



Okanagan

Slides courtesy of Dr. Abdallah Mohamed.

# Announcements

---

- No Tests this week!
- Classes, Labs, Student hours all resume this week.
- Reminder to keep up on labs and activities!
  - According to the suggested schedule, you should be done Week 6 Labs and Activities
  - This week Week 8 Lab and Week 8 Activity have been released
- Results from the mid-course survey are next
- Labs 1-6 and Activities 1-6 must be submitted by Friday March 4<sup>th</sup> at 6 PM + 48 hour grace period (if you need it)

# *Let's discuss the most confusing ideas*

- IDEA1: Deciding based on system variables
  - mousePressed, keyPressed, mouseX, ...etc
- IDEA2: Moving objects with arrow keys
  - keyPressed() and keyReleased() along with keyCode
- IDEA3: Detecting mouse movement over objects
  - comparing mouseX,mouseY to object location
  - The use of dist() function
- IDEA4 & 5: Creating Buttons (clickable and toggle)
  - Using a Boolean variable to “remember” the status of the button
- IDEA6: Bouncing Attributes
  - Reverse how the attribute changes once a limit is reached.

# Example on Bouncing Attributes

- This code bounces four attributes:
  - x location
  - y location
  - Shade (color)
  - Size (diameter)

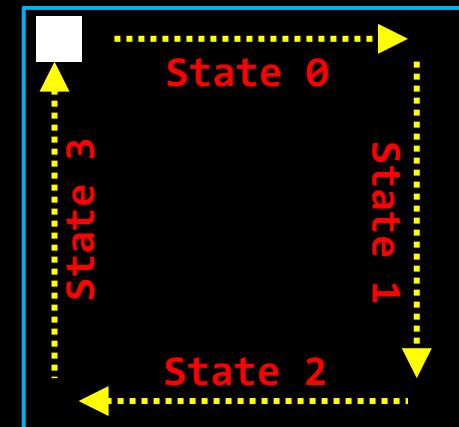
```
float x =100, speedX = 4;
float y =100, speedY = 3;
float sh =100, speedSh = 20;
float d =35, speedD = 5;
void setup(){size(200,200);}
void draw(){
    background(0);
    //1) USE variable attributes to draw
    fill(sh);
    ellipse(x,y,d,d);
    //2) UPDATE attributes
    x += speedX;
    y += speedY;
    sh += speedSh;
    d += speedD;
    //3) BOUNCE attributes that reach their limits
    //once you hit an edge, then reverse the speed
    if(x>=width-d/2 || x<=d/2) //did the ball hit a vertical edge?
        speedX = -speedX;
    if(y>=height-d/2 || y<=d/2) //did the ball hit a horizontal edge?
        speedY = -speedY;
    //once you reach the limits of [0,255], reverse the speedSh
    if(sh>=255 || sh<=0)
        speedSh = -speedSh;
    //once size is outside limits, reverse speedD
    if(d>=60 || d<=30)
        speedD = -speedD;
}
```



## *IDEA 7: Designing Complex Motion Paths*

# Designing Complex Motion Paths

- One can design complex motion paths using conditionals.
- For example, assume we want to move an item along the dotted path as in the shown figure (i.e. around the window).
- One way to solve this problem is to
  - Create a variable “state” that ‘remembers’ which direction the rectangle is going.
    - We have 4 rectangle states: 0 to 3. Each state will represent the motion in one directions.
  - use the variable to decide how the rectangle moves.
- The code for doing this is on the next slide.



# Designing Complex Motion Paths (2)

```
float d = 20, x = d, y = d; // location of ball
float speed=5, state=0;      // initial speed and state of square
void setup() {size(200, 200); noStroke(); fill(255);}

void draw() {
    background(0);
    ellipse(x, y, d, d);
    if(state==0){           // RIGHT state
        x += speed;        // keep moving right
        if (x > width-d) { // when the square reaches the right edge:
            state = 1;       //   a) switch to DOWN state
            x = width-d;     //   b) align square for accuracy
        }
    }else if(state==1){     // DOWN state
        y = y + speed;
        if (y > height-d) {y=height-d; state=2;}
    }else if(state==2){     // LEFT state
        x = x - speed;
        if (x < d) {x=d; state=3;}
    }else if(state==3){     // UP state
        y = y - speed;
        if (y < d) {y=d; state=0;}
    }
}
```

# Exercise

## Again..

- Same as previous example but we are controlling other aspects of the animation based on the object state.

```
int x = 15, y = 15, r = 15, speed = 2;
int state = 0;

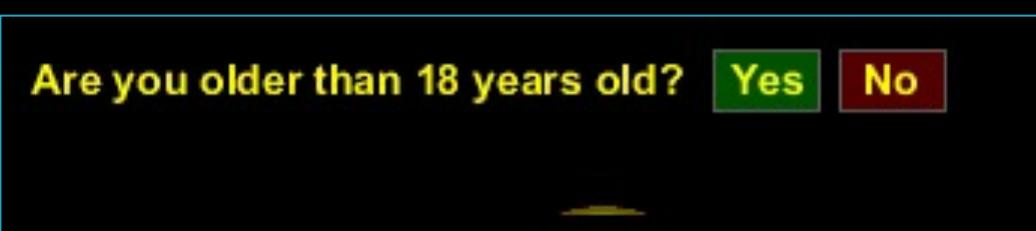
void setup(){
    size(300,300);
    rectMode(CENTER);
}

void draw(){
    //control background COLOR based on object state
    if(state==0) background(0);
    else if(state ==1) background(255,0,0);
    else if(state ==2) background(0,255,0);
    else background(0,0,255);
    //control object SHAPE and SIZE based on state
    if(state == 0 || state == 2)
        ellipse(x, y, 2*r+random(10), 2*r+random(10));
    else
        rect(x,y,2*r,2*r);
    // control object PATH based on state
    if(state == 0){
        x += speed;
        if(x >= width - r) state = 1; //switch state when reaching edge
    } else if(state == 1){
        y += speed;
        if(y >= height - r) state = 2;
    } else if(state == 2){
        x -= speed;
        if(x <= r) state = 3;
    } else if(state == 3){
        y -= speed;
        if(y <= r) state = 0;
    }
}
```

# *Using Buttons to Get User's Input*

# *Getting Input from the User*

- We can use the IDEA 4/5 (buttons) to read input from user.
- The code on the next page displays a question to the user with two possible answers, Yes and No.
- Whenever the user clicks a button, the button brightness increases and a message is displayed.
  - “Yes” displays in green “Congrats! You can get a driving license”
  - “No” displays in red “Sorry! You are too young to get a driving license”



Are you older than 18 years old? **Yes** **No**

# *Getting Input from the User, cont'd*

- The idea:
  - Draw two toggle buttons using the code from IDEA5 in the pre-class readings
    - Note that you will need two sets of variables, one for each button. i.e. we need `x1` for button Yes and `x2` for button No, and so on.
  - Create one variable, `answer`, that “remembers” which button is clicked.
    - e.g. `answer` should be 1 for Yes, 2 for No, and -1 otherwise
  - For each button, create a `color` variable.
  - When mouse is clicked, set `answer` to one of three values:
    - 1 if Yes is clicked
    - 2 if No is clicked
    - -1 if mouse is clicked away from the two buttons.
  - Use the value of `answer` to decide how to **draw ALL items** on the sketch (e.g. Yes should be bright green if answer is 1)

```
int answer = -1; // 1 for YES, 2 for NO, -1 for neither
final int x1 = 237, y1 = 5, x2 = 279, y2 = 5, W = 35, H = 20; //buttons dimensions
int color1, color2, colorMsg; //buttons & msg colors
String msg = "";

void setup() { size(350, 60); stroke(128); textAlign(createFont("Arial Bold", 14)); }

void draw() {
    background(0);
    // set drawing attributes based on answer
    if (answer == 1) { // YES button is active
        color1 = color(0,255,0); color2 = color(90,0,0); colorMsg = color1;
        msg = "Congratulations! You can get a driving license";
    } else if (answer == 2) { // NO button is active
        color1 = color(0,90,0); color2 = color(255,0,0); colorMsg = color2;
        msg = "Sorry! You are too young to get a driving license";
    } else { // neither button is active
        color1 = color(0,90,0); color2 = color(90,0,0);
        msg = "";
    }
    // Draw buttons and put text using above attributes
    fill(color1); rect(x1,y1,W,H);
    fill(color2); rect(x2,y2,W,H);
    fill(255,255,0); text("Are you older than 18 years old? Yes No ", 10, 20);
    fill(colorMsg); text(msg,10, 50);
}

void mouseReleased() {
    if (mouseX>x1 && mouseX<x1+W && mouseY>y1 && mouseY<y1+H) answer = 1;
    else if (mouseX>x2 && mouseX<x2+W && mouseY>y2 && mouseY<y2+H) answer = 2;
    else answer = -1;
}
```

## Making Decisions

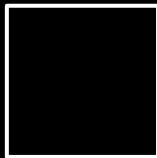
What is the output of this code?

```
noFill(); rectMode(CENTER); stroke(255);

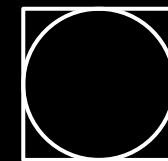
int num = 10;

if (num > 10)
    rect(50,50,50,50);
else
    ellipse(50,50,50,50);
```

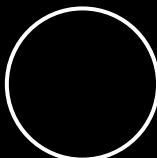
A.



C.



B.



D. Something else

## Making Decisions

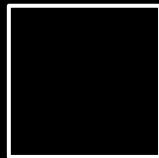
What is the output of this code?

```
noFill(); rectMode(CENTER); stroke(255);

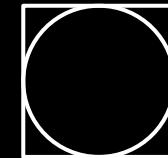
int num = 9;

if (num != 10)
    rect(50,50,50,50);
ellipse(50,50,50,50);
```

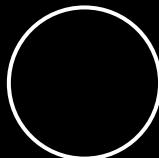
A.



C.



B.



D. Something else

## Testing Multiple conditions

What is the output?

```
int grade = 90;  
if (grade > 50)      print("D");  
else if (grade > 60) print("C");  
else if (grade > 70) print("B");  
else if (grade > 85) print("A");  
else                  print("F");
```

- A. A
- B. DCBA
- C. DCBAF
- D. D
- E. Something else

## Testing Multiple conditions

What is the output?

```
int grade = 90;  
if (grade > 85)      print("A");  
else if (grade > 70) print("B");  
else if (grade > 60) print("C");  
else if (grade > 50) print("D");  
else                  print("F");
```

- A. A
- B. ABCDF
- C. ABCD
- D. F
- E. Something else

## Boolean Expressions

Is **result** true or false?

```
int x = 10, y = 20;  
boolean result = (x > 10) || (y < 20);  
println(result);
```

- A.** true
- B.** false

## Boolean Expressions

Is **result** true or false?

```
int x = 10, y = 20;  
boolean result = !(x != 10) && (y == 20);  
println(result);
```

- A.** true
- B.** false

## Boolean Expressions

Is **result** true or false?

```
int x = 10, y = 20;  
boolean result = (x >= y) || (y <= x);  
println(result);
```

- A.** true
- B.** false

## Making Decisions

What is the output of this code?

```
int num=12;
if (num >= 8)
    print("big");
if (num == 10)
    print("ten");
else
    print("small");
```

- A. big
- B. small
- C. bigsmall
- D. ten
- E. bigten

# Objectives

- After reading these, you should be able to:
  - Use the switch statement.
  - Compare Strings and objects.

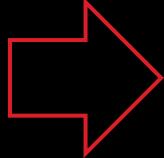


# *The Switch Statement*

# Making Decisions: the *Switch* Statement

- There are cases where you want to compare a single value against many alternatives. One way of doing this is to use a multi-part *if* statement. Another ways is to use `switch`.

```
if (x == value1) {  
    statements;  
}  
  
else if (x == value2) {  
    statements;  
}  
  
... //other else-if  
  
else {  
    statements;  
}
```

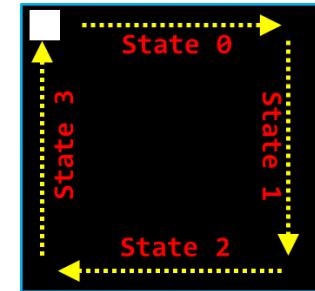


```
switch (x) {  
    case value1:  
        statements;  
        break;  
  
    case value2:  
        statements;  
        break;  
  
    ... //other cases  
  
    default:  
        statements;  
}
```

# Complex Motion Paths Using if

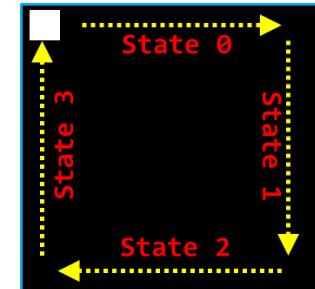
```
int x = 0, y = 0, speed=5, state=0;
void setup() { size(200, 200);   noStroke();   fill(255); }
void draw() {
    background(0);
    rect(x, y, 10, 10);

    if(state == 0)           // RIGHT state
    {   x += speed;
        if (x > width-10) { x = width-10; state = 1; }
    }
    else if(state == 1)      // DOWN state
    {   y = y + speed;
        if (y > height-10) { y = height-10; state = 2; }
    }
    else if(state == 2)      // LEFT state
    {   x = x - speed;
        if (x < 0)          { x = 0;         state = 3; }
    }
    else if(state == 3)      // UP state
    {   y = y - speed;
        if (y < 0)          { y = 0;         state = 0; }
    }
}
```



# Complex Motion Paths Using `switch`

```
int x = 0, y = 0, speed=5, state=0;
void setup() { size(200, 200);   noStroke();   fill(255); }
void draw() {
    background(0);
    rect(x, y, 10, 10);
    switch(state) {
        case 0: // RIGHT state
            x += speed;
            if (x > width-10) { x = width-10; state = 1; }
            break;
        case 1: // DOWN state
            y = y + speed;
            if (y > height-10) { y = height-10; state = 2; }
            break;
        case 2: // LEFT state
            x = x - speed;
            if (x < 0) { x = 0; state = 3; }
            break;
        case 3: // UP state
            y = y - speed;
            if (y < 0) { y = 0; state = 0; }
    }
}
```



# Switch Statement

```
int num = 2;

switch (num){
    case 1: text("Sunday", 10, 10); break;
    case 2: text("Monday", 10, 10); break;
    case 3: text("Tuesday", 10, 10); break;
    case 4: text("Wednesday", 10, 10); break;
    case 5: text("Thursday", 10, 10); break;
    case 6: text("Friday", 10, 10); break;
    case 7: text("Saturday", 10, 10); break;
    default: text("Invalid day!", 10, 10); break;
}
```

**Output:** Monday

# Ordering the Cases

In most situations, Order doesn't matter!

```
int num = 2;  
  
switch (num){  
    case 3: text("Tuesday", 10, 10); break;  
    case 1: text("Sunday", 10, 10); break;  
    case 4: text("Wednesday", 10,10); break;  
    case 6: text("Friday", 10, 10); break;  
    default: text("Invalid day!",10,10);break;  
    case 2: text("Monday", 10, 10); break;  
    case 7: text("Saturday", 10, 10); break;  
    case 5: text("Thursday", 10, 10); break;  
}
```

**Output:** Monday

# The use of break

- Each **case** usually ends with a **break** statement which means: “exit the switch block right after finishing this case”.
  - Execution of each case continues until the **break** statement. If you don’t put a break statement, the next case will be executed.

```
int x = 0;  
switch (x){  
    case 0: print("0"); break;  
    case 1: print("1"); break;  
    default: print("X"); break;  
}
```

Output: 0

```
int x = 0;  
switch (x){  
    case 0: print("0");  
    case 1: print("1");  
    default: print("X");  
}
```

Output: 01X

# *Limitations*

---

- You can use switch statement **only when:**
  - 1) You are comparing values for equality
    - Can't compare using `>`, `<`, `<=`, etc
  - 2) The values are **integers** or **Strings**.

# *Comparing Strings and Objects*

# *Comparing Strings and Objects*

- Comparing strings and objects is different than numbers.
  - Operators such as <, > are not useful for strings and objects.
  - Operator “==“ can be used but it is not very useful.
    - The “==“ operator compares if two string/object references refer to the same object NOT if the string/object has the same value.
      - We will discuss more about this later
- To compare strings, use the equals() method:

```
String st1 = "Hello", st2="He", st3="Test", st4 ="He";  
  
println(st1.equals(st2));           // false  
println(st2.equals(st4));           // true  
  
st4 = st4.toUpperCase();           // st4 is now HE  
st2 = st2.toLowerCase();           // st2 is now he  
println(st2.equals(st4));           // false  
println(st2.equalsIgnoreCase(st4)); // true
```

## Switch Statement

What is the output of this code?

```
int num=2;  
switch (num){  
    case 1: print("one");      break;  
    case 3: print("three");   break;  
    case 2: print("two");     break;  
    default:print("other");  break;  
}
```

- A. one
- B. two
- C. three
- D. other

## Switch Statement

What is the output of this code?

```
int num=2;  
switch (num){  
    case 1: print("one");  
    case 2: print("two");  
    case 3: print("three");      break;  
    default:print("other");  
}
```

- A. one
- B. two
- C. twothree
- D. onetwothree
- E. other

## Switch Statement

What is the output of this code?

```
int num=2;  
switch (num){  
    case 1: print("one");  
    case 3: print("three");      break;  
    case 2: print("two");  
    default:print("other");     break;  
}
```

- A. three
- B. two
- C. twothree
- D. twoother
- E. other

## String Comparisons

What is the output of this code?

```
String str1 = new String("abc");
String str2 = new String("abc");
print(str1.equals(str2));
```

This is another way of creating strings that ensures new string objects have their own memory space.

- A. true
- B. false

## String Comparisons

What is the output of this code?

```
String str1 = new String("abc");
String str2 = new String("abc");

print(str1 == str2);
```

- A. true
- B. false

# *Physics 101*

# Physics 111 / 121

---

- We know from before that we update the location of an object every frame with its speed:

`y = y + speedY`

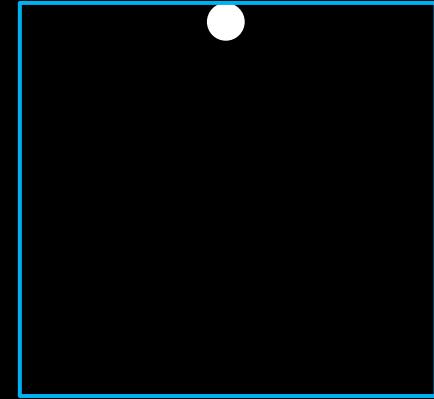
- Now, let's try to code “**gravity**”. Gravity is the *rate of change of the speed* (i.e. the acceleration). This means, to code gravity we need to **also change the speed every frame** by some amount
- `speedY = speedY + gravity; // e.g. gravity = 0.1`

- Now let's see how the results look like with and without gravity

# Bouncing Ball with Gravity (version 1)

```
float x = 150, y = 10, r = 15; // location of ball
float speedY = 0;           // speed of ball
float gravity = 0.2;
void setup() {
    size(300, 300);
}
void draw() {
    background(0);
    ellipse(x, y, 2*r, 2*r);
    // update ball's speed and location
    speedY += gravity; ←
    y += speedY;
    // reverse speed when ball hits the bottom
    if (y > height-r){
        y = height-r;      // ball can't penetrate ground
        speedY = -speedY; //not so natural looking!!
    }
}
```

comment this out to see  
how it looks without gravity



# *Physics 111/121, cont'd*

- The motion in the previous example doesn't look very natural.
- When the ball bounces off the ground there will be some loss of energy (speed) – this is known as **dampening effect**.
- This means, we need to replace:

**Speed = -speed**

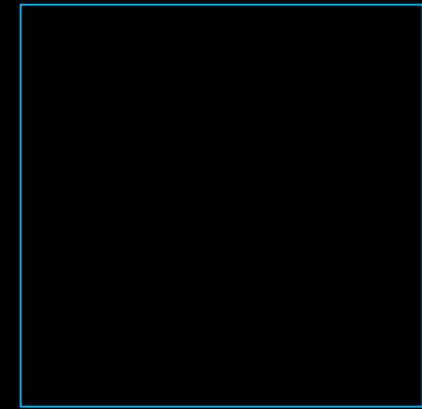
With

**speed = -0.9 \* speed; // or another dampening factor**

## Example

# Bouncing Ball with Gravity (version 2)

```
float x = 150, y = 10, r = 15; // ball location, size
float speedY = 0; // speed of ball
float gravity = 0.2;
void setup() {
    size(300, 300);
}
void draw() {
    background(0);
    ellipse(x, y, 2*r, 2*r);
    // update ball's speed and location
    speedY += gravity;
    y += speedY;
    // reverse speed when ball hits the bottom
    if (y > height-r){
        y = height-r; // ball can't penetrate ground
        speedY = - 0.9 * speedY; //natural looking
    }
}
```



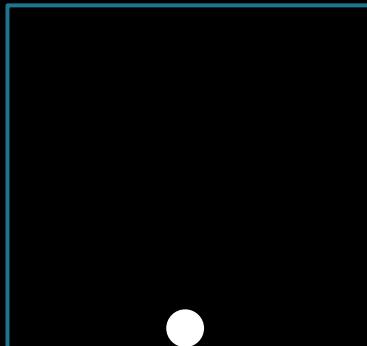
## Example

# Jumping Ball

In this example, pressing space bar causes the ball to jump.

The boolean variable `isJumping`

- used to track the state of the ball. The space bar sets this variable to true. The variable is set back to false when the ball lands on floor.
- The jumping action only happens in the `draw()` method when `isJumping` is true.



```
float x=150,y=285,speedY=0,gravity=0.15;  
boolean isJumping = false; int r = 15;  
void setup() { size(300, 300);}  
  
void draw() {  
    background(0);  
    ellipse(x, y, 2*r, 2*r);  
    if (isJumping) {  
        speedY += gravity;  
        y += speedY;  
        if (y >= height-r) { //ball lands on floor  
            speedY = 0; y = height-r;  
            isJumping = false;  
        }  
    }  
}  
  
void keyPressed() {  
    if (key == ' ' && !isJumping) {  
        isJumping = true;  
        speedY = -6; //go up  
    }  
}
```

## Example

# Jumping Ball Step-by-Step

```
float r=20, x=200, y =400-r;  
float speedY=-6, gravity=0.2;  
  
void setup(){  
    size(400,400);  
}  
  
void draw(){  
    background(0);  
    ellipse(x,y,2*r,2*r);  
  
    speedY += gravity;  
    y += speedY;  
}
```

```
float r=20, x=200, y =400-r;  
float speedY=-6, gravity=0.2;  
boolean jumping = true;  
void setup(){  
    size(400,400);  
}  
void draw(){  
    background(0);  
    ellipse(x,y,2*r,2*r);  
    if(jumping){  
        speedY += gravity;  
        y += speedY;  
        if(y>height) jumping=false;  
    }  
}
```

```
float r=20, x=200, y =400-r;  
float speedY = 0, gravity=0.2;  
boolean jumping = true;  
void setup(){  
    size(400,400);  
}  
void draw(){  
    background(0);  
    ellipse(x,y,2*r,2*r);  
    if(jumping){  
        speedY += gravity;  
        y += speedY;  
        if(y>height) jumping=false;  
    }  
}  
void keyPressed(){  
    if(key == ' '){  
        jumping = true;  
        speedY = -6;  
    }  
}
```

Initial code – ball is thrown upwards from bottom edge

*We need to stop it once it hits the ground again*

Now ball stops at bottom edge

*We need to add user interaction (jump only when spacebar is pressed)*

Done!

**Note:** this code is a bit different from previous slide – it allows for multi-jumping (see condition in keyPressed)

# **Aside: Android Mode**

---

- If you are interested in converting your animation/game to an Android app, do the following:
  - 1) Save your sketch (give it a meaningful name as this will be your app name).
  - 2) Switch to Android mode from your PDE (top-right corner)
  - 3) If asked, accept installing “Android Mode” and “SDK”
  - 4) Connect your Android device using a USB cable –allow “USB Debugging” if asked on your phone.
  - 5) Run your sketch (press Run button) – after a few seconds, your sketch will appear as an app on your phone.

## Example

# Aside: iOS mode – Try it!

Processing for iOS

More Apps ▾

Blog

Press

Team

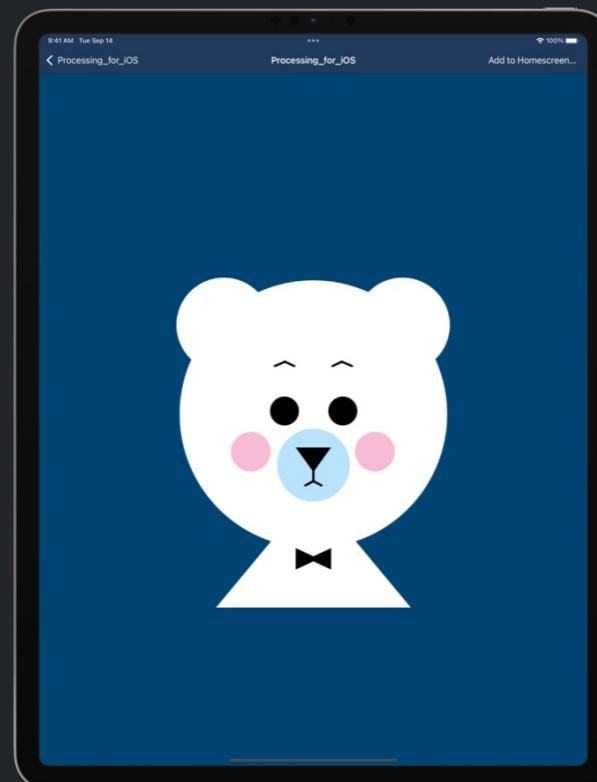
Support



## Processing & p5.js for iOS

The popular programming languages on  
iPhone and iPad.

With Processing for iOS you can write code on the go, on your iPhone, iPad or iPod Touch. And you can even export your code as apps to your home screen!



# Aside: Android Mode – try it yourself

- Here is the same code we saw before after modifying it to work on fullscreen.

```
float r,x,y;  
float speedY=0, speed, gravity;  
boolean jumping = true;  
  
void setup(){  
    fullScreen();  
    r = width/20;  
    x = width/2;  
    y = height-r;  
    speed = -height/30;  
    gravity = height/1000;  
}
```

```
void draw(){  
    background(0);  
    ellipse(x,y,2*r,2*r);  
    if(jumping){  
        speedY += gravity;  
        y += speedY;  
        if(y>height-r){  
            jumping = false;  
            y = height-r;  
        }  
    }  
}  
void mousePressed(){  
    jumping = true;  
    speedY = speed;  
}
```

## ***Aside: Android Mode – Choosing App ICON***

If you want to change the app icon, design/download an icon of your choice and put it under the sketch folder (not the data folder). Your icon's name should be **icon-48.png**.

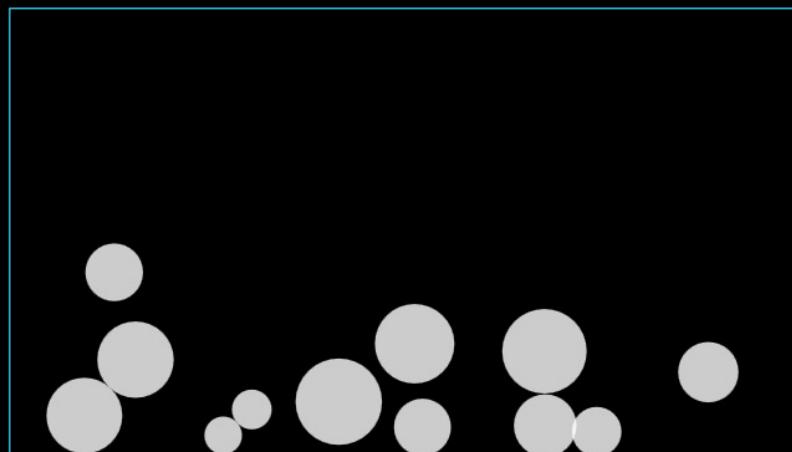
You can generate other sizes and save them in the same folder to be used if your phone launcher needs to use other icon sizes. Your icon files should be named:

- icon-36.png**
- icon-48.png**
- icon-72.png**
- icon-96.png**
- icon-180.png**
- icon-192.png**

You can find free icon files only e.g. [iconfinder.com](http://iconfinder.com)

# *Aside: Advanced Physics*

- We will not do advanced physics in this course, but if you wish to learn more, use Shiffman's book: *The Nature of Code*.
- (*Optional*) Here is another advanced example of several bouncing balls which also collide.
  - <https://processing.org/examples/bouncybubbles.html>



# Objectives

- Define: Boolean, condition
- List and use the comparison operators.
- Construct and evaluate Boolean expressions using AND, OR, NOT.
- Write and use decisions using the if/else and switch statements
- Use conditionals with common animation themes such as bouncing attributes, complex paths, buttons, etc.
- Use conditionals to react to user response to our questions (e.g. Yes/No questions)
- Explain the dangling else problem.
- Explain how to compare Strings/Objects and how why cannot use == with them.



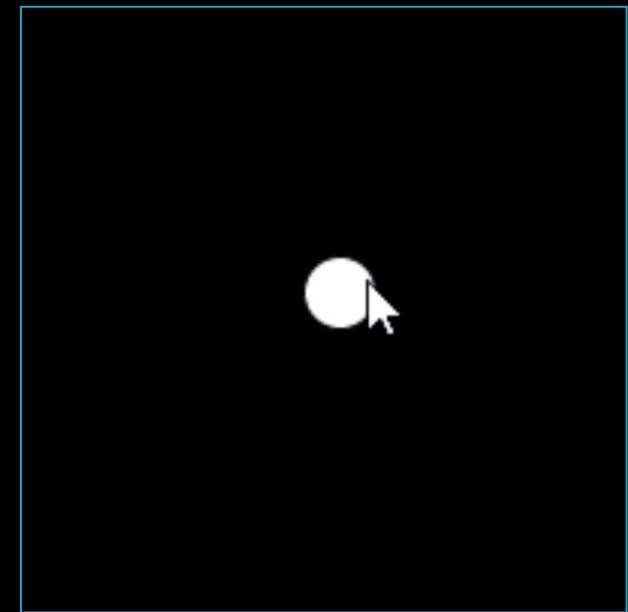
# *Revisiting map() and constrain()*

---

- See week 6 slides Useful Functions section.

# Using the map() function

- Using the map() function, draw a circle that follows the mouse cursor “*to some extent*” as in the shown animation.
- To do this, you need to map:
  - mouseX from range (0 to window width) to range (25% to 75% of the width), and
  - mouseY from range (0 to window height) to the range (25% to 75% of the height).



# Using the constrain() function

- Repeat ***the last exercise*** but with the ball exactly following the mouse cursor as long as the mouse is within the dotted box (below). The ball cannot follow the mouse outside the box.
- To do this, replace the map() function with a constrain() function.
- (***you don't have to draw the dotted box***)

