

## As you arrive:

1. Start up your computer and plug it in
2. **Log into Angel** and go to CSSE 120
3. Do the **Attendance Widget** – the PIN is on the board
4. Go to the course **Schedule Page**
5. Open the **Slides** for today if you wish
6. Check out today's project:

*Plus in-class time working on these concepts AND practicing previous concepts, continued as homework.*

## Exam 1 preview


- Date and time of exam
- Exam location
- Format of exam (paper part + programming part)
- Possible topics on exam

## Decision Structures

- Simple decisions
- Computing with Booleans
- If-else statements (plus nesting)
- Multi-way decisions

# Exam 1 information

- **When? Where?: See schedule page**
  - Please get in the habit of checking the **schedule page regularly**.
  - Time management is a problem solving process also
- **Format:**
  - **Paper part:** Zelle book, 1 double-sided sheet of notes, *closed computer*
  - **Programming part:** Zelle book, any written notes, and your computer
    - **Any resources you can reach from Angel or the course web site by clicking only!**

- 
- ▣ Amnesty.
  - ▣ How to work efficiently.
  - ▣ Has files example per email.
  - ▣ Has how to do input.
  - ▣ Has debugging skills example per email.
  - ▣ Short and long version of Topics for exam.

# Possible Topics for Exam 1

- ❑ Zelle chapters 1-7, 8.4
- ❑ algorithm
- ❑ Comment
- ❑ variable, assignment
- ❑ identifier, expression
- ❑ Loop
  - ❑ definite (for)
  - ❑ counted (range function)
- ❑ phases of software development
- ❑ print, input
- ❑ import, math functions
- ❑ int, float conversion
- ❑ strings (basic operations)
- ❑ character codes (chr, ord)
- ❑ lists (concatenation, slices)
  - ❑ list methods
  - ❑ indexing
- ❑ reading, writing files
- ❑ formatted output
  - ❑ reading
  - ❑ Writing
- ❑ using objects, graphics
- ❑ method vs. function

# More topics for exam 1

- Using zellegraphics library
- Functions
  - ▣ defining
  - ▣ calling (invoking)
  - ▣ parameter-passing
  - ▣ mutable parameters
  - ▣ optional parameters
  - ▣ return values
- decision structures
  - ▣ if, elif, else
  - ▣ computing with Booleans

# Control structures

- Normally, statements in a program **execute in order**, one after the other
- Sometimes **we want to alter** the sequential flow of a program
  - What examples have we seen of this?

- **Control structures**
  - Statements that **alter the flow** of execution of a program
  - **Examples include**
    - **Loops**
      - Repeat execution of a block of code
    - **Function call**
      - Causes execution to jump around in the code
    - **Other** →

# Decision, Decisions, Decisions

**Decision structures** are **control structures** that allow programs to **choose** between different sequences of instructions.

Expressed as an **if** statement

```
if <condition>:  
    <body>
```

If the <condition> is **True**, execute the <body>

**True** and **False** are  
Boolean constants

# Simple conditions

**<condition>** → **<expr>** **<relop>** **<expr>**

*Relational operator*

*Some Relational operators*

Math	<	≤	=	≥	>	≠
Python	<	<=	==	>=	>	!=



# Checkout today's project: 10-DecisionStructures

***Troubles getting  
today's project?***

***If so:      →***

***Are you in the Pydev perspective? If not:***

- **Window ~ Open Perspective ~ Other**  
then **Pydev**

***Messed up views? If so:***

- **Window ~ Reset Perspective**

***No SVN repositories view (tab)? If it is not there:***

- **Window ~ Show View ~ Other**  
then **SVN ~ SVN Repositories**

***In your SVN repositories view (tab), expand your repository (the top-level item) if not already expanded.***

- If no repository, perhaps you are in the wrong Workspace. Get help as needed.

***Right-click on today's project, then select Checkout.***  
***Press OK as needed.***

The project shows up in the

**Pydev Package Explorer**

to the right. Expand and browse the modules under **src** as desired.

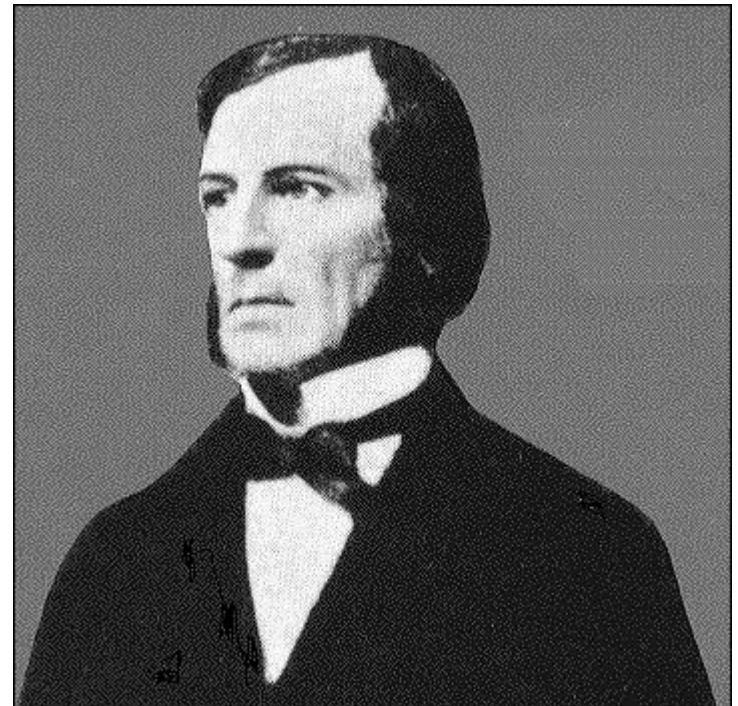
# Class Exercise

- In module `01-grade.py`, define a function `grade(score)`
  - ▣ where `score` is an exam score
  - ▣ and result is "perfect", "passing", or "failing" based on the score

# Comparisons--Boolean expressions

- **Conditions** are *Boolean expressions*
  - They evaluate to **True** or **False** → **Boolean constants**
- Try in PyDev console:

```
>>> 3 < 4
>>> 42 > 7**2
>>> "ni" == "Ni"
>>> "A" < "B"
>>> "a" < "B"
```



George Boole

# Boolean Variables and Operations

- **Boolean** constants: **True**, **False**
- **Relational operators** (<, etc.) produce **Boolean** values.

```
>>> 4 < 5
True
>>> 6 != 6
False
```

- Other **Boolean** operators: **and**, **or**, **not**

$P$	$Q$	$P \text{ and } Q$
T	T	T
T	F	F
F	T	F
F	F	F

$P$	$Q$	$P \text{ or } Q$
T	T	T
T	F	T
F	T	T
F	F	F

$P$	$\text{not } P$
T	F
F	T

Truth tables

# Having It Both Ways: if-else

## Syntax:

```
if <Condition>:  
    <statementsForTrue>  
else:  
    <statementForFalse>
```

## Semantics:

If <condition> is **True**,  
execute these  
statements

If <condition> is **False**,  
execute these  
statements

# A Mess of Nests

---

- Can we modify the **grade** function to return letter grades—A, B, C, D, and F?

# Multi-way Decisions

## □ Syntax:

if <condition1>:

<case 1 statements>

reach here if  
condition1 is false

elif <condition2>:

<case 2 statements>

reach here if  
condition1 is false  
AND condition2 is true

elif <condition 3>:

<case 3 statements>

reach here if BOTH  
condition1 AND  
condition2 are false

...

else:

<default statements>

# Cleaning the Bird Cage

- Advantages of **if-elif-else** vs. **nesting**
  - Number of cases is clear
  - Each parallel case is at same level in code
  - Less error-prone
- Fix **grade** function to use **if-elif-else** statement instead of **nesting**



# Individual Exercise on Using if-else

- Finish the quiz first. Turn it in.
- Then open **02-countPassFail.py**
- Define (in that file) a function **countPassFail(scores)** that
  - ▣ takes a list of exam scores
  - ▣ *returns* two values:
    - the count of passing scores in the list (those at least 60), and
    - the count of failing scores in the list
- Examples:
  - ▣ **print(countPassFail([57, 100, 34, 87, 74]))** prints (3,2)
  - ▣ **print(countPassFail([59]))** prints (0,1)
  - ▣ **print(countPassFail([]))** prints (0,0)
- Commit your project to your repository.

# Begin working on your homework

- A version of star that uses conditionals
- Follow the homework 10 instructions in this order:
  - circleOfCircles
  - Star
- Use the appropriate PyDev modules in the **10-DecisionStructures** project to solve these exercises
- Commit your solutions to your repository