DECISION STRUCTURES, COMPUTING WITH BOOLEANS

CSSE 120 – Rose-Hulman Institute of Technology

Exam 1

- When? Where?: See schedule page
  - Please get in the habit of checking the schedule page regularly. Time management is a problem solving process too!
- 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 by clicking only.

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, raw_input
- import, math functions
- int, float, long, conversion
- strings (basic operations)
- character codes (chr, ord)
- lists (concatenation, slices)
  - list methods
  - indexing
- reading, writing files
- formatted output using %
- using objects, graphics
- method *vs.* function
- event-driven program

More topics for exam 1

- functions
  - defining
  - calling (invoking)
  - parameter-passing
  - mutable parameters
  - optional parameters
  - return values
- decision structures
  - if, elif, else
  - computing with Booleans

Decision, Decisions

- 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?
- Statements that alter the flow are called

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

Simple Decisions

- The **if** statement
  - if <condition>:
    <body>
  - Semantics:
    "if the condition is True, run the body, otherwise skip it"
- Simple conditions
  - <expr> <relop> <expr>
  - Some relational operators:

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

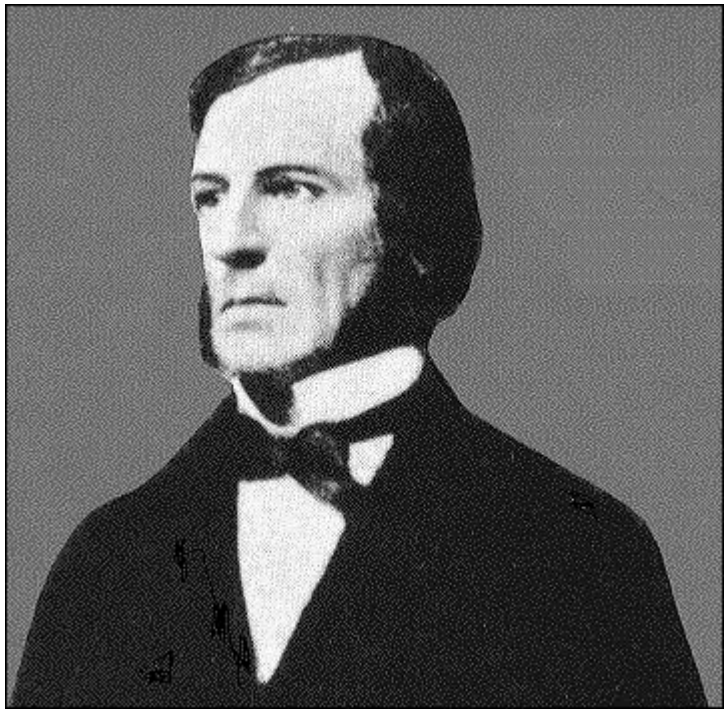
Class Exercise

- Checkout **Session10** project from your SVN repository
- In module `grade.py`, define a function **grade(score)**
  - where score is an exam score
  - and result is "perfect", "passing", or "failing" based on the score

More on Comparisons

- Conditions are *Boolean expressions*
  - They evaluate to **True** or **False**
- Try in IDLE:

```
>>> 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 and Q
T	T	T
T	F	F
F	T	F
F	F	F

P	Q	P or Q
T	T	T
T	F	T
F	T	T
F	F	F

P	not P
T	F
F	T

Having It Both Ways: if-else

- Syntax:

  if <condition>:

  <statementsForTrue>

  else:

  <statementsForFalse>

- Semantics:

  "If the condition is true, execute the statements for true, otherwise execute the statements for false"

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>
elif <condition2>:
        <case 2 statements>
elif <condition 3>:
        <case 3 statements>
…
else:
        <default statements>
```

reach here if condition1 is false

reach here if condition1 is false AND condition2 is true

reach here if BOTH condition1 AND condition2 are false

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 **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 pizza, polygon, and star that use conditionals
- Follow the homework 10 instructions in this order:
  - circleOfCircles
  - Pizza
  - Polygon
  - Star
- Use the appropriate PyDev modules in the **Session10** project to solve these problems
- Commit your solutions to your repository