Functions, Decision Structures

Rose-Hulman Institute of Technology

Computer Science and Software Engineering

Check out 10-DecisionStructures from SVN

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Exam 1

- Friday in class
  - Paper part: Zelle book, one double-sided sheet of notes, closed computer
  - Programming part: Zelle book, any written notes, and your computer
    - Resources you can reach from Angel by clicking

Possible Topics for Exam 1

- Zelle chapters 1-7
- algorithm
- comment
- variable, assignment
- identifier, expression
- loop
  - definite (for)
  - counted (range function)
- phases of software development
- print, 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 format()
- using objects, graphics
- event-driven programming

Grading Status

- Everything through HW7 is graded
- See ANGEL for grades
- See SVN for comments…

Viewing Grader Comments

- Open project in Eclipse
- Right-click project and choose **Team → Update to HEAD**
  - Downloads grader's comments, merges
- Use **Tasks** view to look for CONSIDER comments

Try this with HW2

Function Review

- Functions can take multiple parameters
  - **def distance (p1, p2): # p1, p2 are points**
    xdist = abs(p1.getX()- p2.getX())
    ydist = abs(p1.getY()- p2.getY())
    return math.sqrt(xdist*xdist + ydist*ydist)
- Invoke a function; must supply actual parameters:
  - **d = distance(Point(-1,2), Point(2,6))**
- Functions can return values

Passing parameters in Python

- What type of information do formal parameters receive?
- If we assign new values to formal parameters, does this affect the actual parameters?
- Consider this version of square:
  - **def squareNext(x):**
    x = x + 1
    return x * x

See mutatorExample.py

Q2-4

Passing a mutable parameter

- Function can change contents of a mutable parameter
- What does this print? What actually gets passed to the function?

```
def addOneToAll(listOfNums):
    '''Adds one to each item in
    the given list.'''
    for i in range(len
(listOfNums)):
        listOfNums[i] += 1
    listOfNums = [0,1,2]


myList = [1,3,5,7]
addOneToAll(myList)
print("myList is ", myList)
```

See mutatorExample.py

Optional parameters

- Function definition can make some parameters *optional*

```
>>> int("37")
37
>>> int("37", 10)
37
>>> int("37", 8) # specify base 8
31
```

Optional parameters

- We can declare a parameter to be optional by supplying a default value.

```
>>> def printDate(month, day, year=2007):
        print month, str(day)+",", year


>>> printDate("January", 4, 2006)
January 4, 2006
>>> printDate("January", 4)
January 4, 2007
```

Multiple optional parameters

- Can use *named* actual arguments:

```
def printDate(month="Jan", day="4", year="2011"):
    print("{0} {1}, {2}".format(month, day, year))

printDate()
printDate(26)
printDate(day=26)
```

See mutatorExample.py

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Returning Multiple Values

- A function can return multiple values
  - **def powers(n):**
    return n2, n**3, n**4**
- What's the type of the value returned by this call?
  powers(4)
- Assign returned values individually, or to a tuple:
  thePowers = powers(5)
  p2, p3, p4 = powers(5)

Control Freaks

- Typical: statements execute in order
- Sometimes we want other orders
  - What examples have we seen of this?
- Statements that alter the flow are called *control structures*

Decision, Decisions

- *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>**
  - 6 * 7 >= 42

Math	Python
<	<
≤	<=
>	>
≥	>=

Math	Python
=	==
≠	!=

Q5

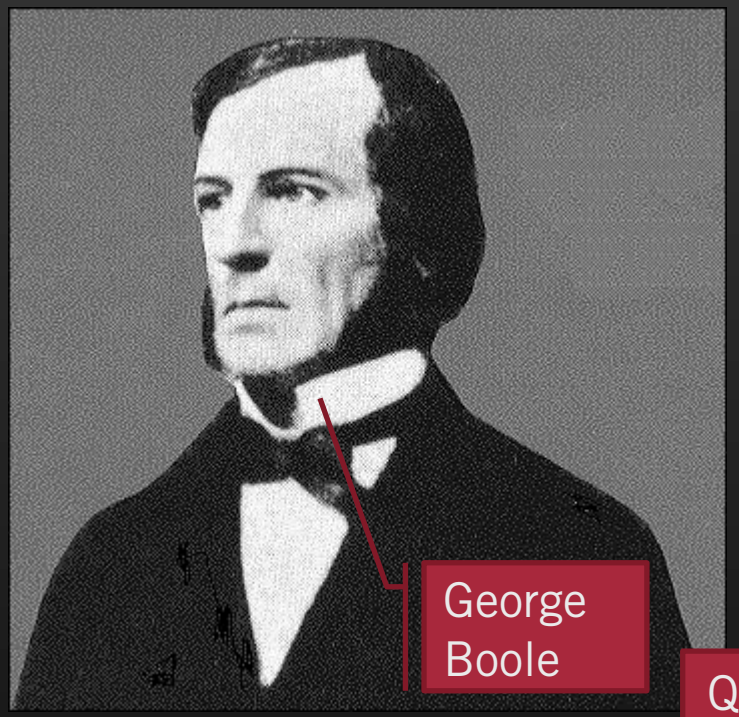
Exercise

- In module **grade.py**, define a function **grade(score)**
  - where score is from 0 to 100
  - and result is "perfect", "passing", or "failing" based on the score

More on Comparisons

- Conditions are Boolean expressions
  - Evaluate to **True** or **False**
- Try these:

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



Q6

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
- Implement **gradeFixed** to use **if-elif-else** statement instead of nesting

Q9

Wrap up the quiz before starting homework

Finish the quiz

Q10

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

Complete the TODOs in countPassFail.py

Individual Exercise on Decisions

If you finish this, continue with rest of HW10

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY