


FUNCTIONS THAT RETURN RESULTS, DECISION STRUCTURES

Questions on concepts on Exam 1

- Last session's slides contained a list of concepts that will be on exam 1
- What questions on these concepts have come up so far as you have started studying?
- Any other questions?

Functions Can Return Values

- We've **written** functions that just do things
 - ▣ `printFactorial(5)`
 - ▣ `printDistance(p1,p2)`
- We've **used** functions that *return* values
 - ▣ `abs(-1)`
 - ▣ `fn_root_1 = math.sqrt(b*b - 4*a*c)`
- Define a function that returns a value
 - ▣ `"""Returns the square of x """`
`def square(x):`
 `return x * x`


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
```

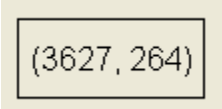
```
    return x * x
```

Multiple Value Return

- 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:
 powers(4)

Pair Programming: Three Squares

- Download `threeSquares.py` from Angel
- Run the program to be sure it works
- Add a function, `stats` that takes a Rectangle, `r`, as a parameter and returns the area of `r`
- Change the program to display the area of each rectangle inside the shape
- Finally, change `stats` to return the area and perimeter (see figure at right)
- Upload to `threeSquares` dropbox on Angel



(3627, 264)

Example
Display

Decision, Decisions

- Sometimes we have 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	<	<=	==	>=	>	!=

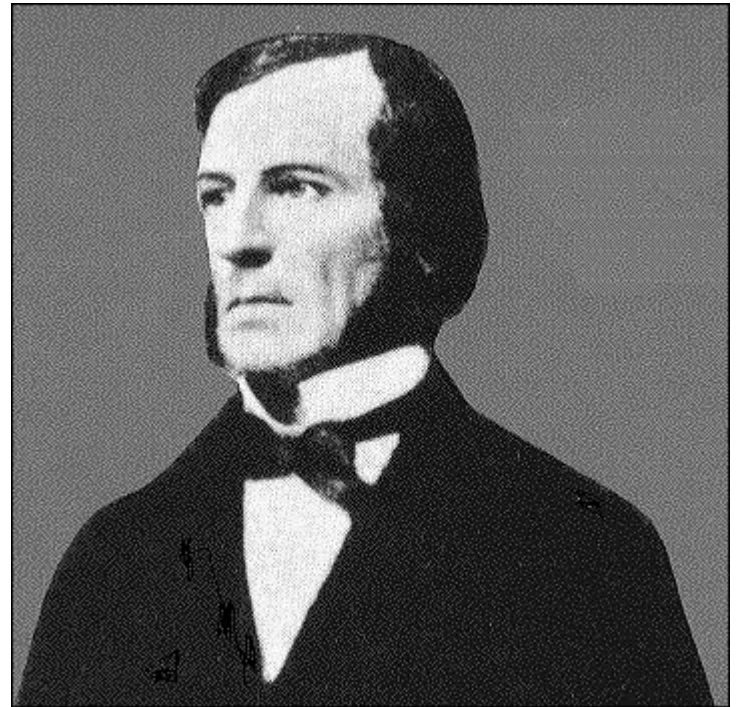
Class Exercise

- 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

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>

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

- Define a function **countFailPass(scores)** that
 - takes a list of exam scores
 - returns two values:
 - the count of failing scores in the list (those less than 60), and
 - the count of passing scores in the list
- Examples:
 - **countFailPass([57, 100, 34, 87, 74])** returns **(2,3)**
 - **countFailPass([59])** returns **(1,0)**
 - **countFailPass([])** returns **(0,0)**
- Upload to **countFailPass** dropbox on Angel