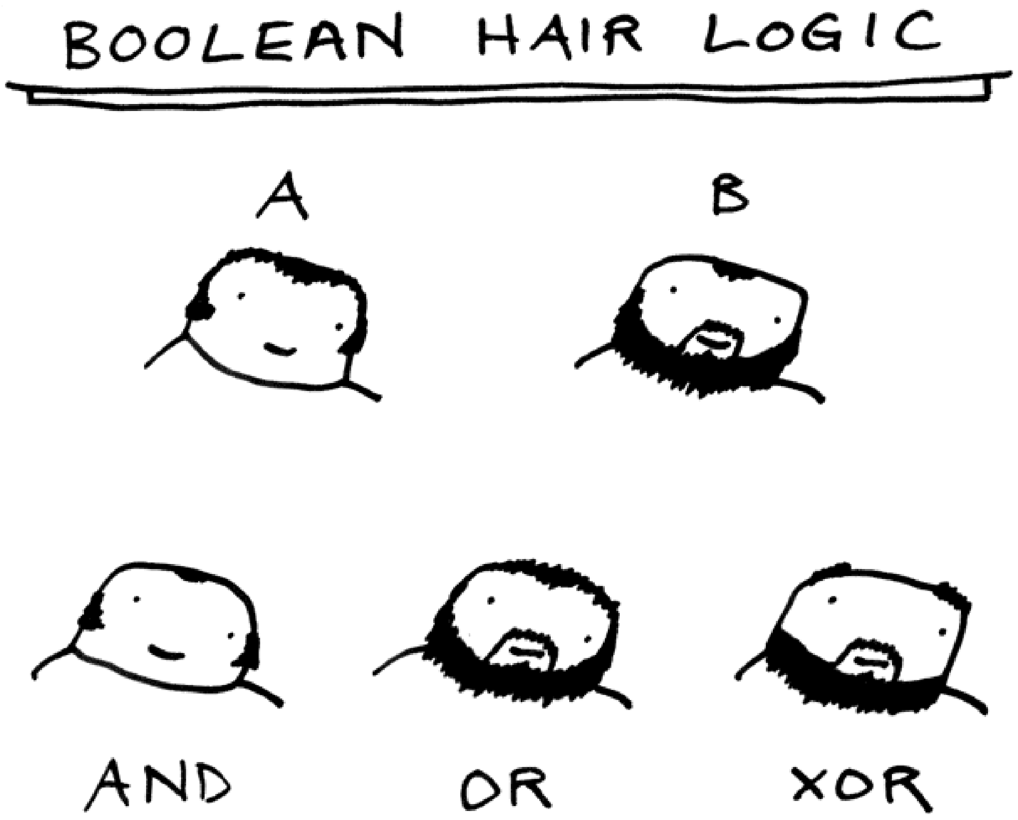
CSc 110, Autumn 2016

Lecture 10: Advanced `if/else`; Cumulative sum

Adapted from slides by Marty Stepp and Stuart Reges



Factoring `if/else` code

- **factoring**: Extracting common/redundant code.
  - Can reduce or eliminate redundancy from `if/else` code.

- Example:

```
if (a == 1):
    print(a)
    x = 3
    b = b + x
elif (a == 2):
    print(a)
    x = 6
    y = y + 10
    b = b + x
else:          # a == 3
    print(a)
    x = 9
    b = b + x
```

→

```
print(a)
x = 3 * a
if (a == 2):
    y = y + 10
b = b + x
```

Relational expressions

- `if` statements use logical tests.

```
if (i <= 10) { ...
```

  - These are `boolean` expressions.

- Tests use *relational operators*:

Operator	Meaning	Example	Value
`==`	equals	`1 + 1 == 2`	`true`
`!=` `<>`	does not equal	`3.2 != 2.5` `3.2 <> 2.5`	`true`
`<`	less than	`10 < 5`	`false`
`>`	greater than	`10 > 5`	`true`
`<=`	less than or equal to	`126 <= 100`	`false`
`>=`	greater than or equal to	`5.0 >= 5.0`	`true`

Logical operators

- Tests can be combined using *logical operators*:

Operator	Description	Example	Result
`and`	and	`(2 == 3) and (-1 < 5)`	`False`
`or`	or	`(2 == 3) or (-1 < 5)`	`True`
`not`	not	`not (2 == 3)`	`True`

- "Truth tables" for each, used with logical values *p* and *q*:

P	q	p and q	p or q
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

p	not p
True	False
False	True

Evaluating logical expressions

- Relational operators have lower precedence than math; logical operators have lower precedence than relational operators

```
5 * 7 >= 3 + 5 * (7 - 1) and 7 <= 11
5 * 7 >= 3 + 5 * 6 and 7 <= 11
35    >= 3 + 30 and 7 <= 11
35    >= 33 and 7 <= 11
True and True
True
```

- Relational operators cannot be "chained" as in algebra

```
2 <= x <= 10
True   <= 10
```

(assume that `x` is `15`)

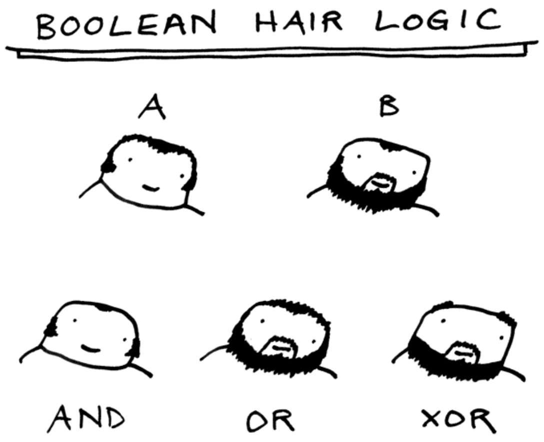
  - Instead, combine multiple tests with `and` or `or`

```
2 <= x and x <= 10
True   and False
False
```

Logical questions

- What is the result of each of the following expressions?

```
x = 42
y = 17
z = 25
```



- `y < x and y <= z`
- `x % 2 == y % 2 or x % 2 == z % 2`
- `x <= y + z and x >= y + z`
- `not(x < y and x < z)`
- `(x + y) % 2 == 0 or not((z - y) % 2 == 0)`

  - Answers: `True`, `False`, `True`, `True`, `False`

Cumulative algorithms

Adding many numbers

- How would you find the sum of all integers from 1-1000?

```
# This may require a lot of typing
sum = 1 + 2 + 3 + 4 + ...
print("The sum is " + str(sum))
```

- What if we want the sum from 1 - 1,000,000?
  Or the sum up to any maximum?
  - How can we generalize the above code?

Cumulative sum loop

```
sum = 0
for i in range(1, 1001):
    sum = sum + i

print("The sum is " + str(sum))
```

- **cumulative sum**: A variable that keeps a sum in progress and is updated repeatedly until summing is finished.
    - The `sum` in the above code is an attempt at a cumulative sum.
    - Cumulative sum variables must be declared *outside* the loops that update them, so that they will still exist after the loop.

Cumulative product

- This cumulative idea can be used with other operators:

```
product = 1
for i in range(1, 21):
    product = product * 2

print("2 ^ 20 = " + str(product))
```

  - How would we make the base and exponent adjustable?

`input` and cumulative sum

- We can do a cumulative sum of user input:

```
sum = 0;
for i in range(1, 101):
    next = int(input("Type a number: "))
    sum = sum + next
}
print("The sum is " + str(sum))
```

Cumulative sum question

- Modify the `Receipt` program from lecture 2
  - Prompt for how many people, and each person's dinner cost.
  - Use functions to structure the solution.
- Example log of execution:

```
How many people ate? 4
Person #1: How much did your dinner cost? 20.00
Person #2: How much did your dinner cost? 15
Person #3: How much did your dinner cost? 30.0
Person #4: How much did your dinner cost? 10.00

Subtotal: $75.0
Tax: $6.0
Tip: $11.25
Total: $92.25
```

Cumulative sum answer

```
# This program enhances our Receipt program using a cumulative sum.
def main():
    subtotal = meals()
    results(subtotal)

# Prompts for number of people and returns total meal subtotal.
def meals():
    people = float(input("How many people ate? "))
    subtotal = 0.0;            # cumulative sum

    for i in range(1, people + 1):
        person_cost = float(input("Person #" + str(i) +
                       ": How much did your dinner cost? "))
        subtotal = subtotal + person_cost;  # add to sum
    return subtotal
...
```

Cumulative answer, cont'd.

```
# Calculates total owed, assuming 8% tax and 15% tip
def results(subtotal):
    tax = subtotal * .08
    tip = subtotal * .15
    total = subtotal + tax + tip

    print("Subtotal: $" + str(subtotal))
    print("Tax: $" + str(tax))
    print("Tip: $" + str(tip))
    print("Total: $" + str(total))
```

if/else, return question

- Write a function count_factors that returns the number of factors of an integer.
  - count_factors(24) returns 8 because 1, 2, 3, 4, 6, 8, 12, and 24 are factors of 24.

- Solution:

```
# Returns how many factors the given number has.
def count_factors(number):
    count = 0
    for i in range(1, number + 1):
        if (number % i == 0):
            count += 1      # i is a factor of number
    return count
```