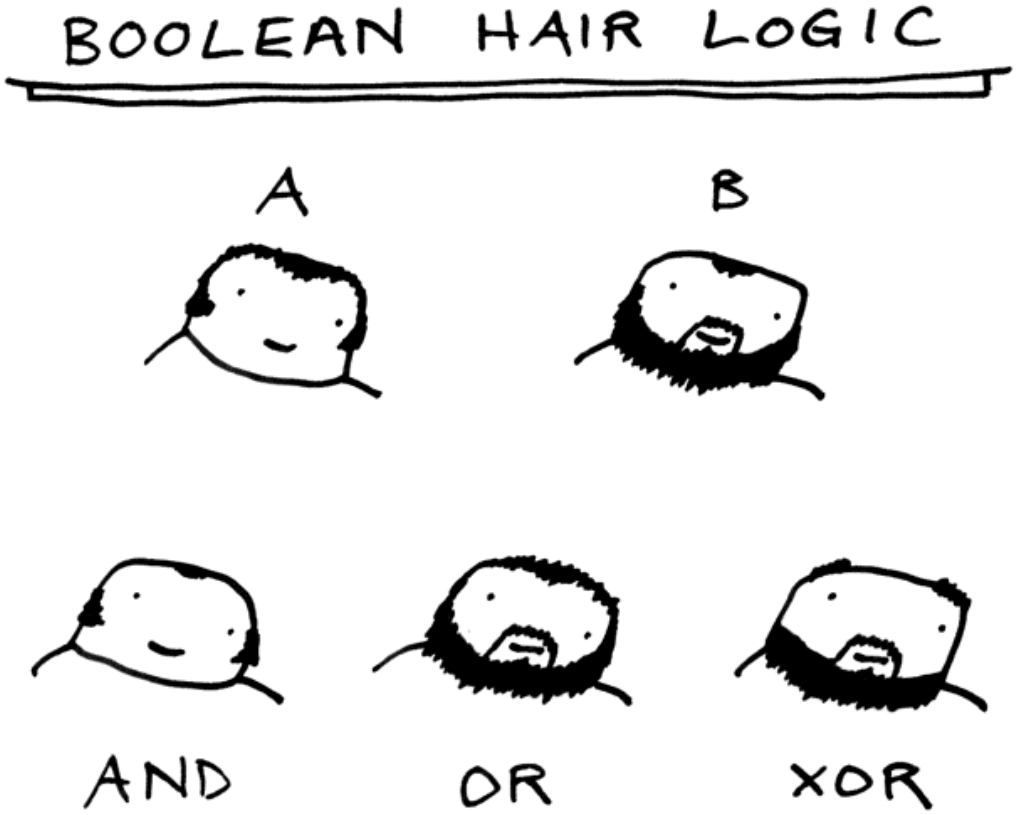
CSc 110, Autumn 2016

Lecture 12: Advanced `if/else`; Cumulative sum

Adapted from slides by Marty Stepp and Stuart Reges



Nested `if/else` question

Write a program that produces output like the following:

```
This program reads data for two
people and computes their basal
metabolic rate and burn rate.

Enter next person's information:
height (in inches)? 73.5
weight (in pounds)? 230
age (in years)? 35
gender (male or female)? male

Enter next person's information:
height (in inches)? 71
weight (in pounds)? 220.5
age (in years)? 20
gender (male or female)? female

Person #1 basal metabolic rate = 2042.3
high resting burn rate
Person #2 basal metabolic rate = 1868.4
moderate resting burn rate
```

- Basal Metabolic Rate Formula:

male BMR = 4.54545 x (weight in lb) + 15.875 x (height in inches) - 5 x (age in years) + 5

female BMR = 4.54545 x (weight in lb) + 15.875 x (height in inches) - 5 x (age in years) - 161

BMR	Burn Level
below 12000	low
1200 to 2000	moderate
above 2000	high

Nested `if/else` answer

```
# This program finds the basal metabolic rate (BMR) for two
# individuals. This variation includes several functions
# other than main.

# introduces the program to the user
def give_intro():
    print("This program reads data for two")
    print("people and computes their basal")
    print("metabolic rate and burn rate.")
    print()

# prompts for one person's statistics, returning the BMI
def get_bmr(person):
    print("Enter person", person, "information:")
    height = float(input("height (in inches)? "))
    weight = float(input("weight (in pounds)? "))
    age = float(input("age (in years)? "))
    gender = input("gender (male or female)? ")
    bmr = bmr_for(height, weight, age, gender)
    print()
    return bmr
...
```

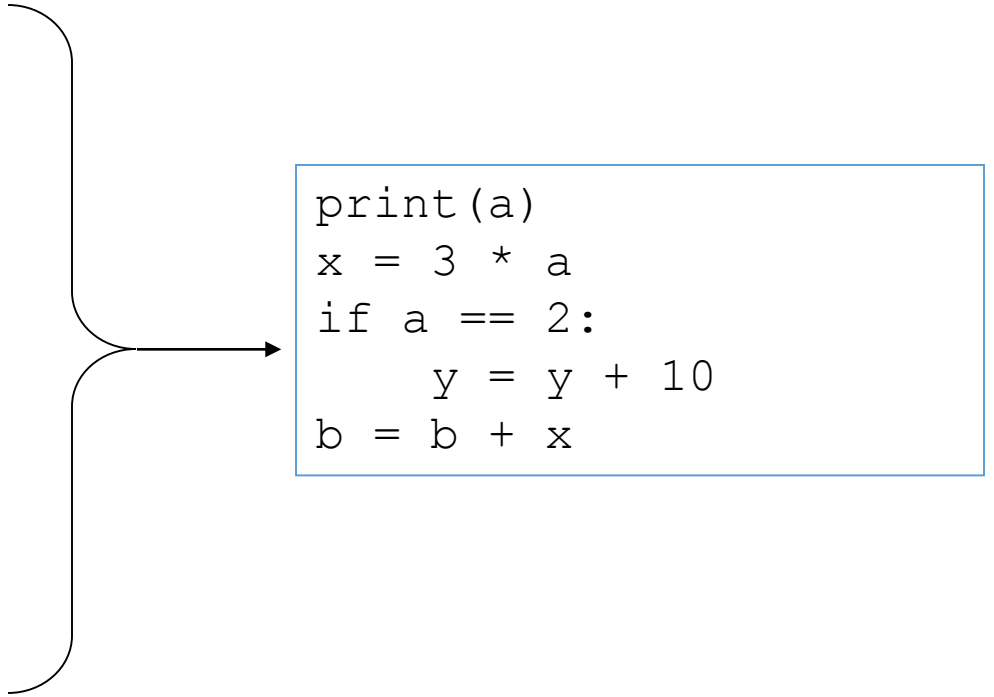
Nested `if/else`, cont'd.

```
# this function contains the basal metabolic rate formula for
# converting the given height (in inches), weight
# (in pounds), age (in years) and gender (male or female) into a BMR
def bmr_for(height, weight, age, gender):
    bmr = 4.54545 * weight + 15.875 * height - 5 * age
    if gender.lower() == "male":
        bmr += 5
    else:
        bmr -= 161
    return bmr

# reports the overall bmr values and status
def report_results(bmr1, bmr2):
    print("Person #1 basal metabolic rate =", round(bmr1, 1))
    report_status(bmr1)
    print("Person #2 basal metabolic rate =", round(bmr2, 1))
    report_status(bmr2)

# reports the burn rate for the given BMR value
def report_status(bmr):
    if bmr < 1200:
        print("low resting burn rate");
    elif bmr <= 2000:
        print("moderate resting burn rate")
    else: # bmr1 > 2000
        print("high resting burn rate")

def main():
    give_intro()
    bmr1 = get_bmr(1)
    bmr2 = get_bmr(2)
    print(bmr1, bmr2)
    report_results(bmr1, bmr2)

main()
```

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
```

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", 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", 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 =", 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", 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
```