Welcome back to CS 5 Green!

Learning Goals
• Explain when conditionals are useful
• Implement conditionals
• Explain when loops are useful
• Implement for loops

Reminder
• Homework 0 due on Tuesday at 11:59 PM
The Secret of Happiness is…
(in programming)

• Don’t memorize!
• Look at examples of similar problems
• Experiment
• Syntax that looks weird now will become second nature soon
A friendly note from the autograder…

• Please check your function names
  – They need to be spelled as specified (case-sensitive). Otherwise, the autograder will not see them (e.g. findSpam is different from findsspam).

• If the autograder reports problems, please try to fix them before submission
  – Come to office hours and tutoring hours
Looking up stuff on the web…

Looking up Python documentation (e.g., Googling “python documentation”) is fine!
Types of data in Python

>>> best_food = "spam" ← String (single or double quotes work)

>>> ok_food = 'chocolate'

>>> good_num = 42 ← Integer (int)

>>> pi = 3.1415926 ← “Floating point” number (float)

>>> special = [2.718, 3.141, 42] ← List (list)

>>> 'chocolate' = ok_food

>>> 42 = good_num

Why does Python barf?
Booleans

>>> 3 == 1+2
True

>>> 1+2 == 3
True

>>> 42 == "spam"
False

>>> [1, 2, 3] == [1, 2, 3]
True

>>> [1, 2, 3] == [3, 2, 1]
False

>>> 42 > 5
True

>>> 42 != 5
True

>>> not 42 == 5
True

Relational Operators

== is equal to
!= is not equal to
> is greater than
< is less than
>= is greater than or equal to
<= is less than or equal to

George Boole
1815-1864
temp = 80
temp_is_low = (temp < 65)
forecast = 'rain'
is_raining = (forecast == 'sunny')
result = (temp_is_low != is_raining)
print result

Questions
1. What does this code do?

2. What could be changed to improve code style?
def special(x):
    """This function demonstrates the use of if and else"""
    if x == 42:
        return "Nice!"
    else:
        return "Yuck!"

>>> special(42)
"Nice!"

>>> special(43)
"Yuck!"
The Collatz Conjecture

Paul Erdos

def collatz(n):
    
    """Returns n/2 if n is even and returns 3n+1 otherwise"""

    if n % 2 == 0:   # if n is even...
        return n/2
    else:
        return 3*n + 1

the docstring

a comment
def special(x):
    """This function demonstrates the use of if, elif, and else"""
    if x < 42:
        return "Silly little number!"
    elif x == 42:
        return "Secret to all happiness!"
    else:
        return "Wow, that's big!"
if, elif, else...

def special(x):
    """This function demonstrates the use of if, elif, and else"""
    if x < 42:
        if x % 2 == 0:
            return "Silly small even number"
        else:
            return "Silly small odd number"
    elif x == 42:
        return "Secret to all happiness!"
    else:
        if x % 2 == 0:
            return "Big & even"
        else:
            return "Big & odd"

This entire function is odd!
def schedule(hour):
    """Describes your day in military time"""
    if hour >= 2 and hour <= 9:
        return "Sleep"
    elif hour <= 17:
        return "In class"
    elif hour <= 20:
        return "Hang out"
    else:
        return "Do CS homework"

You really otta get more sleep!

Boolean Operators

and : True if both are True
or  : True if at least one is True
not : True if argument is False
Getting things in order...

def schedule(hour):
    """Describes your day in military time""
    if hour >= 2 and hour <= 9:
        return "Sleep"
    elif hour <= 17:
        return "In class"
    elif hour <= 20:
        return "Hang out"
    else:
        return "Do CS homework"

def schedule(hour):
    """Describes your day in military time""
    if hour >= 2 and hour <= 9:
        return "Sleep"
    elif hour <= 20:
        return "Hang out"
    elif hour <= 17:
        return "In class"
    else:
        return "Do CS homework"

What's wrong here?!
A Cheesy Example…

def special(x):
    if x == 42:
        dinner = "spam!"
    else:
        dinner = "mac & cheese"
    return dinner

Alternatively??

Option 1

def special2(x):
    if x == 42:
        dinner = "spam"
    dinner = "mac & cheese"
    return dinner

Option 2

def special3(x):
    if x == 42:
        dinner = "spam"
    return dinner
dinner = "mac & cheese"
    return dinner

Worksheet: Which of these two alternatives does the same thing as the special function above? Why?
1 min break!
Python Gets Loopy!

The Python, Busch Gardens Florida
Python Gets Loopy!

`loopy` is a function, but it does not expect any input!

We choose the name of a variable…

```python
def loopy():
    sum = 0
    for my_number in [20, 17, 5]:
        sum = sum + my_number
    return sum
```

... and we provide a list

```python
>>> loopy()
42
```
Carl Friedrich Gauss
1777-1855
If Gauss had Python…

Goal: Add up \( 1 + 2 + \ldots + n \)

def gauss(n):
    """Returns 1 + 2 + \ldots + n""
    sum = 0
    for num in list(range(1, n+1)):
        sum = sum + num
    return sum
The Anatomy of a for loop

```python
for my_variable in list:
    Do all the stuff that is indented beneath the for loop

Stuff at this level of indentation is done afterwards!
```
Factorial...

Goal: Take an integer n as input and return n!

def factorial(n):
    """Returns the factorial of n"""

Try this (and the next two problems) in your notes.
Approximating e…

Goal: Take a value \( n \) as input and return 
\[ 1 + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{n!} \]

```python
def e(n):
    """Returns 1 + 1/1! + 1/2! + ... + 1/n!
    Assume that factorial(n) is available""
```

In your notes
def mystery(n):
    for d in range(2, n):
        if n % d == 0:
            return False
    return True

What is this function saying about its input \( n \)?
Factorial...

**Goal:** Take an integer \( n \) as input and return \( n! \)

def factorial(n):
   
   """Returns the factorial of n""
   
   product = 1
   for factor in range(1, n+1):
       product = product * factor
   return product
Goal: Take a value n as input and return 1 + 1/1! + 1/2! + … + 1/n!

def e(n):
    """Returns 1 + 1/1! + 1/2! + ... 1/n!
    Assume that factorial(n) is available""
    sum = 1  # why 1?
    for denominator in range(1, n+1):
        sum = sum + 1/factorial(denominator)
    return sum
A mystery...

```python
def mystery(n):
    for d in range(2, n):
        if n % d == 0:
            return False
    return True
```

What is this function saying about its input `n`?

**Is n prime?**
(Returns `True` if `n` is prime, `False` if `n` is not prime)

In your notes...

For example, `7 % 2` is 1 (the remainder when the integer 7 is divided by the integer 2)
A perfect worksheet problem!

\[6 = 1 + 2 + 3\]
\[28 = 1 + 2 + 4 + 7 + 14\]

6, 28, 496, 8128, 33,550,336,
Known to ancient Greeks, Nicomachus, 100 AD!, 1456!

8,589,869,056, 137,438,691,328,
Pietro Cataldi, 1588!
>>> perfect(6)
True

>>> perfect(7)
False

def perfect(n):
    """Returns True is n is perfect, False otherwise"""
    sod = 0
    for d in range(1, n):
        if n % d == 0:
            sod = sod + d

Achieving perfection!

What is this doing? A few more lines of code in your notes and we are done!
>>> perfect(6)
True

>>> perfect(7)
False

def perfect(n):
    """Returns True is n is perfect, False otherwise"""
    sod = 0
    for d in range(1, n):
        if n % d == 0:
            sod = sod + d
    if n == sod:
        return True
    else:
        return False

Achieving perfection!

What if we indented this stuff? (Check out Prof Wu’s magic indenting!)
>>> perfect(6)
 True

>>> perfect(7)
 False

def perfect(n):
    """Returns True is n is perfect, False otherwise""
    sod = 0
    for d in range(1, n):
        if n % d == 0:
            sod = sod + d
    return n == sod

Demo perfect.py
Extra Practice!
(if time allows)
Collatz Revisited

```python
def collatz(n):
    """Returns n/2 if n is even and returns 3n+1 otherwise""
    if n % 2 == 0:   # if n is even...
        return n/2
    else:
        return 3*n + 1

>>> test_num(16, 10)
True
```

If we start with 16 and apply collatz repeatedly, do we get to 1 within the first 10 repeats?

```python
def test_num(number, repeats):
    """Returns True if the number collatzes within the give number of repeats""
    for i in range(repeats):
        number = collatz(number)
```

Write the last two lines of code in your notes!
Collatz Revisited

```python
def collatz(n):
    """Returns n/2 if n is even and returns 3n+1 otherwise""
    if n % 2 == 0:   # if n is even...
        return n/2
    else:
        return 3*n + 1
```

```python
>>> test_num(16, 10)
True
```

If we start with 16 and apply collatz repeatedly, do we get to 1 within the first 10 repeats?

```python
def test_num(number, repeats):
    """Returns True if the number collatzes within the given number of repeats""
    for i in range(repeats):
        number = collatz(number)
        if number == 1: return True
    return False
```

Collatz Revisited

```python
def collatz(n):
    
    # Returns n/2 if n is even and returns 3n+1 otherwise

    if n % 2 == 0:   # if n is even...
        return n/2
    else:
        return 3*n + 1

>>> test_num(16, 10)
True
```

If we start with 16 and apply collatz repeatedly, do we get to 1 within the first 10 repeats?

```python
def test_num(number, repeats):
    
    # Returns True if the number collatzes within the given number of repeats

    for i in range(repeats):
        number = collatz(number)
        if number == 1:
            return True
        else:
            return False
```

Indentation of else OK?
From xkcd…

The Collatz Conjecture states that if you pick a number, and if it’s even divide it by two and if it’s odd multiply it by three and add one, and you repeat this procedure long enough, eventually your friends will stop calling to see if you want to hang out.
def leppard(input_string):
    """What does this do?""
    output_string = ""
    for symbol in input_string:
        if symbol == "o":
            output_string = output_string + "ooo"
        else:
            output_string = output_string + symbol
    return output_string

>>> leppard("hello")

>>> leppard("hello to you")
```
import speech

def leppard(input_string):
    """Leppard-ifies the input string and speaks it!""
    output_string = ""
    for symbol in input_string:
        if symbol == "o":
            output_string = output_string + "ooo"
        else:
            output_string = output_string + symbol
    return output_string

speech.say("hello")
speech.say(leppard("hello"))

speech.say("hello to you")
speech.say(leppard("hello to you"))
```