



CS 5 Green

Learning Goals

- Practice conditionals and `for` loops
- Use Turtle Graphics

Reading and Lab

The reading
reinforces the
lecture material.



- This week: 1.6 – 1.11, Chapter 2
- Lab this week...
 - Please check in at 1:15 PM to get credit for lab
 - stay until done with lab problems or 3:15 PM, whichever comes first ;^)
 - Practice with `for` loops and `if-elif-else`
 - Getting ready for gene finding!
 - Some fun optional bonus problems

Last time...

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

***Perfect-ing the
return statement***



More Mysteries!

```
def mystery1(n):  
    for k in range(1, n):  
        if k*k == n: return True  
    return False
```

Assume that we run
mystery1 with
positive integers
and mystery2 with
n>2 as input!



Try these on your worksheet!

```
def mystery2(n):  
    for k in range(1, n):  
        if n == 1:  
            return True  
        elif not (n % 2 == 0): # n % 2 != 0  
            return False  
    else:  
        n = n/2
```

If you have time: how could you
change these functions to print
an error message if the input is
too low?

Collatz Revisited

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

```
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
```

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

```
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 Re-Revisited

Q

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

```
>>> test_conjecture(20, 10)
```

```
False
```

```
>>> test_conjecture(20, 50)
```

```
True
```

Try all numbers from 2 to 20

Up to this many repeats
each time!

```
def test_conjecture(up_to, repeats):  
    """Determines if all numbers from 2 to up_to  
    collatz to 1 within given number of repeats"""  
    for number in range(2, up_to+1):
```

Fill in the missing parts!

Mystery

I love a good mystery!



```
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
```

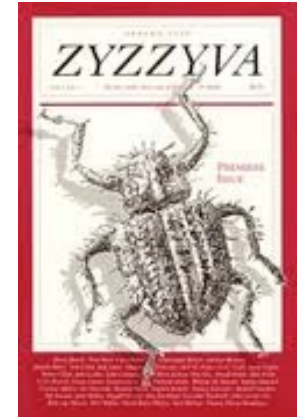
```
>>> leppard("hello")
```

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





z detector



0123456789111111

012345

```
>>> z("I like zyzzyvas!")
```

3

```
>>> z("I am opposed to the letter after y")
```

0

```
def z(input):  
    counter = 0  
    for symbol in input:  
        if symbol == 'z':  
            counter = counter+1  
    return counter
```

The “direct” method

```
def z(input):  
    counter = 0  
    for i in range(len(input)) [0,1,2,3,...15]:  
        if input[i] == 'z':  
            counter = counter + 1  
    return counter
```

The “indirect” or “index” method

Spam counter!



```
>>> spam_count("I like spam with spamspamspam!")
```

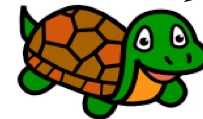
```
4
```

```
>>> spam_count("spamityspampampam!")
```

```
2
```

```
def spam_count(input):  
    counter = 0  
    for letter in input:  
        ???
```

First attempt...
The direct method





Spam counter!

```
>>> spam_count("I like spam with spamspamspam!")
```

```
4
```

```
01234567891111111111  
01234567
```

```
>>> spam_count("spamityspampampam!")
```

```
2
```

Finish this in your notes. Hint: Use slicing! (e.g., input[2:5])



```
def spam_count(input):
```

```
    counter = 0
```

```
    for i in range(len(input)):
```

Ah, the indirect/index method!



Functions that return lists

```
>>> squares(5)
[1, 4, 9, 16, 25]
```

```
def squares(n):
    output = []
    for x in range(1, n+1):
        output = output + [x*x]
    return output
```

upgrade to list-hood!

```
def squares(n):
    output = []
    for x in range(1, n+1):
        output.append(x*x)
    return output
```



Spam finder!

01234567891
0

```
>>> spam_finder("spamspamity")
```

```
[0, 4]
```

```
>>> spam_finder("ssspam!")
```

```
[2]
```

```
def spam_finder(input):
```



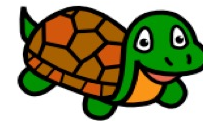
Stepping!



```
def return_codons(DNA_string):  
    codon_list = []  
    for i in range(0, len(DNA_string), 3):  
        codon_list.append(DNA_string[i:i+3])  
    return codon_list
```

```
0123456789  
>>> return_codons("AAATTTGGGC")  
["AAA", "TTT", "GGG", "C"]
```

**What colorful
codons you
have!**



Example: Do pesticides affect bumble bees?

Concern about imidacloprid crop seed treatments potentially harming bumble bees

GROUP	INSECTICIDE
4A	

Gaucha[®] 600 SC Insecticide

For uses in pest management, suppression of insect vectored diseases and maintenance of plant health.

ACTIVE INGREDIENT:

Imidacloprid, 1-[(6-Chloro-3-pyridinyl)methyl]-N-nitro-2-imidazolidinimine.....	48.7%
---	-------

OTHER INGREDIENTS:.....	51.3%
-------------------------	-------

100.0%

EPA Reg. No. 264-828

EPA Est. No. 3125-MO-001

Contains 5 pounds of imidacloprid per gallon.

SHAKE WELL BEFORE USING



Weighed bumble bees as they exited/entered nest

Photo credit: Richard Gill



Bumble bee collecting pollen

Photo credit: Dave Goulson

Example: returning lists



```
massCollected = [49, 40, 60, 38, 36, 37, 35, 51, 47, 32]
```

```
treatment = ['C', 'P', 'C', 'C', 'P', 'P', 'P', 'C', 'C', 'P']
```

```
def listInCategory(numList, catList, category):  
    """Returns a list of the numbers in numList  
    that correspond to a particular category from  
    catList"""  
    output = []  
    for index in range(len(numList)):  
        if catList[index] == category:
```

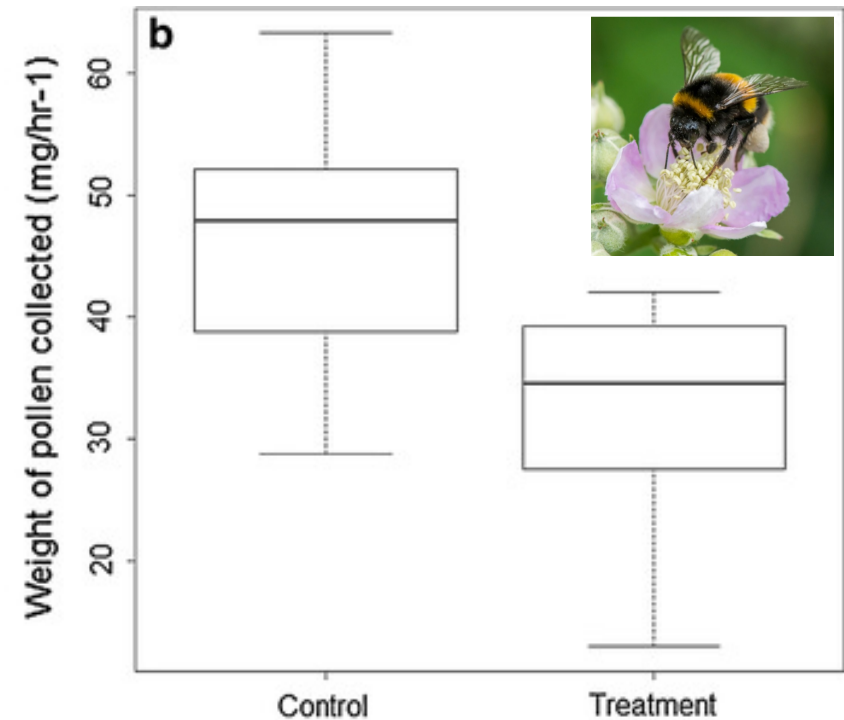


```
massCollectedPest = [49, 60, 38, 51, 47]
```

```
massCollectedContr1 = [40, 36, 37, 35, 32]
```



```
def mean(numList):  
    """Returns the mean of  
    a list of numbers"""  
    sum = 0  
    count = 0  
    for num in numList:  
        sum = sum + num  
        count = count + 1  
    return sum / count
```



36.0

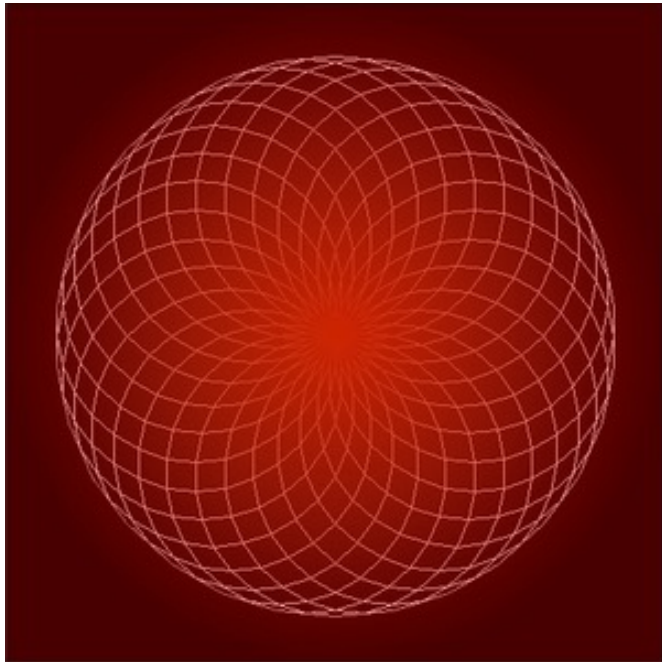
49.0

```
mean(massCollectedPest) < mean(massCollectedContr1)
```

True

Turtle Graphics

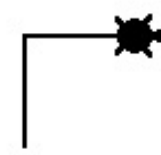
Logo (programming language) [1967]



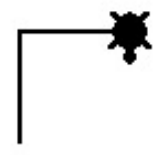
forward 50



right 90



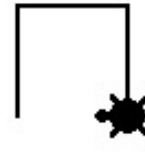
forward 50



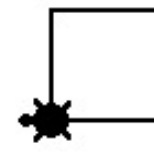
right 90



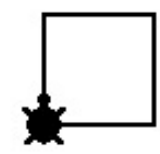
forward 50



right 90



forward 50



right 90



Meet Python's Turtle...

```
>>> import turtle
```

```
>>> turtle.forward(100)
```

```
>>> turtle.right(90)
```



Hey, is it legal to
import turtles?

DEMO!

Turtle Functions...

```
import turtle
```

```
def square(length):  
    """Draws a square with given side length"""  
    for x in range(0, 4):  
        turtle.forward(length)  
        turtle.right(90)
```

```
def polygon(length, sides):  
    """Draws a polygon with given side length  
    and number of sides"""  
    for x in range(0, sides):  
        turtle.forward(length)  
        turtle.right(360.0/sides)
```

Notice that this `for` loop
is just being used to
repeat something 4 times!

Also nothing is returned
by these functions!



DEMO!

Spirograph!



```
import turtle
```

```
def polygon(length, sides):  
    for x in range(0, sides):  
        turtle.forward(length)  
        turtle.right(360.0/sides)
```

```
def spirograph(length, sides, polys):  
    for iteration in range(0, polys):  
        polygon(length, sides)  
        turtle.right(360.0/polys)
```

```
>>> spirograph(50, 6, 10)
```

DEMO!



Name _____

More Mysteries!

```
def mystery1(n):  
    for k in range(1, n):  
        if k*k == n: return True  
    return False
```

```
def mystery2(n):  
    for k in range(1, n):  
        if n == 1:  
            return True  
        elif not (n % 2 == 0): # n % 2 != 0  
            return False  
    else:  
        n = n/2
```

Assume that we run
mystery1 with
positive integers
and mystery2 with
 $n > 2$ as input!



If you have time: how could you
change these functions to print
an error message if the input is
too low?