“Prof. to be Replaced by Penguins,” says HMC Dean

Claremont (AP): Beginning next week CS 5 “Green” will be taught by a pair of penguins announced an HMC dean. “Penguins are very smart,” said the dean, “and we only need to pay them 6 fish per day. Prof. Bush gets paid 9 fish per day and that’s very expensive!” Why are you reading this drivel? You should be paying attention to lecture.
Reading and Lab

• This week: 1.6 – 1.11, Chapter 2
• Lab this week…
  – Please check in at 3 PM in order to get credit for the lab (and stay until done with lab problems or 5 PM, whichever comes first ;^)
  – Practice with for loops and if-elif-else
  – Getting ready for gene finding!
  – Some fun optional bonus problems
• Office hours moved to Sunday 1:45-4:45
  (This week only because I’ll be gone Mon, Tues)
A friendly note from the autograder...

• Please check your function names. They need to be spelled (case-sensitive) as specified in order for the auto-grader to see them (e.g. `findSpam` is different from `findspam`)

• If the autograder reports problems, please try to fix them and resubmit
Biology note: pathogenicity islands and GC content

Due to historical horizontal transfer events, the GC content inside pathogenicity islands sometimes differs from the rest of the genome.

Sp 1, 47% GC

Sp 2, 38% GC

Horizontal transfer of a pathogenicity island

GC: ~47%

GC: ~38%
Collecting perfection…

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

def collect(upTo):
    """ What does this do? """
    perfects = []
    for i in range(1, upTo):
        if perfect(i):
            perfects = perfects + [i]
    return perfects
```

We tried to run this function with `upTo = 100,000...`
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

Try these on your worksheet!

Assume that we run these functions only with positive integers as input!
Collatz Revisited

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

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

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

```python
def testNum(number, repeats):
```
Collatz Revisited

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

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

```python
def testNum(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 few lines of code in your notes!
Collatz Revisited

>>> testNum(16, 10)
True

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

def testNum(number, repeats):
    """ Returns True if the number collatzes within the give 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:
        return n/2
    else:
        return 3*n + 1

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

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

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

Is this indentation of `else` OK?
Collatz Re-Revisited

```python
def testConjecture(upTo, repeats):
    """ Determines if all numbers from 1 to upTo collatz to 1 within given number of repeats """
    for number in range(2, upTo + 1):
        if not testNum(number, repeats):
            return False
    return True

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

Try all numbers from 1 to 20

Up to this many repeats each time!

```python
>>> testConjecture(20, 10)
False
>>> testConjecture(20, 50)
True
DEMO testConjecture.py!
```
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.
Mystery

I love a good mystery!

```python
3 def leppard(inputString):
    """ What does this do? """
4     outputString = ""
5     for symbol in inputString:
6         if symbol == "o":
7             outputString = outputString + "oo"
8         else:
9             outputString = outputString + symbol
10    return outputString

>>> leppard("hello")

>>> leppard("hello to you")
```
I love a good mystery!

```python
3 def leppard(inputString):
    """ What does this do? """
4     outputString = ""
5     for symbol in inputString:
6         if symbol == "o":
7             outputString = outputString + "ooo"
8         else:
9             outputString = outputString + symbol
10    return outputString
```

>>> leppard("hello")
"hellooo"

>>> leppard("hello to you")
"hellooo tooo yooou"
import speech

def leppard(inputString):
    """ Leppard-ifies the input string and speaks it! """
    outputString = ""
    for symbol in inputString:
        if symbol == "o":
            outputString = outputString + "ooo"
        else:
            outputString = outputString + symbol
    speech.say(outputString)
    return outputString
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

The “indirect” or “index” method
Spam counter!

>>> spamCount("I like spam with spamspamspam!")
4
>>> spamCount("spamityspampampam!")
2

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

First attempt...
The direct method
Spam counter!

```python
def spamCount(input):
    counter = 0
    for i in range(len(input)):
        # Finish this in your notes.
        # Hint: Use slicing! (e.g., input[2:5])
        spamCount("I like spam with spamspamspam!")
        4
        012345678911111111
        012345
        spamCount("spamityspampampam!")
        2
```

Ah, the indirect/index method!
Spam counter!

>>> spamCount(“I like spam with spamspamspam!”)
4
   01234567891111111
      0123456
>>> spamCount(“spamityspamspampam!”)
2

def spamCount(input):
    counter = 0
    for i in range(len(input)):
        if input[i:i+4] == “spam”:
            counter = counter + 1
    return counter

Notice that there is no need for an else after the if!
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]  # upgrade to listhood!
    return output

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

01234567891
0

>>> spamFinder(“spamspamity”)  
[0, 4]

>>> spamFinder(“ssspam!”)  
[2]

def spamFinder(input):

Try this in your worksheet!
Spam finder!

01234567891

>>> spamFinder("spamspamity")
[0, 4]

>>> spamFinder("ssspam!")
[2]

def spamFinder(input):
    output = []
    for i in range(0, len(input)):
        if input[i:i+4] == "spam":
            output.append(i)
    return output

Try this in your worksheet!
def returnCodons(DNAstring):
    codonList = []
    for i in range(0, len(DNAstring), 3):
        codonList.append(DNAstring[i:i+3])
    return codonList

0123456789

>>> returnCodons("AAATTTGGGC")
["AAA", "TTT", "GGG", "C"]

What colorful codons you have!
Meet Python’s Turtle…

```python
>>> import turtle

>>> turtle.forward(100)

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

Hey, is it legal to import turtles?

DEMO!
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!
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)