While loops

I will not chew gum in CS 5 Green!
I will not chew gum in CS 5 Green!
I will not chew gum in CS 5 Green!
I will not chew gum in CS 5 Green!
I will not chew gum in CS 5 Green!
I will not chew gum in CS 5 Green!
I will not chew gum in CS 5 Green!
I will not chew gum in CS 5 Green!
I will not chew gum in CS 5 Green!
I will not chew gum in CS 5 Green!
I will not chew gum in CS 5 Green!
Learning Goals

- Distinguish between for loops and while loops
- Implement while loops
- Provide useful background on homework
Gauging your Workload

On your worksheet…

On the front, bottom-right corner…

How many hours, outside of class or lab time, did you spend on this course this past week?
Emma
Naming homework files and functions

In this problem, you'll write several helper functions for manipulating DNA sequences. These functions will be useful in subsequent problems where you'll analyze DNA data in various ways. All of your code for this problem should be in a file called `dna.py`.

### Base complements

Write a function `compBase(N)` that takes a string that is a single DNA base "A", "G", "T", or "C" as input and returns the base that is complementary to it. Here are some examples:

```python
>>> compBase("A")
'T'
>>> compBase("G")
'C'
```

- Case sensitive
- One solution: cut and paste function name when you define it
Editor and shell revisited

(There are many different text editors, you can use whichever you like.)
while loops

def mystery(n):
    k = 1
    while k < n:
        k = k * 2
    return k

>>> mystery(1)

>>> mystery(5)

>>> mystery(10)
while loops

def mystery(n):
    start = 0
    while start <= n:
        print(start)
        start = start + 1
while loops

def bart():
    while True:
        print("I will not chew gum in CS5 Green.")
>>> bart(4)
1 I will not chew gum in CS5 Green.
2 I will not chew gum in CS5 Green.
3 I will not chew gum in CS5 Green.
4 I will not chew gum in CS5 Green.

def bart(times):
    # use for loops
    for line in range(1, times + 1):
        print(line, "I will not chew gum in CS5 Green.")

def bart(times):
    # use while loops
    line = 1
    while line <= times:
        print(line, "I will not chew gum in CS5 Green.")
        line = line + 1
>>> bart(4)
1 I will not chew gum in CS5 Green.
2 I will not chew gum in CS5 Green.
3 I will not chew gum in CS5 Green.
4 I will not chew gum in CS5 Green.

```python
def bart(times):
    # use for loops
    for line in range(1, times + 1):
        print(line, "I will not chew gum in CS5 Green.")

def bart(times):
    # use while loops
    line = 1
    while line <= times:
        print(line, "I will not chew gum in CS5 Green.")
        line = line + 1
```
**while loops**

```python
def collatz(n):
    '''Applies the collatz function to n.'''
    if n % 2 == 0:
        return n/2
    else:
        return 3*n +1

def how_many_times(n):
    ''' Determines the number of times collatz must be
    applied to n before we get 1.'''
    counter = 0
    while n != 1:
        n = collatz(n)
        counter += 1
    return counter

>>> 5 == 5
True
>>> 5 != 6
True

>>> how_many_times(5)
5

5 → 16 → 8 → 4 → 2 → 1
```
while loops

```python
def collatz(n):
    '''Applies the collatz function to n.'''
    if n % 2 == 0:
        return n/2
    else:
        return 3*n + 1

def how_many_times(n):
    '''Determines the number of times collatz must be applied to n before we get 1.'''
    counter = 0
    while n != 1:
        n = collatz(n)
        counter += 1
    return counter

>>> how_many_times(5)
5

5 -> 16 -> 8 -> 4 -> 2 -> 1
```
```python
>>> this_many_primes(5)
[2, 3, 5, 7, 11]

def prime(k):
    '''Returns True if k is prime and False otherwise.''
    for d in range(2, k):
        if k % d == 0:
            return False
    return True

def this_many_primes(n):
    '''Returns a list of the first n primes.''
```
```python
>>> this_many_primes(5)
[2, 3, 5, 7, 11]

def prime(k):
    '''Returns True if k is prime and False otherwise'''
    for d in range(2, k):
        if k % d == 0:
            return False
    return True

def this_many_primes(n):
    '''Returns a list of the first n primes.''
    k = 2
    prime_list = []
    while len(prime_list) < n:
        if prime(k):
            prime_list.append(k)
        k += 1
    return prime_list
```
Homework: gene finding in a region of DNA unique to salmonella
The central dogma in a nutshell

Protein

ATG TAG

RNA

DNA

Promoter S
Finding open reading frames (ORFs): check all 3 reading frames

ATGCCCTAACATGAAAATGACTTAGG

ATGCCCTAACATGAAAATGACTTAGG

ATGCCCTAACATGAAAATGACTTAGG
Genes can occur on either strand

ATG = start codon
TGA, TAG, TAA = Stop codons

Gene 1: coding strand is on top
5' - AATGCCGTGCTTGTTAGACGTTAGGCTTAGATCGTCATGGG - 3'
3' - TTACGGCACGAACATCTGCATCCGAATCTAGCAGTACCC - 5'

Gene 2: coding strand is on bottom
Noncoding sequence between Salmonella genes

It’s orfully strange to find this here!

GCTATCTCAGCAGCCAAATCTCTGCAATGCTCAACAAACGCACGCGGGTTTGAAAGCTTTCGT
GGACGGCCGGTTAGGGGTAGCTTCCGCAATCATCCTCTACGTTTTCTTCAGTTACGCTTGGGTCTGCTGCGC
TGGCGGTATCTCGGTACTTGCTGCAGGATGCTTCAAACGAATTCTAAAAATTATTTTTGGTTTTGATACCGGATAAGCAGTGC

nuoM

nuoL
The unfortunate truth

- Not every ATG is a start codon
- Not every ORF is a gene

How can we separate gene ORFs from ORFs due to random chance?

http://www.flickr.com/photos/lester/3158341/
A simple gene finding strategy

A sequence of interest

AATGGGCGACCAAGGCACATAGACGCGAATCGGACCAGACGACGGCTCACCTGTTCATCTACCTTTCTGTGCTTTGGCGCTAAAAGTTAACGATCGGGCCCTGCGCCGAAACGAAACGTCAGGAATCGACAAATACCAAGTAATCTAAGCTACGGGATAAGCCCCCCCCCCCCCCCCCTCGCGAGAGAGGGGAAGGGGTCAATATTTCCCTGGCCGACTGAAATGGAGTGTACTTACCGGTATACAGTTTGTACTCATTACAGCATTACGCTGTCTTACGACGTATTTCGGGCAATTTCAACATGCTGTCTCTACAGGAGTTTTCGCGCGCTGAAAGAACTCCCATCTAAACCCTG

ORF: 318 nucs
A simple gene finding strategy

A sequence of interest

```
AATGGGCCGACCAAGGCGACATAGACGCGAATCGGACCAGACGCCGGCTCACCTGTTCATCTACCTTTCTG
CGTTGGCGCTAAAAGTTAACGATCGGGCCCTGCGCCGAAACGAAACGTCAGGAATCGACAAATACCAAGTA
TCTAAGCTACGGGATAAGCCCCCCCTCGCGAGAGAGGGGAAGGGGTCAATATTTCCCTGGCCGACTGACAA
TGGAGTGTACTTACCAGGATATACAGTTTGATACCTACAGCCATCGCTGTCTTACGACGTATTCGGGGCATTT
CAACATGCTGTCTCTCAGGAGTTTTCGCGCGCTGAAAGAACTCCCATCTAAACCCTG
```

**ORF:**

```
318 nucs
```

Randomly shuffled versions of this sequence

```
CGCTAGGACCACAAGGATGCGTCCCAACATATCAACGAGGTACGTTTGTGGAAGGCCCCGTATTACCGTC
AGAGACCTGTACGAGGTTGACTATTTTACGCGGAGCCCCAGAAAACCTCAAGCAAGCCGGCATCTTT
GTACGGTACAGCCGCTGTAACCTCGGACCCTGATATGGGAATACACCATGCTGTACGGTC
CTACCACCCCCTGATCCCCCGGTCTCTCTTTCGCTGGTTATAGCTCAAAACTGATCAGGGCTTTAAAAGC
AACAACTGTAACGCCATACCCCGAATTCCCCCGTACACGACGCTAAATGGCTTTCC
```

**Longest ORF:**

```
153 nucs
```

```
CTGCCCTCGCGACGTAAGGGCCTACCCCCTATTCCGGCGCGCTGCTCGTCTCGTCACCTTCTGACGATTTA
ATCCGCTTTAACGACCCGGGGAATGTGCTCAGTAGCAAAACGACTTTGGGGTTACAGCAATGAGGTACCAGGAACCTCTGA
AGTAGGTGGTACGAAACGACACCACTACCTAATGGGAGGGAGGGGACAGCTCCAAACCCGGGAGGGGCAAAAGGCCACCAACCGGG
TATTTTAAGAGGAACCACCCGACTGATCACCACCGGAAAGTTAGGCGCTAAATTATCGTATATCGTAAATCTACCCCTCAAA
CACAAAAACCTCGGCCTGAACTGCTATTACCTGAAAGCTCATTCCCACATTTCCGGGTTCAGCCC
```

**Longest ORF:**

```
156 nucs
```

Don't forget to look at the reverse complements!
Modules and the import statement

```python
>>> L = ['A','C','G','G','T','C','A']

>>> L
['A', 'C', 'G', 'G', 'T', 'C', 'A']

>>> import random

>>> random.shuffle(L)

>>> L
['C', 'T', 'A', 'A', 'C', 'G', 'G']
```
Homework bonus: look and say

- The look-and-say sequence...

1
11
21
1211
111221

What’s next!?
Look-And-Say Sequences (aka “Read-It-And-Weep”)

Number of digits in the $n^{th}$ term of the sequence is given by:

$$C \lambda^n$$

$$\lambda = 1.30357726034296...$$

Conway’s Constant
Homework bonus: average length to ATG vs. AAA

'CGAGGC CGGAT ATCTGGTTTACCCGTACATACTACATTGATGGTGTA . . . '
Monty Hall

Let’s make a deal 1963-1986

inspiring the “Monty Hall paradox”
def monty():
    ''' Simulates one round of the Monty Hall game without switching. '''
    car_door = random.choice([1, 2, 3])
    guess_num = int(input("Which door would you like me to open? "))
    if guess_num == car_door:
        print("You've won a CAR!")
    else:
        print("You've won a goat!")

DEMO!
def nice_monty():
    ''' Simulates one round of the Monty Hall game with switching. '''
    car_door = random.choice([1, 2, 3])
    guess_num = int(input("Which door would you like me to open? "))
    if car_door == 1:
        if guess_num == 1:
            open = random.choice([2, 3])
        elif guess_num == 2:
            open = 3
        elif guess_num == 3:
            open = 2
    elif car_door == 2:
        if guess_num == 1:
            open = 3
        elif guess_num == 2:
            open = random.choice([1, 3])
        elif guess_num == 3:
            open = 1
    elif car_door == 3:
        if guess_num == 1:
            open = 2
        elif guess_num == 2:
            open = 1
        elif guess_num == 3:
            open = random.choice([1, 2])
    print("Look, there's nothing behind door ", open)
    response = input("Would you like to switch doors? (y or n) ")
    if response == 'y':
        guess_num = 6 - guess_num - open
    if guess_num == car_door:
        print("You've won a CAR!")
    else:
        print("You've won a goat!")
import random

def noswitch():
    car_door = random.choice([1, 2, 3])
    guess_num = random.choice([1, 2, 3])
    if guess_num == car_door:
        return 1  # We won 1 car
    else:
        return 0  # We won 0 cars

def switch():
    car_door = random.choice([1, 2, 3])
    guess_num = random.choice([1, 2, 3])
    if car_door == guess_num:
        # We guessed correctly but we are switching!
        return 0
    else:
        # We guessed a door X, the car is behind Y, so Monty opens Z
        # and we switch to Y
        return 1

def marilyn(games):
    print("No switching strategy...")
    cars = 0
    for game in range(games):
        cars = cars + noswitch()
    print("You won a car this many times: ", cars, " out of ", games, " games")
    print("The win-lose ratio was: ", 1.0*cars/games)

    print("Switching strategy...")
    cars = 0
    for game in range(games):
        cars = cars + switch()
    print("You won a car this many times: ", cars, " out of ", games, " games")
    print("The win-lose ratio was: ", 1.0*cars/games)
What’s the explanation?


xkcd: Monty Hall (https://xkcd.com/1282/)