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

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

>>> mystery(1)
>>> mystery(5)
>>> mystery(10)
```
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):
    for line in range(1,times+1):
        print(line,"I will not chew gum in CS5 Green.")

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

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

def howManyTimes(n):
   ''' Determines the number of times that collatz must be applied to n before we get 1 '''
   counter = 0
   while n > 1:
      counter += 1
      n = collatz(n)

>>> 5 == 5
True

>>> 5 != 6
True

>>> howManyTimes(5)
5

5 -> 16 -> 8 -> 4 -> 2 -> 1
while loops

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

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

>>> howManyTimes(5)

DEMO!
A Prime Example of Looping!

```python
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 thisManyPrimes(n):
    '''Returns a list of the first n primes. '''
```

```python
c> thisManyPrimes(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 thisManyPrimes(n):
    ''' Returns a list of the first n primes. '''
    k=2
    primeList=[]
    while len(primeList) < n:
        if prime(k):
            primeList.append(k)
        k += 1
    return primeList

A Prime Example of Looping!

>>> thisManyPrimes(5)
[2, 3, 5, 7, 11]
Genes can occur on either strand

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

Gene 1: coding strand is on top
5' – AATGCCGTGCTTTGATACGATAGGCTTTAGATCGTCATGGG – 3'
3' – TTACGGCAGCAACATCTGCATCCGAAATCTAGCAGTACCC – 5'

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

GCTATCTCAGTTGCGAGCTTAGCTCAGTTTTTGAGTCCGTTAA
GGACGGCCGTAATTTGGAGCTTGCAGCGGTACATCCTACAGTCTCAGTCCGTTTGCTGCGC
TGCCGGTTATCTCGGCTGGACTTCCCGGATGCGCTGTGCGC
ATGACCGCTGTACAGGAGAAGGATTTTCAACGAATTCTAAAA
ATTATTTTGGTGTTTAGGCAGGATAAGCAGCTGCC

nuoM

nuoL

Its orfully strange to find this here!
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?
A simple gene finding strategy

A sequence of interest

AATGGGCGGACCAAGGCACATAGACGCAGAATCTGGACCAGACGCGCCGCTCACCTGTTGTCATCTACCTTTCTG
CGTTGGCGCTAAAAGTTAACGATCGGGCCCTGCGCCGAAACGAAACGTCAGGAATCGACAAATACCAAGTA
TCTAAGCTACGGGATAAGCCCCCCCTCGCGAGAGAGGGGAAGGGGTCAATATTTCCCTGGCCGACTGACAA
TGGAGTGTACTTACCAGGTATACAGTTTTGTACTCTACAGCCATCGCTGTCTTTACGACGTATTCCGAGGCATTT
CAACATGCTGTCTCAGGAGTTTTCGCGCGCTGAAAGAACTCCCATCTAAACCTG

ORF: 318 nucs
A simple gene finding strategy

A sequence of interest

AATGGGCGGACCAAGGCGACATAGACGCGAATCGGACCAGACGCCGCTCACCTGTCATCTACTACCCCTTG
CGTTGGCGCTAAAGTTAACGATCAGGCCCTGCGCCAACGAAACGTCAGGAATCGACAAATACCAAGTA
TCTAAGCTACGGGATAAGCCCCCTCTCAGGAGAGGGGAAGGGGTCAATATTCCCTTGGCCGACTGACAA
TGGAGTGTACTTACCGGTATACGTCTTTGTAACCTACAGTCATCCGTGCTTACGCATTTGCGGCAATTT
CAACATGCTGTCTCTCAGGAGCTTTTTCGCGGCGCTGAAAGAACTCCCATCTAAACCCTG

Randomly shuffled versions of this sequence

ORF: 318 nucs

CGCTAGGCACGAAAGGATGGCGTCCCAACATATCAACGAGGTACGTTTGTGGAAGGCCCCGTATTACCGTC
AGAGACCTGGATCGAGGGGTACATTAAACGGCGGAGCCCCAAAGAAAAACTCAAAGAAACGGGCGATCATCTTT
GTCGGGTACACGGCTGTCAACTCGGACCGGTATTTGAATACTACCGGATGCAACTGGGATTGACGGGATG
CTACCACCCTCGATCCCACCGTCTCTCTCTCTTGCTGGGGTTATAGCTCAAACCTCTGATCAGCGCTTTAAAGC
AACAAACTGTAACGCCCATACCCCCGAAATCCCCGTTACCAGACGACGTTAATGCTTTCC

CTGCCCTCGCGACGTAAAGGCTTACCCCTTATTCCGCGGTCTGCTTCTGGTGTCATCTTATTCGACAATTATTA
ATCCGCTTTACGCAACCGCGGATGGTCTGACATCGCCACAATCCCGATCATGGTGATGTTGCTACAGTTTA
AGTATGGGTGGGAGTCGACCACGTCCATTTAGGAGGGAGCCAGTTCAACCCGAGGCGCGGGCGACCGACACCCGG
TATTTAAGGAGAACCACCGCTTGTATCACCACCGGAGAACGGTAGCGCTCAAATTATTCGCTCAATCTACCCCTCAA
CACAACCGCTCGGCTGAAAGAACTCCCATCTACTTTTCGCGGTCAGGCC

Longest ORF: 153 nucs

Randomly shuffled versions of this sequence

CGCTAGGCACGAAAGGATGGCGTCCCAACATATCAACGAGGTACGTTTGTGGAAGGCCCCGTATTACCGTC
AGAGACCTGGATCGAGGGGTACATTAAACGGCGGAGCCCCAAAGAAAAACTCAAAGAAACGGGCGATCATCTTT
GTCGGGTACACGGCTGTCAACTCGGACCGGTATTTGAATACTACCGGATGCAACTGGGATTGACGGGATG
CTACCACCCTCGATCCCACCGTCTCTCTCTCTTGCTGGGGTTATAGCTCAAACCTCTGATCAGCGCTTTAAAGC
AACAAACTGTAACGCCCATACCCCCGAAATCCCCGTTACCAGACGACGTTAATGCTTTCC

CTGCCCTCGCGACGTAAAGGCTTACCCCTTATTCCGCGGTCTGCTTCTGGTGTCATCTTATTCGACAATTATTA
ATCCGCTTTACGCAACCGCGGATGGTCTGACATCGCCACAATCCCGATCATGGTGATGTTGCTACAGTTTA
AGTATGGGTGGGAGTCGACCACGTCCATTTAGGAGGGAGCCAGTTCAACCCGAGGCGCGGGCGACCGACACCCGG
TATTTAAGGAGAACCACCGCTTGTATCACCACCGGAGAACGGTAGCGCTCAAATTATTCGCTCAATCTACCCCTCAA
CACAACCGCTCGGCTGAAAGAACTCCCATCTACTTTTCGCGGTCAGGCC

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: gene finding in a region of DNA unique to salmonella

Salmonella pathogenicity island 1
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 $n^{\text{th}}$ term of the sequence is given by:

$$C \lambda^n$$

$$C = 1.567...$$

$$\lambda = 1.30357726034296...$$

The quantity $\lambda$ is known as Conway’s constant (Sloane’s A014715), and amazingly is given by the unique positive real root of the polynomial:

$$D = x^{71} - x^{69} - 2 x^{68} - x^{67} + 2 x^{66} + 2 x^{65} + x^{64} - x^{63} - x^{62} - x^{61} - x^{60} - x^{59} + 2 x^{58} + 5 x^{57} + 3 x^{56} - 2 x^{55} - 10 x^{54} - 3 x^{53} - 2 x^{52} + 6 x^{51} + 6 x^{50} + x^{49} + 9 x^{48} - 3 x^{47} - 7 x^{46} - 8 x^{45} - 8 x^{44} + 10 x^{43} + 6 x^{42} + 8 x^{41} - 5 x^{40} - 12 x^{39} + 7 x^{38} - 7 x^{37} + 7 x^{36} + x^{35} - 3 x^{34} + 10 x^{33} + x^{32} - 6 x^{31} - 2 x^{30} - 10 x^{29} - 3 x^{28} + 2 x^{27} + 9 x^{26} - 3 x^{25} - 14 x^{24} - 8 x^{23} - 7 x^{21} + 9 x^{20} - 3 x^{19} - 4 x^{18} - 10 x^{17} - 7 x^{16} + 12 x^{15} + 7 x^{14} + 2 x^{13} - 12 x^{12} - 4 x^{11} - 2 x^{10} - 5 x^9 + x^7 - 7 x^5 + 7 x^4 - 4 x^3 + 12 x^2 - 6 x^2 + 3 x - 6,$$

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

'CGAGGC\textcolor{red}{GCGGAT}ATATCTGGTTTACC\textcolor{red}{CGTACA}TACTAC\textcolor{red}{ATTG}ATGTTGTA...'}
Monty Hall

Let’s make a deal 1963-1986

inspiring the “Monty Hall paradox”

Sept. 1990
```python
import random

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

def niceMonty():
    '''Simulates one round of the Monty Hall game with switching.''
    cardoor = random.choice([1, 2, 3])
    guessNum = int(input("Which door would you like me to open? "))
    if cardoor == 1:
        if guessNum == 1:
            open = random.choice([2, 3])
        elif guessNum == 2:
            open = 3
        elif guessNum == 3:
            open = 2
    elif cardoor == 2:
        if guessNum == 1:
            open = 3
        elif guessNum == 2:
            open = random.choice([1, 3])
        elif guessNum == 3:
            open = 1
    elif cardoor == 3:
        if guessNum == 1:
            open = 2
        elif guessNum == 2:
            open = 1
        elif guessNum == 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':
        guessNum = int(input("Which door would you like? "))
    if guessNum == cardoor:
        print("You've won a CAR!")
    else:
        print("You've won a goat!")
```python
import random

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

def switch():
    cardoor = random.choice([1, 2, 3])
    guessNum = random.choice([1, 2, 3])
    if cardoor == guessNum:
        # 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(0, 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(0, 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?