Before you turn this assignment in, make sure everything runs as expected.

You can either **restart the kernel** (in the menu bar, select Kernel → Restart and Clear Output) and then **run all cells** (in the menu bar, select Cell → Run All). Or, you can simply **clear the output for all cells** (in the menu bar, select Cell → All Output → Clear) and then **run all cells** (in the menu bar, select Cell → Run All).

In [1]:
```
NAME = "Jacqueline Bungay"
COLLABORATORS = ""
```

Assignment 1 - Intro to Python

(15 points)

Learning Outcomes

In this assignment you will practice:

- Basic Python syntax
- Writing simple Python functions
- Working with strings and dictionaries

Question #1: Count symbols

(3 points)

In this question, you are given a string `s` which represents a DNA string. The string `s` consists of symbols `'A', 'C', 'G',` and `'T'`. An example of a length 21 DNA string is `"ATGCTTCAGAAAGGTCTTACG."`

Your task is to write a code which will count the number of times each of the symbols `'A', 'C', 'G'`, and `'T'` occur in `s`. Your code should generate a **list of 4 integers** and **print it out**.

In [2]:
```
# Here is the DNA string:

s = 'AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGC'
```

In [3]:
```
symbols_list = ['A', 'C', 'G', 'T']
symbol_counts = {}      # Empty dictionary to hold counts for each symbol.
                        #     key = symbol value from symbols_list
                        #   value = count of the occurances of symbol in DNA string s

''' Add to dictionary symbol i as a key and associated count as the value.
        for i in symbols_list:
            symbol_counts[i] = s.count(i)
'''
symbol_counts = {i:s.count(i) for i in symbols_list}

for j in symbol_counts:   # Print out the count of each symbol in string s.
    print('Number of', j ,"in string s =",symbol_counts[j])
```

```
Number of A in string s = 20
Number of C in string s = 12
Number of G in string s = 17
Number of T in string s = 21
```

In [4]:
```
print(symbol_counts.values())
```

```
dict_values([20, 12, 17, 21])
```

In [5]:
```
# ---- Practice with my own count_symbols() function ----

def count_symbols(dna_string, symbol):
    '''This function will count the number of times a symbol occurs in the DNA string

       Input:
           dna_string: A string of DNA symbols
               symbol: A DNA symbol to search in the string

       Output:
           count: The number of times the symbol occurs in the DNA string

    '''
    count = 0         #initialize counter to 0
    for x in dna_string:
        if x == symbol:
            count += 1    #increment counter each time symbol is found in the dna_string
        else:
            pass
    return count
```

In [6]:
```
symbols_list = ['A', 'C', 'G', 'T']
symbol_counts = {}      # Empty dictionary to hold counts for each symbol.
                        #     key = symbol from symbols_list
                        #   value = count of symbol in DNA string s

# Add to dictionary symbol i as a key and associated count as the value.
symbol_counts = {i:count_symbols(s, i) for i in symbols_list}

for j in symbol_counts:   # Print out the count of each symbol in string s.
    print('Number of', j ,"in string s =",symbol_counts[j])
```

```
Number of A in string s = 20
Number of C in string s = 12
Number of G in string s = 17
Number of T in string s = 21
```

Question #2: Find a substring

(4 points)

You are given a dictionary of the US states and their capitals. The keys in the dictionary are states and the values are capital names.

Write a code to return a **list of all capitals** that contain the name of a state in their name as a substring.

HINT: For example, `Indianapolis` as a capital name and `Indiana` as a state name is one of the key/value pairs that your code would find. Your

In [7]:
```
# Run this cell to create a dictionary of states' capitals
capitals={
    'Alabama': 'Montgomery',
    'Alaska': 'Juneau',
    'Arizona':'Phoenix',
    'Arkansas':'Little Rock',
    'California': 'Sacramento',
    'Colorado':'Denver',
    'Connecticut':'Hartford',
    'Delaware':'Dover',
    'Florida': 'Tallahassee',
    'Georgia': 'Atlanta',
    'Hawaii': 'Honolulu',
    'Idaho': 'Boise',
    'Illinios': 'Springfield',
    'Indiana': 'Indianapolis',
    'Iowa': 'Des Monies',
    'Kansas': 'Topeka',
    'Kentucky': 'Frankfort',
    'Louisiana': 'Baton Rouge',
    'Maine': 'Augusta',
    'Maryland': 'Annapolis',
    'Massachusetts': 'Boston',
    'Michigan': 'Lansing',
    'Minnesota': 'St. Paul',
    'Mississippi': 'Jackson',
    'Missouri': 'Jefferson City',
    'Montana': 'Helena',
    'Nebraska': 'Lincoln',
    'Neveda': 'Carson City',
    'New Hampshire': 'Concord',
    'New Jersey': 'Trenton',
    'New Mexico': 'Santa Fe',
    'New York': 'Albany',
    'North Carolina': 'Raleigh',
    'North Dakota': 'Bismarck',
    'Ohio': 'Columbus',
    'Oklahoma': 'Oklahoma City',
    'Oregon': 'Salem',
    'Pennsylvania': 'Harrisburg',
    'Rhoda Island': 'Providence',
    'South Carolina': 'Columbia',
    'South Dakota': 'Pierre',
    'Tennessee': 'Nashville',
    'Texas': 'Austin',
    'Utah': 'Salt Lake City',
```

```
    'Vermont': 'Montpelier',
    'Virginia': 'Richmond',
    'Washington': 'Olympia',
    'West Virginia': 'Charleston',
    'Wisconsin': 'Madison',
    'Wyoming': 'Cheyenne'
}
capitals
```

Out[7]:

```
{'Alabama': 'Montgomery',
 'Alaska': 'Juneau',
 'Arizona': 'Phoenix',
 'Arkansas': 'Little Rock',
 'California': 'Sacramento',
 'Colorado': 'Denver',
 'Connecticut': 'Hartford',
 'Delaware': 'Dover',
 'Florida': 'Tallahassee',
 'Georgia': 'Atlanta',
 'Hawaii': 'Honolulu',
 'Idaho': 'Boise',
 'Illinios': 'Springfield',
 'Indiana': 'Indianapolis',
 'Iowa': 'Des Monies',
 'Kansas': 'Topeka',
 'Kentucky': 'Frankfort',
 'Louisiana': 'Baton Rouge',
 'Maine': 'Augusta',
 'Maryland': 'Annapolis',
 'Massachusetts': 'Boston',
 'Michigan': 'Lansing',
 'Minnesota': 'St. Paul',
 'Mississippi': 'Jackson',
 'Missouri': 'Jefferson City',
 'Montana': 'Helena',
 'Nebraska': 'Lincoln',
 'Neveda': 'Carson City',
 'New Hampshire': 'Concord',
 'New Jersey': 'Trenton',
 'New Mexico': 'Santa Fe',
 'New York': 'Albany',
 'North Carolina': 'Raleigh',
 'North Dakota': 'Bismarck',
 'Ohio': 'Columbus',
 'Oklahoma': 'Oklahoma City',
 'Oregon': 'Salem',
 'Pennsylvania': 'Harrisburg',
```

```
 'Rhoda Island': 'Providence',
 'South Carolina': 'Columbia',
 'South Dakota': 'Pierre',
 'Tennessee': 'Nashville',
 'Texas': 'Austin',
 'Utah': 'Salt Lake City',
 'Vermont': 'Montpelier',
 'Virginia': 'Richmond',
 'Washington': 'Olympia',
 'West Virginia': 'Charleston',
 'Wisconsin': 'Madison',
 'Wyoming': 'Cheyenne'}
```

In [8]:
```
''' Create an empty list that will hold names of capitals where the state name
is a substring of the capital name.'''

list_of_capitals = []

'''If the state name can be found as a substring in the capital name, then add to the list_of_capitals

for state in capitals:
    if capitals[state].find(state) != -1:
        list_of_capitals.append(capitals[state])
    else:
        pass
'''

list_of_capitals = [capitals[state] for state in capitals if capitals[state].find(state) != -1]

print("The list of captials are", list_of_capitals ,"that contain the name of a state in their name as a substri
```

```
The list of captials are ['Indianapolis', 'Oklahoma City'] that contain the name of a state in their name as a
substring.
```

In [ ]:
```

```

Question #3: Is a data point within a rectangle?

(4 points)

Write a function `isIn()` which returns **boolean `True`** if a point is within a rectangle specified by two sets of coordinates and **boolean `False`** if the point is outside the rectangle. The function should accept three parameters:

- the first parameter is a set of coordinates which defines one of the corners of the rectangle,
- the second parameter is also a set of coordinates that defines the second corner,
- the third set of coordinates defines a single point which is being tested.

For example,

- `isIn((1,2), (3,4), (1.5, 3.2))` should return `True`,
- `isIn((4,3.5), (2,1), (3, 2))` should return `True`,
- `isIn((-1,0), (5,5), (6,0))` should return `False`,
- `isIn((4,1), (2,4), (2.5,4.5))` should return `False`.

Test your function with at least 2 different sets of data points in addition to the examples above.

NOTES:

1. If the point being tested is on the side of the rectangle, consider it to be within the rectangle. For example, if the rectangle is defined as `(1,2), (3,4)` and the point is `(2,2)`, the function should return `True`.
2. In this assignment, we assume that the edges of the rectangle are parallel to coordinate axes.
3. We also assume that the first parameter does not always represent the left corner of the rectangle and the second parameter is not always the right corner. The function should work correctly either way. Please note the second test condition above where the first parameter, `(4,3.5)`, represents the top-right corner and the second parameter, `(2,1)`, represents left-bottom corner.

In [9]:
```
def isIn(firstCorner=(0,0), secondCorner=(0,0), point=(0,0)):

    '''This function will determine if a point is within a rectangle specified by two sets of coordinates
       which are the two corner points of a rectangle that are diagonal to each other (AC or BD or CA or DB) .

       A(x1,y1)---------B(x2,y2)
          |                |
          |                |
       D(x4,y4)---------C(x3,y3)


       Input:
          firstCorner: Is a set of coordinates which defines one of the corners of the rectangle
         secondCorner: Is a set of coordinates that defines the second corner
                point: Defines a single point which is being tested if it is in the rectangle

        Output:
            Boolean value: True, if the point is within the rectangle or
                           False, if the point is outside the rectangle
    '''

    corner_points = []     # Empty list for corner points firstCorner and secondCorner.
    y_coords = []          # Empty list of y coordinates from corner_points.
    x = 0                  # List index for x coordinates
    y = 1                  # List index for y coordinates
    first_coord = 0        # First coordinate location in the list
    second_coord = 1       # Second coordinate location in the list

    ''' If two corner points are not valid, then return False if:
            It's the same point
            It's a vertical line
            It's a horizontal line '''

    if firstCorner == secondCorner or firstCorner[x] == secondCorner[x] or firstCorner[y] == secondCorner[y]:
        print ("First and second rectangle corner points must be end points of a diagonal line.")
        return False


    corner_points.append(firstCorner)
    corner_points.append(secondCorner)
    corner_points.sort()   # sort corner_points list so they are in ascending order by x.

    ''' Is the x coordinate of point on or within the x coordinates of the firstCorner and the secondCorner?
        If True then check y coordinates else return False'''

    if point[x] >= corner_points[first_coord][x] and point[x] <= corner_points[second_coord][x]:
```

```
        ''' Sort corner_points y coordinates first before determining whether point y coordinate is on or within
            the range of the y corner_points'''
        y_coords.append(corner_points[first_coord][y])
        y_coords.append(corner_points[second_coord][y])
        y_coords.sort()      # sort corner_points y so they are in ascending order.

        if point[y] >= y_coords[first_coord] and point[y] <= y_coords[second_coord]:
            return True   # point(x, y) is in the rectangle.
        else:
            return False  # point y is not in the rectangle.
    else:
        return False      # point x is not in the rectangle.
```

In [10]:
```
isIn((1,2), (3,4), (1.5, 3.2))   #Question 3 test data points
```

Out[10]: True

In [11]:
```
isIn((4,3.5), (2,1), (3, 2))     #Question 3 test data points
```

Out[11]: True

In [12]:
```
isIn((-1,0), (5,5), (6,0))       #Question 3 test data points
```

Out[12]: False

In [13]:
```
isIn((4,1), (2,4), (2.5,4.5))    #Question 3 test data points
```

Out[13]: False

In [14]:
```
isIn((-2,-2), (-5,-4), (1,-3))   #Jackie's test data points
```

Out[14]: False

In [15]:
```
isIn((-4,2), (1,-1), (-2,1))     #Jackie's test data points
```

Out[15]: True

In [16]:
```
isIn((1,5), (1,9), (3,-2))  # Jackie's test data points - vertical line - not valid
```

```
First and second rectangle corner points must be end points of a diagonal line.
```

Out[16]: False

Question #4: Are all points within a rectangle?

(4 points)

Modify your function from the previous question so it takes a list of points rather than a single point and returns **boolean `True`** only if all points in the list are in the rectangle.

For example,

- `allIn((0,0), (5,5), [(1,1), (0,0), (5,5)])` should return `True`
- but `allIn((0,0), (5,5), [(1,1), (0,0), (5,6)])` should return `False`
- empty list of points `allIn((0,0), (5,5), [])` should return `False`

Use the same assumptions as above about the placement of the points and how rectangle is defined. Make sure that your function returns `False` for empty list of points (no values).

Test your function with at least 3 different sets of data points.

In [17]:
```
def allIn(firstCorner=(0,0), secondCorner=(0,0), pointList=[]):
    '''This function will determine if a list of points is within a rectangle specified by two sets of coordinat
       which are the two corner points of a rectangle that are diagonal to each other (AC or BD or CA or DB) .

       A(x1,y1)---------B(x2,y2)
          |                |
          |                |
       D(x4,y4)---------C(x3,y3)


       Input:
          firstCorner: Is a set of coordinates which defines one of the corners of the rectangle
         secondCorner: Is a set of coordinates that defines the second corner
                pointList: A list of points which is being tested if they are all in the rectangle

        Output:
            b_AllPts: True, if all the points are within the rectangle or
                      False, if at least one of the points in the list of points is outside the rectangle

        Exception output: A message is printed and False is returned when firstCorner and secondCorner points
                          do not represent a diagonal line or the pointList is empty.

    '''

    corner_points = []     # Empty list for corner points firstCorner and secondCorner.
    y_coords = []          # Empty list of y coordinates from corner_points.
    x = 0                  # List index for x coordinates
    y = 1                  # List index for y coordinates
    first_coord = 0        # First coordinate location in the list
    second_coord = 1       # Second coordinate location in the list


    ''' If two corner points are not valid, then return False if:
            It's the same point
            It's a vertical line
            It's a horizontal line '''

    if firstCorner == secondCorner or firstCorner[x] == secondCorner[x] or firstCorner[y] == secondCorner[y]:
        print ("First and second rectangle corner points must be end points of a diagonal line.")
        return False

    if pointList == []:
        print ("List of points is empty.")
        return False

    corner_points.append(firstCorner)
```

```
    corner_points.append(secondCorner)
    corner_points.sort()   # sort corner points list so they are in ascending order by x.

    y_coords.append(corner_points[first_coord][y])
    y_coords.append(corner_points[second_coord][y])
    y_coords.sort()        # sort corner_points y so they are in ascending order.

    '''
        If all points in the pointsList are within the rectangle then return True.
        If at least one point in the pointList is not in the rectangle then return False.'''

    for point in pointList:
        '''Is the x coordinate of each point in the pointList on or within the x coordinates of the firstCorner
           and the secondCorner? '''
        if point[x] >= corner_points[first_coord][x] and point[x] <= corner_points[second_coord][x]:
            # Then is the y coordinate on or within the range of the y corner_points
            if point[y] >= y_coords[first_coord] and point[y] <= y_coords[second_coord]:
                b_AllPts = True      # This point(x,y) of pointList is in the rectangle.
            else:
                b_AllPts =  False    # point y is not in the rectangle.
                break;
        else:
            b_AllPts =  False        # point x is not in the rectangle.
            break;

    return b_AllPts
```

In [18]:
```
allIn((0,0), (5,5), [])    # Question 4 test data points
```

```
List of points is empty.
```

Out[18]: False

In [19]:
```
allIn((0,0), (5,5), [(1,1), (0,0), (5,5)])   # Question 4 test data points
```

Out[19]: True

In [20]:
```
allIn((0,0), (5,5), [(1,1), (0,0), (5,6)])   # Question 4 test data points
```

Out[20]: False

In [21]:
```
allIn((0,0), (5,5), [(1,1), (9,2), (5,5)])   # Jackie's test data points
```

Out[21]: False

In [22]:
```
allIn((0,0), (5,5), [(1,1), (-5.2,0), (5,2)])   # Jackie's test data points
```

Out[22]:
```
False
```

In [23]:
```
allIn((5,2), (3,-2), [(4,1), (3,0), (5,1),(4,-1.2)])   # Jackie's test data points
```

Out[23]:
```
True
```

In [ ]:
```

```