Deliverables: 36 total points

- seqCleaner 9 points
- fastqParse 9 points
- coordinateMathSoln 9 points
- converter 9 points
- Optional: Triad class rewrite (5 points extra credit)

  These must be submitted as 4(or 5) .py files,

  Due: Monday January 18, 2021 11:55pm

Lab 2 – Manipulating Data Types (36 points)

Overview

For this assignment, create a Lab02 folder inside your BME160 folder. Download the Lab02.ipynb file to that new folder and we will work on the notebook in that folder.

Last week we introduced the Python function input(), which is used to take in string data. This week we will use input() to take in string and numeric (float and integer) data.

Remember that all data returned from input() is a string object, so this will mean that you need to convert any numeric data to their respective numeric objects. The exercises in this lab will give you practice manipulating various types of data that commonly arise in computational biology problems.

We are again using jupyter for this assignment, and as a final step we will create distinct command line programs from each of the cells that make up these programs. Each of those program files are text files and will have names like: seqCleaner.py, or fastqParse.py. Please save each of the four (or five) python programs as .py files into your LAB02 folder. You can copy the text from the code cell, then paste it into an editor like notepad or textedit. Save that new file with the appropriate name (eg. seqCleaner.py) into your LAB02 folder. Using terminal or cmd or conda, you can now do a final test on that new program with the command:
python3 seqCleaner.py

All together, you will submit:

- seqCleaner.py,
- fastqParse,py,
- coordinateMathSoln.py,
- converter.py, and
- (optionally) Triad.py

Sequence cleanup

In this exercise, you will create a program to "clean up" a sequence of DNA by removing ambiguous bases (denoted by "N") output from a sequencer. Your task is to create a Python program called seqCleaner that

- asks for and collects a sequence of DNA using input()
- removes the ambiguous parts of the sequence, outputs the "cleaned" sequence, replacing the ambiguous parts with a count in {}'s. For example, if I enter the sequence of DNA "AaNNNNNNGTC" (without quotes), the program will output:
  AA{6}GTC

Hints:

- The input sequence is not guaranteed to be uppercase, but should be interpreted as though it is all uppercase.
- Only the letters (A,C,G,T,N) will be included in the input.
- Only the DNA characters (A,C,G,T) should remain after the cleanup.
- The input will include, at most, one block of 'N' characters.

To get full credit on this assignment, your code needs to:

- Run properly (execute and produce the correct output)
- Contain docstrings and specific line or block comments that explain the semantics of your implementation.
- Include any assumptions or design decisions you made in writing your code
- Include an overview describing what your program does with expected inputs and outputs. This should be in the form of a program level dosctring.

seqCleaner

In [ ]:
```python
#!/usr/bin/env python3
# Name: Your full name (CATS account username)
# Group Members: List full names (CATS usernames) or "None"

'''
Read a DNA string from user input and return a collapsed substring of embedded Ns to: {count}.

Example:
 input: AaNNNNNNGTC
output: AA{6}GTC

Any lower case letters are converted to uppercase
'''

class DNAstring (str):
    def length (self):
        return (length(self))

    def purify(self):
        ''' Return an upcased version of the string, collapsing a single run of Ns.'''
        pass # this is just a placeholder, so dont leave "pass" here

def main():
    ''' Get user DNA data and clean it up.'''
    data = input('DNA data?')
    thisDNA = DNAstring (data)
    pureData = thisDNA.purify()
    print (pureData)

main()
```

Sequence information parsing

In this exercise, you will create a program to "parse" sequence name information from a single line of a FASTQ formatted file. Your task is to create a Python script called fastqParse that:

- asks for and collects the seqname line of a FASTQ file using input()
- parses out each field of the run information from the string and displays each of them on a new line For example, if I enter the FASTQ seqname line: ***@EAS139:136:FC706VJ:2:2104:15343:197393*** then the program will output:
  Instrument = EAS139
  Run ID = 136
  Flow Cell ID = FC706VJ
  Flow Cell Lane = 2
  Tile Number = 2104
  X-coord = 15343
  Y-coord = 197393 ### Hints:
- The input string is guaranteed to have 7 fields.
- The first character of the FASTQ seqname line is "@" and each field of the run information is separated by a colon":".
- A reasonable solution would be around 16 lines of code excluding comments.

To get full credit on this assignment, your code needs to:

- Run properly (execute and produce the correct output)
- Contain documentation/comments
- Include any assumptions or design decisions you made in writing your code
- Include an overview describing what your program does with expected inputs and outputs

fastqParse

In [ ]:
```python
#!/usr/bin/env python3
# Name: Your full name (CATS account username)
# Group Members: List full names (CATS usernames) or "None"

'''
Program docstring goes here.
'''

class FastqString (str):
    ''' Class docstring goes here.'''
    def parse(self):
        ''' Method docstring goes here.'''
        pass

def main():
    ''' Function docstring goes here.'''
    pass

main()
```

Protein coordinates

In this exercise, you will create a program that takes three sets of atomic coordinates, all provided on a **single line**. The program then calculates the bond lengths and angles. For this program, you can start with the Triad class (provided). Your task is to create a Python program called coordinateMathSoln.py that:

- asks for and collects three sets of coordinates using input(), only use 1 line for this data !!
- outputs the N-C and N-Ca bond lengths and the C-N-Ca bond angle with correct number of significant digits (see below)

For example, if I enter the following coordinates **(notice.. they are all on one line !!!)** :

C = (39.447, 94.657, 11.824) N = (39.292, 95.716, 11.027) Ca = (39.462, 97.101, 11.465)

then the program will output the following three lines:
N-C bond length = 1.33
N-Ca bond length = 1.46
C-N-Ca bond angle = 124.0
(Note: make sure that the angle returned is in **degrees** !!)

Hints:

- Each coordinate will contain only 3 numeric values.
- Bond lengths are the distance between two points, which for points P and Q in 3-space, (P_x, P_y, P_z) and (Q_x, Q_y, Q_z) respectively, the distance between them is:

$$\|PQ\| = \sqrt{(P_x - Q_x)^2 + (P_y - Q_y)^2 + (P_z - Q_z)^2)}$$
$$= \sqrt{\sum^{i \in x,y,z} (P_i - Q_i)^2}$$

- Bond angles can be calculated from the dot product.

Let's say that we have three points in space labeled P, Q and R. We are interested in the angle at point Q that is made from line segments QP and QR. The dot product tells us (for standard vectors P and R) that:

$$P \cdot R = \|P\|\|R\| \cos\theta$$

in this notation, $\|P\|$ refers to the length of vector P as a standard vector (assumed to begin at the origin (0,0,0)). We can then see that the angle between vectors P and R can be found by:

$$\cos\theta = \frac{P \cdot R}{\|P\|\|R\|}$$

We can calculate the dot product using the sum of products of the vector components:

$$P \cdot R = \sum^{i \in x,y,z} P_i R_i$$

Now, to find vector P in standard form, we need to remember that QP starts at Q, so we need to place the origin at Q and find out where P is in that new space. We do that by subtracting the components of Q from P. Putting all of this together, we get:

$$\theta = \cos^{-1} \frac{\sum^{i \in x,y,z} (P_i - Q_i)(R_i - Q_i)}{\|QP\|\|QR\|}$$

Remember, θ is in radians.

Below I have given you a class (Triad) with methods to calculate dot products (dot), dot products of translated vectors (ndot), distances (dPQ, dPR, dPR) and angles in radians (angleP, angleQ and angleR) for each of the three points in a Triad object. A reasonable solution for this exercise involves around 12 lines of additional code excluding comments.

To get full credit on this assignment, your code needs to:

- Run properly (execute and produce the correct output)
- Contain docstrings and line comments (using #)
- Include any assumptions or design decisions you made in writing your code
- Include an overview describing what your program does with expected inputs and outputs as a program level docstring

coordinateMathSoln

In [ ]:
```python
#!/usr/bin/env python3
# Name: Your full name (CATS account username)
# Group Members: List full names (CATS usernames) or "None"

'''
Program docstring goes here
'''

import math
class Triad :
    """
    Calculate angles and distances among a triad of points.

    Author: David Bernick
    Date: March 21, 2013
    Points can be supplied in any dimensional space as long as they are consistent.
    Points are supplied as tupels in n-dimensions, and there should be three
    of those to make the triad. Each point is positionally named as p,q,r
    and the corresponding angles are then angleP, angleQ and angleR.
    Distances are given by dPQ(), dPR() and dQR()

    Required Modules: math
    initialized: 3 positional tuples representing Points in n-space
             p1 = Triad( p=(1,0,0), q=(0,0,0), r=(0,1,0) )
    attributes: p,q,r the 3 tuples representing points in N-space
    methods:  angleP(), angleR(), angleQ() angles measured in radians
          dPQ(), dPR(), dQR() distances in the same units of p,q,r

    """

    def __init__(self,p,q,r) :
        """ Construct a Triad.

        Example construction:
            p1 = Triad( p=(1.,0.,0.), q=(0.,0.,0.), r=(0.,0.,0.) ).
        """
        self.p = p
        self.q = q
        self.r = r
# private helper methods
    def d2 (self,a,b) : # calculate squared distance of point a to b
        return float(sum((ia-ib)*(ia-ib)  for  ia,ib in zip (a,b)))

    def dot (self,a,b) : # dotProd of standard vectors a,b
        return float(sum(ia*ib for ia,ib in zip(a,b)))

    def ndot (self,a,b,c) : # dotProd of vec. a,c standardized to b
        return float(sum((ia-ib)*(ic-ib) for ia,ib,ic in zip (a,b,c)))

# calculate lengths(distances) of segments PQ, PR and QR
    def dPQ (self):
        """ Provides the distance between point p and point q """
        return math.sqrt(self.d2(self.p,self.q))

    def dPR (self):
        """ Provides the distance between point p and point r """
        return math.sqrt(self.d2(self.p,self.r))

    def dQR (self):
        """ Provides the distance between point q and point r """
        return math.sqrt(self.d2(self.q,self.r))

    def angleP (self) :
        """ Provides the angle made at point p by segments pq and pr (radians). """
        return math.acos(self.ndot(self.q,self.p,self.r) /   math.sqrt(self.d2(self.q,self.p)*self.d2(self.r,self.p)))

    def angleQ (self) :
        """ Provides the angle made at point q by segments qp and qr (radians). """
        return math.acos(self.ndot(self.p,self.q,self.r) /  math.sqrt(self.d2(self.p,self.q)*self.d2(self.r,self.q)))

    def angleR (self) :
        """ Provides the angle made at point r by segments rp and rq (radians). """
        return math.acos(self.ndot(self.p,self.r,self.q) /  math.sqrt(self.d2(self.p,self.r)*self.d2(self.q,self.r)))

def main():
    ''' Function docstring goes here'''
    pass

main()
```

Extra credit (5 points): Rewrite the Triad class.

For extra-credit, provide a direct replacement for the Triad class. The external methods that calculate angles, distances, and points (tuples) p,q and r must be maintained such that either version of the Triad class can be used.

You could use the cosine law to calculate angles instead of the dot product. You might make use of the numpy module. You might recode each of the methods to avoid using zip. You might consider using list iterations. Your Triad replacement must reimplement all of Triad public function, without using zip and without being a trivial rewrite. Your implementation need not be as compact as the current implementation, and it needs to be correct and fully documented to receive full credit.

Triad rewrite

In [ ]:
```python
class Triad:
    def __init__(self,p,q,r) :
        """ Construct a Triad.

        Example object construction:
            p1 = Triad( p=(1.,0.,0.), q=(0.,0.,0.), r=(0.,0.,0.) ).
        """
        self.p = p
        self.q = q
        self.r = r
    def dPQ (self):
        """ Provides the distance between point p and point q """
        pass

    def dPR (self):
        """ Provides the distance between point p and point r """
        pass

    def dQR (self):
        """ Provides the distance between point q and point r """
        pass

    def angleP (self) :
        """ Provides the angle made at point p by segments pq and pr (radians). """
        pass

    def angleQ (self) :
        """ Provides the angle made at point q by segments qp and qr (radians). """
        pass

    def angleR (self) :
        """ Provides the angle made at point r by segments rp and rq (radians). """
        pass
```

Codon tables and amino acid letter converters

In this exercise, you will create a program that uses mappings to convert sequence information between different amino acid representations. This includes the 3-letter codon code (RNA and DNA), the one letter amino acid code and the 3-letter amino acid code. The program will use different dictionaries that represent : codon tables – one for DNA and the other for RNA, and amino acid letter representation converters.

Your task is to create a Python program called converter that asks for and collects a single input string using input() parses the string, looks up the information in the appropriate dictionary, and outputs the correct conversion For example:

if I enter "ATG" (without quotes), then the program will output:
ATG = MET

if I enter "UAG" (without quotes), then the program will output:
UAG = ---
if I enter "E" (without quotes), then the program will output:
E = GLU
if I enter "Asp" (without quotes), then the program will output:
ASP = D

Hints:

- The program might not get a valid codon. In that case, it should output 'unknown'. You can use the dictionary method 'get' and include a default_value to handle the 'unknown' case. See Model p. 72 for an example.

To get full credit on this assignment, your program needs to:

- Run properly (execute and produce the correct output)
- Contain docstrings and line comments (using #)
- Include any assumptions or design decisions you made in writing your code
- Include an overview describing what your program does with expected inputs and outputs as a program level docstring

converter

In [ ]:
```python
#!/usr/bin/env python3
# Name: Your full name (CATS account username)
# Group Members: List full names (CATS usernames) or "None"

'''
Program docstring goes here
'''
short_AA = {
            'CYS': 'C', 'ASP': 'D', 'SER': 'S', 'GLN': 'Q', 'LYS': 'K',
            'ILE': 'I', 'PRO': 'P', 'THR': 'T', 'PHE': 'F', 'ASN': 'N',
            'GLY': 'G', 'HIS': 'H', 'LEU': 'L', 'ARG': 'R', 'TRP': 'W',
            'ALA': 'A', 'VAL': 'V', 'GLU': 'E', 'TYR': 'Y', 'MET': 'M'
            }

long_AA = {value:key for key,value in short_AA.items()}

RNA_codon_table = {
# Second Base
# U             C              A              G
#U
'UUU': 'Phe', 'UCU': 'Ser', 'UAU': 'Tyr', 'UGU': 'Cys',
'UUC': 'Phe', 'UCC': 'Ser', 'UAC': 'Tyr', 'UGC': 'Cys',
'UUA': 'Leu', 'UCA': 'Ser', 'UAA': '---', 'UGA': '---',
'UUG': 'Leu', 'UCG': 'Ser', 'UAG': '---', 'UGG': 'Trp',
#C
'CUU': 'Leu', 'CCU': 'Pro', 'CAU': 'His', 'CGU': 'Arg',
'CUC': 'Leu', 'CCC': 'Pro', 'CAC': 'His', 'CGC': 'Arg',
'CUA': 'Leu', 'CCA': 'Pro', 'CAA': 'Gln', 'CGA': 'Arg',
'CUG': 'Leu', 'CCG': 'Pro', 'CAG': 'Gln', 'CGG': 'Arg',
#A
'AUU': 'Ile', 'ACU': 'Thr', 'AAU': 'Asn', 'AGU': 'Ser',
'AUC': 'Ile', 'ACC': 'Thr', 'AAC': 'Asn', 'AGC': 'Ser',
'AUA': 'Ile', 'ACA': 'Thr', 'AAA': 'Lys', 'AGA': 'Arg',
'AUG': 'Met', 'ACG': 'Thr', 'AAG': 'Lys', 'AGG': 'Arg',
#G
'GUU': 'Val', 'GCU': 'Ala', 'GAU': 'Asp', 'GGU': 'Gly',
'GUC': 'Val', 'GCC': 'Ala', 'GAC': 'Asp', 'GGC': 'Gly',
'GUA': 'Val', 'GCA': 'Ala', 'GAA': 'Glu', 'GGA': 'Gly',
'GUG': 'Val', 'GCG': 'Ala', 'GAG': 'Glu', 'GGG': 'Gly'
}
dnaCodonTable = {key.replace('U','T'):value for key, value in rnaCodonTable.items()}

def main():
    ''' Function docstring goes here'''
    pass

main()
```

Submit your code and answers

Important: please save your work before logging out of the computer. This will ensure that you have a copy of your work and you will avoid having to redo everything in the event that something happens with the lab computers. The two recommended solutions (pick one) at this point are to:

- Email your code files to yourself
- Copy your code files to a pen drive
- Save your work on your own computer if you are using Anaconda (or equiv)

The class will be using canvas to submit the source code files created as part of this lab assignment. For this lab, you should upload the following programs as files:

- seqCleaner.py 9 points
- fastqParse.py 9 points
- coordinateMathSoln.py 9 points
- converter.py 9 points
- triad.py Triad rewrite (optional) 5 points

Important: to get full credit on this lab assignment, each of the code files you submit needs to:

- Run properly (execute and produce the correct output)
- Contain proper docstrings and appropriate line style comments (#)

Congratulations, you finished your second lab assignment!