

Pseudo Code:

bases = a list of the four nucleotide bases found in DNA

self.compDict = a dictionary containing nucleotide basepairs as key:value pairs

self.primerDict = a dictionary with forward and reverse as keys with values initialized at zero

self.dnaMass = a dictionary containing the molecular weight of each nucleotide (the weight of its base pair is included to account for complementary strand

self.REdict = a dictionary containg restriction enzymes and their cutsites as key:value pairs

sequence with unwanted characters (rawSeq) is joined at all whitespaces, and then split into individual characters

self.seq = join rawSeq at all whitespaces and capitalized

cleanSeq = self.seq

for character in cleanSeq:

if character is not a nucleotide:

character is replaced by whitespace

self.seq is saved after removing non-nucleotide characters

Amplicon()

set name forward to value of the key f in self.primerDict

set name reverse to value of the key r in self.primerDict

product = pcr product found in pcrProd method

if there is no product:

print statement notifying using

prompt user if both primers are located on same strand

if yes:

print product

count A, G, C, and T in product

find GC content of product

calculate product molecular weight in kDa

print GC content to output

Protocol()

prompt user to run PCR

if user returns yes:

prompt user for desired reaction volume

while true:

make volume a float

except ValueError:

ask user to input a number

if volume is equal to whitespace or is less than 0:

volume = input(prompt user for a positive number

make volume into a float

print instructions for running PCR line by line so that user must press enter for next step to be printed.

some print statements are dynamic, and output is dependent on inputted value for volume

getProtocol()

numbers = [0-9]

ask user if they want protocol for running gel electrophoresis

if user returns yes:

print introduction to gel electrophoresis

p = input(ask user for percent agarose gel desired)

join p by whitespace, then split into individual characters

join p by whitespace and capatilize

for character in cleaned up p:

if character is not a number 0 - 9:

replace the character with whitespace

make p an integer

set d equal to float of p/10

print protocol for running gel step by step so that user must press enter to proceed to next step

REfinder()

set amp equal to pcr product

ititilize enzyme as an empty dictionary to store specific enzymes found in amplicon

itilitze enzymeList as an empty list

for enzyme name, cut sequence in self.REdict.items():

set cutseq equal to cutsite without | (denotes cleavage site)

if cutSeq is found in amp:

add name:cutsite to enzyme

add name to enzymeList

start position of product = finding position of forward primer in sequence

end position of product = finding position of reverse primer in sequence + lenght of reverse primer

cut sequence = start to stop position in amp

sequence with cutsites = replace cutsequence with \_\_cutsite\_\_

set amp to sequence with cutsites so that each enzyme gets added to sequence

ask user if they want cutsites shown in amplified sequence

if yes:

print sequence with cutsites

print list of enzymes with cutsites in sequence

for key in self.REdict.keys():

if key is in sequence with cutsites:

count each enzyme

print enzyme name and count

if error

print that no cutsites exist in sequence

prompt user to show specific cutsites in amplicon

if yes:

prompt user for desired restriction enzymes

add input to list

make a dictionary copying key:value from REdict with specified restriction enzymes

make copy of pcr product

make another copy of pcr product

for name, cutsite in newly created dictionary:

remove | from cutsite

if cutsite is in amp:

find start position

find stop postion

specificRE = replace start:stop from sequence with cutsite

noName = replace start:stop from sequence with cutsite(no enzyme name)

set amp to specificRE

set amp2 to noName

make list by splitting sequence on |

print expected fragment lengths

loop through fragments

print length of each fragment

error:

print that no fragments are made with specified restriction enzyme

pcrProd()

set forward equal to the value of f in primerdict

set reverse equal to the value or r in primerdict

find start in sequence

find stop in sequence

set product equal to start:stop in sequence

return product

getInfo()

ask for forward primer

add forward primer to dictionary

ask for reverse primer

add reverse primer to dictionary

for primer in primerdict.values:

make primer a string

count A, G, T, C

calculate GC content

calcualte molecular weight

prompt user if primers are from same strand of DNA

if no:

find reverse complement of primer

replace this sequence in primer dictionary

for primer in primerdict.values:

make primer a string

count A, G, T, C

calculate GC content

calcualte molecular weight

print forward primer with statistics

print reverse primer with statistics

print warning if GC content < 50%

main()

Take input from standard in or from a file

if input from standard in:

if input > 0:

make instance of PCR class with sequence

make boolean set to false

while no boolean:

run through methods of PCR class

if amplicon length is 0:

continue

continue running though methods

set p equal to input(run again?)

if yes:

boolean = false

else:

boolean = true

else:

end program

else:

input from file

loop through head and sequence of input file

if input > 0:

make instance of PCR class with sequence

make boolean set to false

while no boolean:

run through methods of PCR class

if amplicon length is 0:

continue

continue running though methods

set p equal to input(run again?)

if yes:

boolean = false

else:

boolean = true