Lecture 11 - String processing.

Let’s start with some data

```
taaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaacccta
accctaaccctaaccctaaccctaacccaaccctaaccctaaccctaaccctaaccctaaccctaaccctaacccctaaccctaac
cctaaccctaaccctaacctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaacccctaaccc
taaccctaaaccctaaaccctaaccctaaccctaaccctaaccctaaccccaaccccaaccccaaccccaaccccaaccc
caaccctaacccctaaccctaaccctaaccctaccctaaccctaaccctaaccctaaccctaaccctaacccctaacccc
taaccctaaccctaaccctaaccctaaccctaaccctaacccctaaccctaaccctaaccctaaccctcgCGGTACCCTC
AGCCGGCCCGCCCGCCCGGGTCTGACCTGAGGAGAACTGTGCTCCGCCTTCAGAGTACCACCGAAATCTGTGCAGAGGAc
aacgcagctccgccctcgcggtGCTCtccgggtctgtgctgaggagaacgCAACTCCGCCGTTGCAAAGGCGcgccgcgc
cggcgcaggcgcagagaggcgcgccgcgccggcgcaggcgcagagaggcgcgccgcgccggcgcaggcgcagagaggcgc
gccgcgccggcgcaggcgcagagaggcgcgccgcgccggcgcaggcgcagagaggcgcgccgcgccggcgcaggcgcaga
```

This is from the file:

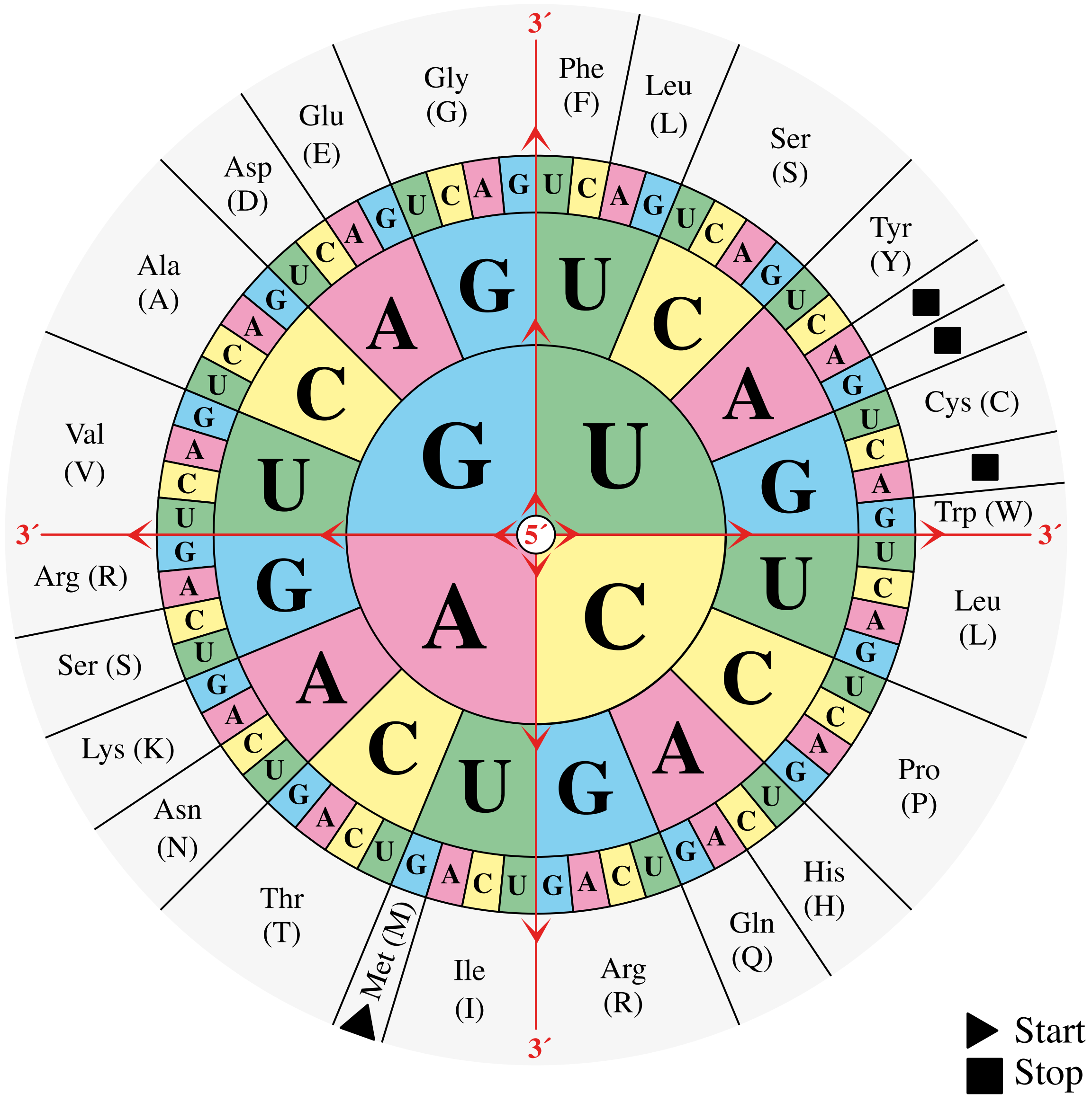
GCF_000001405.39_GRCh38.p41214_genomic.s44a-21.fna

What we want to do is convert this to Amino Acids.

1st thing to note is that the data is upper/lower case. Our life will be simpler if we make it all lower case.

2nd thing is that this is DNA data. It has “T” in it. Let’s make our conversion from messenger RNA that uses U instead of T. So we need a function to start cleaning up the data.

Background on generation of Amino Acids from RNA



Amino Acid Chart

Or if you prefer a table: https://en.wikipedia.org/wiki/DNA_and_RNA_codon_tables

Le's do a google search and you might find:

https://github.com/T101J/Translating_RNA_to_Protein

this is cool. A click on the code, and ... It kind of is what is in the homework ... that has promise. What is the license? OOps! no LICENSE file. Look at the code - they could have put the license in the file. Noope! That means that this is proprietary code - "All Rights Reserved".

Nice chart - but not in the class notes! This is *Prorietary* code and the nice blue chart is also!

The code. Useful to read - but it is not exactly what we want for class. Remember that we want to scan across to find a start-codon, then take 3 at a time and look them up. This assumes that you have already found the start - and just takes 3 at at a time. It has the processing for the stop codon. That is good. So you can't copy this code but you can read it and learn from it.

There are about 5 other places where I found the exact same code on the web. Nice articles like: https://towardsdatascience.com/starting-off-in-bioinformatics-rna-transcription-and-translation-aaa7a91db031 It has a good explanation of how to do this - but not exactly our requirements. (It also has the same chart - from what I can tell the chart is actually copyright a biomedical company.)

Note: going from DAN to RNA change 'T' to 'U'

First convert to one string

We have a list of strings. We need just one. "How do you convert a list of strings to a single string?"

Join them tougher. There is a string operation to do this called join. Join gives us the ability to join with a separator between each string.

```
#!/Users/philip/opt/anaconda3/bin/python

a = [ "aa", "bb", "cc" ]

print ( ", ".join(a) )
```

We can also join with a 0 length separator (this is what we will want in our code).

```
#!/Users/philip/opt/anaconda3/bin/python

a = [ "aa", "bb", "cc" ]

print ( "".join(a) )
```

So we start out with the data in the program:

```
#!/Users/philip/opt/anaconda3/bin/python

dna_input = [
"taaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccta",
"accctaaccctaaccctaaccctaacccaaccctaaccctaaccctaaccctaaccctaaccctaacccctaaccctaac",
"cctaaccctaaccctaacctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaacccctaaccc",
"taaccctaaaccctaaaccctaaccctaaccctaaccctaaccctaaccccaaccccaaccccaaccccaaccccaaccc",
"caaccctaacccctaaccctaaccctaaccctaccctaaccctaaccctaaccctaaccctaaccctaacccctaacccc",
"taaccctaaccctaaccctaaccctaaccctaaccctaacccctaaccctaaccctaaccctaacccaugCGGTACCCTC",
"AGCCGGCCCGCCCGCCCGGGTCTGACCTGAGGAGAACTGTGCTCCGCCTTCAGAGTACCACCGAAATCTGTGCAGAGGAc",
"aacgcagctccgccctcgcggtGCTCtccgggtctgtgctgaggagaacgCAACTCCGCCGTTGCAAAGGCGcgccgcgc",
"cggcgcaggcgcagagaggcgcgccgcgccggcgcaggcgcagagaggcgUGAcgcgccggcgcaggcgcagagaggcgc",
"gccgcgccggcgcaggcgcagagaggcgcgccgcgccggcgcaggcgcagagaggcgcgccgcgccggcgcaggcgcaga",
]

print ( dna_input )
```

And then join it to make 1 string.

How to get to a section of a string

Let’s say we have a string

```
bb = "abc123def"
```

and we want to get to the 123 part of the string. We can pull out just those three characters with:

```
x3 = bb[3:6]
```

So a little program that will do this is:

```
#!/Users/philip/opt/anaconda3/bin/python

bb = "abc123def"
x3 = bb[3:6]
print ( "x3 = {}".format(x3) )
```

Replace - modifying a string

Python has a builtin string function to replace strings.

```
s = "aBBcBBd"
t = s.replace("B","-")
print ( t )
```

Use replace to remove a string.

```
# Remove Bs

s = "aBBcBBd"
t = s.replace("B","")
print ( t )
```

replace with a longer string

```
# Replace Bs with a longer string.

s = "aBBcBBd"
t = s.replace("B","--hi--")
print ( t )
```

Replace just the 1st occurrence

```
# Replace just the 1st B with a longer string.

s = "aBBcBBd"
t = s.replace("B","--hi--",1)
print ( t )
```

Convert to upper/lower case

To convert to lower case

```
a = "This is A Mixed Case"
b = a.upper()
print ( b )
```

To convert to upper case

Functions for Lab 05

First is a function to take our input, convert from upper/lower case to lower, then replace the `t` s with `u` s.

Second is to use a dictionary lookup.

To create a dictionary in Python

```
d = { "name": 1, "name2", 2 }
```

or to lookup other strings

```
d = { "in1": "out1", "in2", "out2" }
```

So we can lookup our 3 letter codon with a dictionary.

```
#!/Users/philip/opt/anaconda3/bin/python

# RNA codon table
rna_codon = {
    "aaa" : "K", "aac" : "N", "aag" : "K", "aau" : "N", "aca" : "T", "acc" : "T", "acg"
    "acu" : "T", "aga" : "R", "agc" : "S", "agg" : "R", "agu" : "S", "aua" : "I", "auc"
    "aug" : "M", "auu" : "I", "caa" : "Q", "cac" : "H", "cag" : "Q", "cau" : "H", "cca"
    "ccc" : "P", "ccg" : "P", "ccu" : "P", "cga" : "R", "cgc" : "R", "cgg" : "R", "cgu"
    "cua" : "L", "cuc" : "L", "cug" : "L", "cuu" : "L", "gaa" : "E", "gac" : "D", "gag"
    "gau" : "D", "gca" : "A", "gcc" : "A", "gcg" : "A", "gcu" : "A", "gga" : "G", "ggc"
    "ggg" : "G", "ggu" : "G", "gua" : "V", "guc" : "V", "gug" : "V", "guu" : "V", "uaa"
    "uac" : "Y", "uag" : ".", "uau" : "Y", "uca" : "S", "ucc" : "S", "ucg" : "S", "ucu"
    "uga" : ".", "ugc" : "C", "ugg" : "W", "ugu" : "C", "uua" : "L", "uuc" : "F", "uug"
    "uuu" : "F"
}
```

We can see if a value is in the lookup table with an “if”

```
from rna_lookup import rna_codon

if "xyz" in rna_codon:
   print ( "found" )
else:
   print ( "not found" )

if "gac" in rna_codon:
   print ( "found value = {}".format(rna_codon["gac"] ) )
else:
   print ( "not found" )
```

A Program to go from DNA to Amino Acids.

Program State

```
#!/Users/philip/opt/anaconda3/bin/python

import conv_t_to_u
import rna_lookup

dna_input = [
"taaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccta"
"accctaaccctaaccctaaccctaacccaaccctaaccctaaccctaaccctaaccctaaccctaacccctaaccctaac"
"cctaaccctaaccctaacctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaaccctaacccctaaccc"
"taaccctaaaccctaaaccctaaccctaaccctaaccctaaccctaaccccaaccccaaccccaaccccaaccccaaccc"
"caaccctaacccctaaccctaaccctaaccctaccctaaccctaaccctaaccctaaccctaaccctaacccctaacccc"
"taaccctaaccctaaccctaaccctaaccctaaccctaacccctaaccctaaccctaaccctaacccaugCGGTACCCTC"
"AGCCGGCCCGCCCGCCCGGGTCTGACCTGAGGAGAACTGTGCTCCGCCTTCAGAGTACCACCGAAATCTGTGCAGAGGAc"
"aacgcagctccgccctcgcggtGCTCtccgggtctgtgctgaggagaacgCAACTCCGCCGTTGCAAAGGCGcgccgcgc"
"cggcgcaggcgcagagaggcgcgccgcgccggcgcaggcgcagagaggcgUGAcgcgccggcgcaggcgcagagaggcgc"
"gccgcgccggcgcaggcgcagagaggcgcgccgcgccggcgcaggcgcagagaggcgcgccgcgccggcgcaggcgcaga"
]

amino_string = ""

rna = conv_t_to_u.conv_t_to_u ( "".join(dna_input) )

st = "before"
i = 0
while i < len(rna)-2:
    three = rna[i:i+3]
    # print ( "st {} three >{}< at i={}".format(st, three,i))
    amino = rna_lookup.rna_to_amino_acid(three)
    if amino == '!' :
        print ( "Invalid 3 letter sequence {} at {}".format(three, i ) )
        break
    if st == "before" and amino == "M" :
        st = "encode"
        i = i + 3
    elif st == "encode" and amino == ".":
        st = "before"
        print ("Protein : ", amino_string)
        i = i + 3
    elif st == "encode" :
        amino_string = amino_string + amino
        i = i + 3
    else:
        i = i + 1

```

Copyright

Copyright © University of Wyoming, 2021.