Day 3: Control flow Practicals

Let's get down to business and start typing some real code. In order to solve the proposed exercises you are required to use the provided examples.

In [1]:
```
from data import cytb, translations, acidAA, IUPAC_codes
```

In this module you will find 3 things:

- examples.sequences
- examples.translations
- examples.acid-aa

Just print() any one to see what's inside each of them.

Problem 1

Verify that there are no illegal characters in any of the sequences in *cytb*.

Tip:
Use the *IUPAC_codes* list that was imported from the data module to check if the sequence's nucleotides are legal.

In [3]:
```
for names,seqs in cytb.items():
    for chars in seqs:
        if chars not in IUPAC_codes:
            print("Character %s in sequence %s is invalid. Please verify this..." %(chars, \
seqs))
            break
    else:
        print("No invalid caharcters found in %s. Good!" %(names))
```

```
No invalid caharcters found in Mo12. Good!
No invalid caharcters found in Mo11. Good!
No invalid caharcters found in Mo10. Good!
No invalid caharcters found in Ib8. Good!
No invalid caharcters found in Ib9. Good!
No invalid caharcters found in Ib16. Good!
No invalid caharcters found in Ib17. Good!
No invalid caharcters found in Ib15. Good!
No invalid caharcters found in Ib1. Good!
No invalid caharcters found in Ib2. Good!
No invalid caharcters found in Ib3. Good!
No invalid caharcters found in Ib4. Good!
No invalid caharcters found in Ib5. Good!
No invalid caharcters found in Ib6. Good!
No invalid caharcters found in Ib7. Good!
No invalid caharcters found in Mo9. Good!
No invalid caharcters found in Mo8. Good!
No invalid caharcters found in Outgroup. Good!
No invalid caharcters found in Mo3. Good!
No invalid caharcters found in Mo2. Good!
No invalid caharcters found in Mo1. Good!
No invalid caharcters found in Mo7. Good!
No invalid caharcters found in Mo6. Good!
No invalid caharcters found in Mo5. Good!
No invalid caharcters found in Mo4. Good!
No invalid caharcters found in Ib18. Good!
No invalid caharcters found in Ib19. Good!
No invalid caharcters found in Ib14. Good!
No invalid caharcters found in Ib11. Good!
No invalid caharcters found in Ib12. Good!
No invalid caharcters found in Ib13. Good!
No invalid caharcters found in Ib10. Good!
No invalid caharcters found in Ib24. Good!
No invalid caharcters found in Ib23. Good!
No invalid caharcters found in Ib22. Good!
No invalid caharcters found in Ib21. Good!
No invalid caharcters found in Ib20. Good!
```

Problem 2:

Translate the sequences in *cytb* from nucleotides into aminoacids.

Tips:
Use the translations dictionary from the data module (*translations*);
Use a *for* loop to iterate through the sequences in *cytb*;
Translating the sequences into a list of aminoacids rather than a string will save you work later on.

In [5]:
```
cytbAA = {}

for names,seqs in cytb.items():
    AAlist = []
    for i in range(3,len(seqs),3):
        codon = seqs[i-3:i]
        if codon in translations:
            AAlist.append(translations[seqs[i-3:i]])
        else:
            AAlist.append("N/A")
    cytbAA[names] = AAlist
```

Problem 3:

Reverse and complement the sequences in *cytb*.

Tip:
Use slicing to reverse the sequence.

In [6]:
```
cytbREV = {}

for names,seqs in cytb.items():
    LOWseq = seqs.lower()
    LOWseq = LOWseq.replace("a","T")
    LOWseq = LOWseq.replace("c","G")
    LOWseq = LOWseq.replace("g","C")
    LOWseq = LOWseq.replace("t","A")
    LOWseq = LOWseq.replace("n","N")
    LOWseq = LOWseq.replace("r","Y")
    LOWseq = LOWseq.replace("y","R")
    LOWseq = LOWseq.replace("k","M")
    LOWseq = LOWseq.replace("m","K")
    LOWseq = LOWseq.replace("s","W")
    LOWseq = LOWseq.replace("W","s")
    LOWseq = LOWseq.replace("b","V")
    LOWseq = LOWseq.replace("v","B")
    LOWseq = LOWseq.replace("d","H")
    LOWseq = LOWseq.replace("h","D")
    revSEQ = LOWseq
    cytbREV[names] = revSEQ[::-1]
```

Problem 4:

a) Find the aminoacids in acidAA in the sequences from cytb. Return the positions of both the amino acid (in the protein sequence) and the codon position (in the nucleotide sequence). number.

Tip:
Remember that an acid aminoacid may be present more than once.

In [7]:
```
cytbAcidAA = {}

for names,AAs in cytbAA.items():
    position = 0
    found = []
    nucleotides = []
    for aa in AAs:
        position += 1
        if aa in acidAA:
            found.append(position)
            nucleotide_position = position * 3
            codon = cytb[names][nucleotide_position:nucleotide_position+3]
            nucleotides.append(nucleotide_position)
    if found != []:
        print("Acid aminoacid found in %s, in position(s) %s of the translated \
sequence." %(names, found))
        print("This is represented by the codons in position(s) %s of the original \
sequence.\n" %(nucleotides))
    else:
        print("Acid aminoacid not found in %s" %(names))
    cytbAcidAA[names] = found
```

```
Acid aminoacid found in Ib8, in position(s) [21, 35, 74, 125, 134, 165, 179, 191, 215, 217] of
This is represented by the codons in position(s) [63, 105, 222, 375, 402, 495, 537, 573, 645, 6

Acid aminoacid found in Ib9, in position(s) [21, 35, 74, 125, 134, 165, 179, 191, 215, 217] of
This is represented by the codons in position(s) [63, 105, 222, 375, 402, 495, 537, 573, 645, 6

Acid aminoacid found in Ib1, in position(s) [21, 35, 74, 125, 134, 165, 179, 191, 215, 217] of
This is represented by the codons in position(s) [63, 105, 222, 375, 402, 495, 537, 573, 645, 6

Acid aminoacid found in Ib2, in position(s) [21, 35, 74, 125, 134, 165, 179, 191, 215, 217] of
This is represented by the codons in position(s) [63, 105, 222, 375, 402, 495, 537, 573, 645, 6

Acid aminoacid found in Ib3, in position(s) [21, 35, 74, 125, 134, 165, 179, 191, 215, 217] of
This is represented by the codons in position(s) [63, 105, 222, 375, 402, 495, 537, 573, 645, 6

Acid aminoacid found in Ib4, in position(s) [21, 35, 74, 125, 134, 165, 179, 191, 215, 217] of
This is represented by the codons in position(s) [63, 105, 222, 375, 402, 495, 537, 573, 645, 6

Acid aminoacid found in Ib5, in position(s) [21, 35, 74, 125, 134, 165, 179, 191, 215, 217] of
This is represented by the codons in position(s) [63, 105, 222, 375, 402, 495, 537, 573, 645, 6

Acid aminoacid found in Ib6, in position(s) [21, 35, 74, 125, 134, 165, 179, 191, 215, 217] of
This is represented by the codons in position(s) [63, 105, 222, 375, 402, 495, 537, 573, 645, 6

Acid aminoacid found in Ib7, in position(s) [21, 35, 74, 125, 134, 165, 179, 191, 215, 217] of
This is represented by the codons in position(s) [63, 105, 222, 375, 402, 495, 537, 573, 645, 6

Acid aminoacid found in Outgroup, in position(s) [21, 35, 74, 125, 134, 165, 179, 191, 215, 217
This is represented by the codons in position(s) [63, 105, 222, 375, 402, 495, 537, 573, 645, 6

Acid aminoacid found in Ib24, in position(s) [21, 35, 74, 125, 134, 165, 179, 191, 215, 217] of
This is represented by the codons in position(s) [63, 105, 222, 375, 402, 495, 537, 573, 645, 6

Acid aminoacid found in Ib23, in position(s) [21, 35, 74, 125, 134, 165, 179, 191, 215, 217] of
This is represented by the codons in position(s) [63, 105, 222, 375, 402, 495, 537, 573, 645, 6

Acid aminoacid found in Ib22, in position(s) [21, 35, 74, 125, 134, 165, 179, 191, 215, 217] of
This is represented by the codons in position(s) [63, 105, 222, 375, 402, 495, 537, 573, 645, 6

Acid aminoacid found in Ib21, in position(s) [21, 35, 74, 125, 134, 165, 179, 191, 215, 217] of
This is represented by the codons in position(s) [63, 105, 222, 375, 402, 495, 537, 573, 645, 6

Acid aminoacid found in Ib20, in position(s) [21, 35, 74, 125, 134, 165, 179, 191, 215, 217] of
This is represented by the codons in position(s) [63, 105, 222, 375, 402, 495, 537, 573, 645, 6

Acid aminoacid found in Mo12, in position(s) [21, 35, 74, 125, 134, 165, 179, 191, 215, 217] of
This is represented by the codons in position(s) [63, 105, 222, 375, 402, 495, 537, 573, 645, 6

Acid aminoacid found in Mo11, in position(s) [21, 35, 74, 125, 134, 165, 179, 191, 215, 217] of
This is represented by the codons in position(s) [63, 105, 222, 375, 402, 495, 537, 573, 645, 6
```

b) How frequent are the acid amino acids? Are they more or less frequent than what you would expect under a random distribution pattern (i.e., where every amino acid has the same probability of occurring)?

In [10]:
```
for names,acids in cytbAcidAA.items():
    frequency = len(acids) / float(len(cytbAA[names]))
    print(frequency)
```

```
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
0.0456621004566
```

c) Return the sequence's nucleotides from each sequence in cytb until any of the acid amino acids from acidAA is found; Return "Target aminoacids not found" if neither of them is present in the sequence.

Tip:
A while loop is the simplest choice here, but feel free to use any other way

In [58]:
```
acid_translations = { 'GAT': 'Asp', 'GAC': 'Asp', 'GAA': 'Glu', 'GAG': 'Glu'}

for names,seqs in cytb.items():
    codon = ""
    index = 3
    while codon not in acid_translations:
        codon = seqs[index-3:index]
        index +=3
    else:
        short_sequence = seqs[:index]
        print(short_sequence)
```

```
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNCAGATATT
GGATTGTGCCTAATTACTCAAATTGTCACAGGGTTATTTTTAGCAATACACTACAATGCAGATATT
GGACTGTGCCTAATTACTCAAATTGTTACAGGGTTATTTTTAGCAATACACTACAATGCAGATATT
GGATTGTGCCTAATTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAATGCAGATATT
GGATTGTGCCTAATTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAATGCAGATATT
GGATTATGCCTAATTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAATGCAGATATT
GGATTGTGCCTAATTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAATGCAGATATT
GGATTGTGCCTAATTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAATGCAGATATT
GGATTGTGCCTAATTACTCAAATTGTTACCGGATTATTTTTAGCAATACACTATAATGCAGATATT
GGATTGTGCCTAATTACTCAAATTGTTACCGGATTATTTTTAGCAATACACTATAATGCAGATATT
GGATTGTGCCTAATTACTCAAATTGTTACCGGACTATTTTTAGCAATACACTATAATGCAGATATT
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNACTATTTTTAGCAATACACTATAATGCAGATATT
GGATTGTGCCTAATTACTCAAATTGTTACCGGACTATTTTTAGCAATACACTATAATGCAGATATT
GGATTGTGCCTAATTACTCAAATTGTTACCGGACTATTTTTAGCAATACACTATAATGCAGATATT
GGACTATGCCTAATTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAATGCAGATATT
GGACTATGCCTAATCACTCAAATTGTTACAGGATTGTTTTTAGCAATACACTACAATGCGGATATT
GGCCTCTGTTTAATTATTCAAACCGTTACAGGCCTTTTTCTAGCCATACACTACACCCCAGACATC
GGATTATGCCTAATTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAACGCAGATATT
GGATTATGCCTAATTACTCAAATTGTTACAGGGTTATTTTTAGCAATACACTACAACGCAGATATT
GGATTATGCCTAATTACTCAAATTGTTACAGGGTTATTTTTAGCAATACACTACAACGCAGATATT
GGACTATGCCTAATCACTCAAATTGTTACAGGATTGTTTTTAGCAATACACTACAATGCAGATATT
GGACTATGCCTAATCACTCAAATTGTTACAGGATTGTTTTTAGCAATACACTACAATGCAGATATT
GGACTATGCCTAATCACTCAAATTGTTACAGGATTGTTTTTAGCAATACACTACAATGCAGATATT
GGATTATGCCTAATTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAACGCAGATATT
GGATTATGCCTAATTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAATGCAGATATT
GGATTATGCCTAATTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAATGCAGATATT
GGATTGTGCCTAATTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAATGCAGATATT
GGATTGTGCCTAATTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAATGCAGATATT
GGATTGTGCCTAATTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAATGCAGATATT
GGATTGTGCCTAATTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAATGCAGATATT
GGATTGTGCCTAATTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAATGCAGATATT
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNGGATTATTTTTAGCAATACACTACAATGCAGATATT
GGATTATGCCTAATTACTCAAATTGTTACAGGGTTATTTTTAGCAATACACTACAACGCAGATATT
GGATTGTGCCTAATTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAATGCAGATATT
GGATTATGCCTAGTTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAATGCAGATATT
GGAATATGCCTAATTACTCAAATTGTTACAGGATTATTTTTAGCAATACACTACAATGCAGATATT
```

Problem 5:

Find all the equal sequences in *cytb* and "collapse" them into a single sequence (with the name of all the collapsed sequences).

In [13]:
```
collapsed_cytb = {}

for names,seqs in cytb.items():
    if seqs not in collapsed_cytb.values():
        collapsed_cytb[names] = seqs
    else:
        for k,v in collapsed_cytb.items():
            if v == seqs:
                merged_names = k + ";" + names
                break
        collapsed_cytb[merged_names] = collapsed_cytb.pop(k)

for i in collapsed_cytb:
    print(i)
```

```
Ib8;Ib9;Ib14;Ib11;Ib12;Ib13;Ib10
Ib1
Ib2
Ib3
Ib4
Ib5
Ib6
Ib7
Outgroup
Mo6;Mo5
Ib24
Ib23
Ib22
Ib21
Ib20
Mo12
Mo11
Mo10
Mo9
Mo8
Mo3
Mo2
Mo1
Mo7
Mo4
Ib18
Ib19
Ib16
Ib17
Ib15
```