Homework 3 - Python data types

BIOINF 575 - Fall 2021

Total 40 points

For each problem part provide the solution in Code cells after the description of the problem part. Answers to questions should be written either as comments together with the code or in Markdown cell(s) for each part of the problem.

In [23]:
```
#first identify the in depth key value from the info provided.
#then split the line given, and then take the last element, then split the second eleme
#code for all lines, not just this line.

#SHould get a True or False answer for part 2.
```

Problem 1 - Extract the depth of coverage for a variant

20 points

.....

Part 1 (15 points)

Given the following line (tab-separated) from a vcf file, write code to extract the total depth value.

```
variant = "Y    2728456 rs2058276   T   C   32  .
AC=2;AN=2;DB;DP=182;H2;NS=65"
```

Below is the header of the vcf file explaining each element on the line above.

```
##fileformat=VCFv4.0
##fileDate=20100610
##source=glfTools v3
##reference=1000GenomesPilot-NCBI36
##phasing=NA
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With
Mapped Reads">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build
129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=NUYR,Description="Variant in non-unique Y region">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Depth">
##INFO=<ID=AC,Number=.,Type=Integer,Description="Allele count in
genotypes">
##INFO=<ID=AN,Number=1,Type=Integer,Description="Total number of alleles
in called genotypes">
#CHROM  POS ID  REF ALT QUAL    FILTER  INFO
Y   2728456 rs2058276   T   C   32  .   AC=2;AN=2;DB;DP=182;H2;NS=65
```

~~~~~~

One suggestion would be to break down the string into smaller parts and then look for the element of interest.

- Score breakdown - these are not ordered steps of an algorithm but just elements of the solution to focus on:
  - **(0.5 points)** Correctly identifying the element you are looking to extract
  - **(4 points)** Reducig the scope of the search at least twice
  - **(4.5 points)** Delimiting the reduced areas correctly and/or using additional data structures correctly
  - **(4 points)** Making correct use of the methods/operators available for the datatype you are working with
  - **(2 points)** Delimiting the result correctly and displaying the number for total depth

In [22]:
```
#Write your solution here, feel free to add new cells.
#One suggestion would be to break down the string into smaller parts and then look for

#I want to extract the total depth value, which is 182 from the INFO column.
#I reduced the scope of the search by a total of three times by first splitting the lis
#element in that list, and finally splitting again to extract the exact value I desired

variant = "Y    2728456    rs2058276    T    C    32    .    AC=2;AN=2;DB;DP=182;H2;NS=
variant_list = variant.split("    ")
#I first tried splitting by the "\t" but that was not working, so I tried four spaces "
#Printing variant_list will give me a new list where each element separated by a tab wi
variant_info_list = variant_list[7].split(";")
#Printing variant_info_list will give me specifically the 8th string element in the var
#wherever there was a ";" and creating another list were those elements become separate
variant_info_DP = variant_info_list[3].split("=")
#Printing variant_info_DP will give me the fourth string element in variant_info_DP and
#where there is an "=" into two separate string values.
value_DP = variant_info_DP[1]
#I am taking the second index of this specific list to get the numerical value of DP.
print(value_DP)
```

```
182
```

.....

Part 2 (5 points)

For the line mentioned at part 1 extract the value for number of samples with mapped reads and the quality (QUAL column) and
check if the total depth > 100 and the number of samples with mapped reads > 50 and the quality is above 20 (means < 1\% error rate).

- **Score breakdown - these are not ordered steps of an algorithm but just elements of the solution to focus on:**
  - **(0.5 points) Correctly identifying the element you are looking to extract**
  - **(3 points) Correctly applying the code from part 1 to extract the number of samples with mapped reads**
~~~~~~

- **(1.5 points) Correctly extracting the QUAL value**
- **(0.5 point) Correctly checking if the required conditions are met or not and displaying the result**

In [24]:
```
#Write your solution here, feel free to add new cells.

#I want to extract two values: the NS (number of samples) = 65 and the quality value (Q
#Below is the code to extract the value for NS.

variant = "Y    2728456    rs2058276    T    C    32    .    AC=2;AN=2;DB;DP=182;H2;NS=
variant_list = variant.split("    ")
variant_info_list = variant_list[7].split(";")
variant_info_NS = variant_info_list[5].split("=")
value_NS = variant_info_NS[1]
print(value_NS)
```

```
65
```

In [25]:
```
#Below is the code to extract the value for QUAL.

variant = "Y    2728456    rs2058276    T    C    32    .    AC=2;AN=2;DB;DP=182;H2;NS=
variant_list = variant.split("    ")
value_QUAL = variant_list[5]
print(value_QUAL)
```

```
32
```

In [29]:
```
#Here I use if loops to check if the required conditions are met or not and displaying

if int(value_DP) > 100:
    print(True)
    print(value_DP)
else:
    print(False)

if int(value_NS) > 50:
    print(True)
    print(value_NS)
else:
    print(False)

if int(value_QUAL) > 20:
    print(True)
    print(value_QUAL)
else:
    print(False)
```

```
True
182
True
65
True
32
```

Problem 2 - Sequence analysis

20 points

Part 1 (10 points)

We have the following list with sequences:

```
sequences = ["AAACCTTG", "TTCAG", "GGATCTT", "AAC", "CCTTGG", "CATG", "AAA",
"GGTTTAACA", "GGGTACGT", "CCGCATCAGACCT", "AACA", "TTGAATCC", "TAAGA",
"AAATTCCGGAAA"]
```

Create a dictionary with sequences that have GC content > 40%.
The keys will be some of "Sequence 1", "Sequence 2", ... "Sequence n",
where 1,2,...n, is the position (1-based) of the sequence in the sequences list.
The value for each key is a tuple.
The tuple has two elements: the GC content and the sequence.
E.g.: If we had only one sequence, `"ACGT"` , the result would be:

```
{"Sequence 1": (0.5, "ACGT") }
```

- **Score breakdown - these are not ordered steps of an algorithm but just elements of the solution to focus on:**
  - **(1.5 points) Correctly using the for loop**
  - **(1.5 points) Correctly computing the GC content for each sequence**
  - **(3 points) Correctly building the tuple**
  - **(4 points) Correctly building and displaying the dictionary, including its keys**

In [33]:
```
#Write your solution here, feel free to add new cells.

sequences = ["AAACCTTG", "TTCAG", "GGATCTT", "AAC", "CCTTGG", "CATG", "AAA", "GGTTTAACA
seq_GC_content_map = {}
i = 0
for seq in sequences:
    math_seq = (seq.count("G") + seq.count("C"))/(len(seq))
    if math_seq > 0.4:
        math_answer = (math_seq, seq)
        i += 1
        seq_GC_content_map["Sequence " + str(i)] = math_answer
print(seq_GC_content_map)
```

```
{'Sequence 1': (0.42857142857142855, 'GGATCTT'), 'Sequence 2': (0.6666666666666666, 'CCT
TGG'), 'Sequence 3': (0.5, 'CATG'), 'Sequence 4': (0.625, 'GGGTACGT'), 'Sequence 5': (0.
6153846153846154, 'CCGCATCAGACCT')}
```

Part 2 (10 points) - Understand the code

Explain what the following code does and how it computes the result it displays.

```
seq_list = ["AAACCTTGA", "AAATTCCGGAAA"]
seq_codon_map = {}
for i, seq in enumerate(seq_list):
    n = len(seq)
```

```
    codon_list = []
    for j in range(0, n - 2, 3):
        codon = seq[j:j+3]
        codon_list.append(codon)
    seq_codon_map[i+1] = (seq, codon_list)

print(seq_codon_map)
```

- **Score breakdown for the 10 lines of code:**
  - **(2 points each) Lines 3 and 6 - the ones with the for**
  - **(0.5 points each) Lines 1, 2, 4, 5, 8, 10**
  - **(1.5 points each) Lines 7 and 9**

~~~~~

Write your solution here and then run the cell (Markdown)

double click here to edit the cell _

In [35]:

```
#Copy the code here and try to break it down to understand it - feel free to add new ce
seq_list = ["AAACCTTGA", "AAATTCCGGAAA"]
seq_codon_map = {}
for i, seq in enumerate(seq_list):
    n = len(seq)
    codon_list = []
    for j in range(0, n - 2, 3):
        codon = seq[j:j+3]
        codon_list.append(codon)
    seq_codon_map[i+1] = (seq, codon_list)

print(seq_codon_map)

#Line 1: You are assigning a list with two string values to the variable called "seq_li

#Line 2: You are creating an empty dictionary and assigning it to the variable called "

#Line 3: You are beginning a for loop, where your variabless called "i" and "seq" are u
#the list called "seq_list". Right before the iterable, you have "enumerate" which allo
#iterations have occured.

#Line 4: You are assigning another new variable called "n" to the length of your variab

#Line 5: You are creating an empty list and assigning it to the variable called "codon_

#Line 6: You are beginning your second for loop within the first for loop (nesting for
#from 0, which is your starting point, to n-2 which is your stopping point in the range

#Line 7: You are setting another new variable called "codon" equal to the task of index
#the beginning of the first nucleotide string to the third nucleotide, outputs those th
#to the fourth nucleotide in the string and repeats the for loop, counting again from o
#telling the computer to start from "j" and go to "j + 3", indicating you want three to
#Line 6 the same but changed line 7 to "codon = seq[j:j+4]". Here, your output will be
#because you never changed line 6, in order to create the output, the computer will sti
#nucleotide the computer begins the count with will be in step of 3 in the original nuc

#Line 8: You are appending or adding the result of variable "codon" to the end of "codo

#Line 9: You are taking your empty dictionary you created and named "seq_condon_map" in
```
~~~~~

```
#In this particular case, you are assigning your keys as an index that is starting from
#your values in the dictionary with two items: the seq variable which will give you you
#will give you the list of your string combinations of three nucleotides taken from the

#Line 10: You are printing your seq_codon_map dictionary.
```

```
{1: ('AAACCTTGA', ['AAA', 'CCT', 'TGA']), 2: ('AAATTCCGGAAA', ['AAA', 'TTC', 'CGG', 'AA
A'])}
```
