Assignment2

October 21, 2020

```
[ ]: #Q1
```

```
[72]: def GC_content(**kwargs): #define function with passing a string as **kwargs
          for key in kwargs: #loop through key in dictionaries
              for nt in kwargs[key].upper(): #loop nucleotides in each dictionary␣
       ↪values
                  if nt not in ["A", "T", "C", "G"]: #detect seq if non nt present,␣
       ↪returning T or F
                      print("%s has non nucleotide characters." % (key))  #print out␣
       ↪the sentence detailing the variable name
                      break #stop looping for the next loop if that sequence of that␣
       ↪variable contains non nt
              if len(kwargs[key]) < 30: #loop through dictionary values and detect if␣
       ↪its length less than 30
                  continue #skip the sequence with length less than 30
              else:
                  if len(kwargs[key]) > 30 and nt in ["A", "T", "C", "G"]: #loop␣
       ↪through dictionary values and only returns with length of values larger than␣
       ↪30 and containing only nt
                      length=len(kwargs[key]) #calculate length of seq
                      g_content = kwargs[key].upper().count('G') #calculate G counts
                      c_content = kwargs[key].upper().count('C') #calculate C counts
                      gc_content = (100*(g_content + c_content))/ length #calculate␣
       ↪GC content
                      round_gc_content = round(gc_content, 1) #round GC content to␣
       ↪one sig fig
                      if round_gc_content > 65: #only return out GC content larger␣
       ↪than 65
                          print ("%s has a high GC content of %s and sequence length␣
       ↪of %s nucleotides." % (key, str(round_gc_content), str(length))) #print out␣
       ↪sentence detailing the variable name, its GC content, and its length
                      elif round_gc_content < 45: #only return out GC content smaller␣
       ↪than 45
                          print ("%s has a low GC content of %s and sequence length␣
       ↪of %s nucleotides." % (key, str(round_gc_content), str(length))) #print out␣
       ↪sentence detailing the variable name, its GC content, and its length
                      else: #only return out GC content in between
```

```
                    print ("%s has a medium GC content of %s and sequence␣
↪length of %s nucleotides." % (key, str(round_gc_content), str(length)))␣
↪#print out sentence detailing the variable name, its GC content, and its␣
↪length
```

[73]:
```
GC_content(SingleSeq =␣
↪"""tccgtgaaacaaagcggatgtaccggatrhptattccggctatggggcaattcgcggatttactcaggggagagcccagataaatgga
↪a string as key-word arg
```

```
SingleSeq has non nucleotide characters.
```

[74]:
```
#Q2
```

[75]:
```
def GC_content(**kwargs): #define function with passing 3 strinfgs as **kwargs
    for key in kwargs: #loop through key in 3 dictionaries
        for nt in kwargs[key].upper(): #loop nucleotides in each dictionary␣
↪values
            if nt not in ["A", "T", "C", "G"]: #detect seq if non nt present,␣
↪returning T or F
                print("%s has non nucleotide characters." % (key)) #print out␣
↪the sentence detailing the variable name
                break #stop looping for the next loop if that sequence of that␣
↪variable contains non nt
        if len(kwargs[key]) < 30: #loop through dictionary values and detect if␣
↪its length less than 30
            continue #skip the sequence with length less than 30
        else:
            if len(kwargs[key]) > 30 and nt in ["A", "T", "C", "G"]: #loop␣
↪through dictionary values and only returns with length of values larger than␣
↪30 and containing only nt
                length=len(kwargs[key]) #calculate length of seq
                g_content = kwargs[key].upper().count('G') #calculate G counts
                c_content = kwargs[key].upper().count('C') #calculate C counts
                gc_content = (100*(g_content + c_content))/ length #calculate␣
↪GC content
                round_gc_content = round(gc_content, 1) #round GC content to␣
↪one sig fig
                if round_gc_content > 65: # only return out GC content larger␣
↪than 65
                    print ("%s has a high GC content of %s and sequence length␣
↪of %s nucleotides." % (key, str(round_gc_content), str(length))) #print out␣
↪sentence detailing the variable name, its GC content, and its length
                elif round_gc_content < 45: #only return GC content smaller␣
↪than 45
```

```
                    print ("%s has a low GC content of %s and sequence length␣
↪of %s nucleotides." % (key, str(round_gc_content), str(length))) #print out␣
↪sentence detailing the variable name, its GC content, and its length
                else: #only return GC content in between
                    print ("%s has a medium GC content of %s and sequence␣
↪length of %s nucleotides." % (key, str(round_gc_content), str(length)))␣
↪#print out sentence detailing the variable name, its GC content, and its␣
↪length
```

[76]:
```
GC_content(B69116 =␣
↪"caccaataaaaaaacaagcttaacctaattccggccagatcbstaccagataaatggaagcttagatctggccgggg",␣
↪B557211 =␣
↪"""gcggatttactcaggggagagcccagataaatggagtctgtgcgtccacagaattcgcaccaagcttagatctggccgggg""",␣
↪AX557349 = "gcggatttactcaggggagagcccagataaatggagtctgtgcgtccacagaattcgcacca")␣
↪#passing 3 strings into function as key-word args
```

```
B69116 has non nucleotide characters.
B557211 has a medium GC content of 55.6 and sequence length of 81 nucleotides.
AX557349 has a medium GC content of 53.2 and sequence length of 62 nucleotides.
```

[77]:
```
#Q3
```

[78]:
```
def GC_content(SeqList): #define function with passing a list
    for i in range(4): #loop through list indices
        for key in SeqList[i].keys(): #loop through keys in key-word␣
↪args(dictionaries)
            for nt in SeqList[i][key].upper(): #loop through nucleotides in␣
↪each dictionary values
                if nt not in ["A", "T", "C", "G"]: #detect seq if non nt␣
↪present, returning T or F
                    print("%s has non nucleotide characters." % (key)) #print␣
↪out the sentence detailing the variable name
                    break #stop looping for the next loop if that sequence of␣
↪that variable contains non nt
            if len(SeqList[i][key]) < 30: #loop through dictionary values and␣
↪detect if its length less than 30
                continue #skip the sequence with length less than 30
            else:
                if len(SeqList[i][key]) > 30 and nt in ["A", "T", "C", "G"]:␣
↪#loop through dictionary values and only returns with length of values␣
↪larger than 30 and containing only nt
                    length=len(SeqList[i][key]) #calculate length of seq
                    g_content = SeqList[i][key].upper().count('G') #calculate G␣
↪counts
                    c_content = SeqList[i][key].upper().count('C') #calculate C␣
↪counts
```

```
                    gc_content = (100*(g_content + c_content))/ length␣
↪#calculate GC content
                    round_gc_content = round(gc_content, 1) #round GC content␣
↪to one sig fig
                    if round_gc_content > 65: # only return out GC content␣
↪larger than 65
                        print ("%s has a high GC content of %s and sequence␣
↪length of %s nucleotides." % (key, str(round_gc_content), str(length)))␣
↪#print out sentence detailing the variable name, its GC content, and its␣
↪length
                    elif round_gc_content < 45: #only return GC content smaller␣
↪than 45
                        print ("%s has a low GC content of %s and sequence␣
↪length of %s nucleotides." % (key, str(round_gc_content), str(length)))␣
↪#print out sentence detailing the variable name, its GC content, and its␣
↪length
                    else: #only return GC content in between
                        print ("%s has a medium GC content of %s and sequence␣
↪length of %s nucleotides." % (key, str(round_gc_content), str(length)))␣
↪#print out sentence detailing the variable name, its GC content, and its␣
↪length
```

```
[79]: SeqList = [{"AX557349":␣
       ↪"gcggatttactcaggggagagcccagataaatggagtctgtgcgtccacagaattcgcacca"},␣
       ↪{"B557211":␣
       ↪"""gcggatttactcaggggagagcccagataaatggagtctgtgcgtccacagaattcgcaccaagcttagatctggccgggg"""},␣
       ↪{"B69116":␣
       ↪"""caccaataaaaaacaagcttaacctaattccggccagatcbstaccagataaatggaagcttagatctggccgggg"""},␣
       ↪{"SingleSeq":␣
       ↪"""tccgtgaaacaaagcggatgtaccggatrhptattccggctatggggcaattcgcggatttactcaggggagagcccagataaatgga
       ↪#passing a list with 4 strings with each having a key-word arg type
```

```
[80]: GC_content(SeqList) #passing a list
```

```
AX557349 has a medium GC content of 53.2 and sequence length of 62 nucleotides.
B557211 has a medium GC content of 55.6 and sequence length of 81 nucleotides.
B69116 has non nucleotide characters.
SingleSeq has non nucleotide characters.
```

```
[ ]:
```