ace02_python_excercise_full

August 17, 2018

0.1 Excercises

1. Calculate GC content of the given DNA sequence
2. Write program that will print the complement of a DNA sequence
3. Write program that will calculate the size of the two fragments produced when digested by EcoRI (G*AATTC)

In [ ]:
```
# excercise 1
# Calculate GC content of the given DNA sequence

dna_seq = "TAAACTTTAAAGTTCAAATAAGACATTCACCGCACTATCAGCGAATTGCCACCCCGGTTTGATCCGTTTTAGAAC
# how many C's and G's
c_count = dna_seq.count("C")
g_count = dna_seq.count("G")
gc_count = c_count + g_count

# or
gc_count = dna_seq.count("C") + dna_seq.count("G")
#print(gc_count)

# length of dna sequence
dna_length = len(dna_seq)
#print(dna_length)

# calculate GC content
gc_content = gc_count / dna_length * 100

# could do the following
#gc_content = (dna_seq.count("C") + dna_seq.count("G")) / len(dna_seq) * 100

# print results
print("GC content is: {:.2f}".format(gc_content))
```

In [7]:
```
# excercise 2
# print the reverse and complement of a DNA sequence
# make a temp seq for ease
dna_seq2 = "ATGC"
print(dna_seq2)
```

```
# attempt 1
replace1 = dna_seq2.replace("A", "T")
replace2 = replace1.replace("T", "A")
replace3 = replace2.replace("G", "C")
final_replace = replace3.replace("C", "G")
#print(final_replace)

# attempt2 - remember that string manipulation is case sensitive
replace1 = dna_seq2.replace("A", "t")
replace2 = replace1.replace("T", "a")
replace3 = replace2.replace("G", "c")
replace4 = replace3.replace("C", "g")
complement = replace4.upper()


# attempt 3
# dictionary holding each bases complement
complement_dict = {"A":"T", "T":"A", "G":"C", "C":"G"}

# reverse the sequence
# https://stackoverflow.com/questions/509211/understanding-pythons-slice-notation
reverse = dna_seq2[::-1]

# empty list to hold complement bases
complement_bases = []

# loop through reverse string and add its complement to a list
for base in reverse:
    complement_bases.append(complement_dict[base])

# join list of complements into new string
rc_dna = "".join(complement_bases)

print(rc_dna)


# *best* approach
# make a reusable function
def reverse_complement(dna):
    complement_dict = {"A":"T", "T":"A", "G":"C", "C":"G"}
#     reverse = dna[::-1]
#     complement_bases = []
#     for base in reverse:
#         complement_bases.append(complement_dict[base])
#     return "".join(complement_bases)

    # list comprehension
```

```
            return "".join([complement_dict[base] for base in dna[::-1]])

        print("the reverse complement of {} is {}".format(dna_seq2, reverse_complement(dna_seq2
```

```
ATGC
GCAT
the reverse complement of ATGC is GCAT
```

In [16]:
```
# excercise 3
# calculate the size of the two fragments produced when digested by EcoRI (G*AATTC)
# make a temp sequence for ease
dna_seq3 = "TTGATCCGTTTTAGAACGTGGCAAGGTGAATTCAAGAATCATTCCTATGAATTCATTATCAGCGAATTGCCAC
ecor1 = "GAATTC"

# length of first fragment1
frag1_length = dna_seq3.find(ecor1) + 1 # account for cut site
frag2_length = len(dna_seq3) - frag1_length
print("fragment 1 length: {}".format(frag1_length))
print("fragment 2 length: {}".format(frag2_length))

frag1_seq = dna_seq3[0:frag1_length]
print("fragment1: {}".format(frag1_seq))
frag2_seq = dna_seq3[frag1_length:]
print("fragment2: {}".format(frag2_seq))
```

```
fragment 1 length: 28
fragment 2 length: 46
fragment1: TTGATCCGTTTTAGAACGTGGCAAGGTG
fragment2: AATTCAAGAATCATTCCTATGAATTCATTATCAGCGAATTGCCACC
```