Counting tetranucleotide frequency

Counting the bases in DNA. The input is assumed to be a single positional argument on the command line, but we'll alter our program to additionally read an input file.

Iterating the characters in a string

The first solution uses a `for` loop to iterate through each character in the input string. We'll establish counter variables to hold the number of As, Cs, Gs, and Ts. Each time we identify one of these bases, we'll add 1 to the counter. Then we'll print them out in the correct order with a single space in between:

```
#!/usr/bin/env python3
"""
Author : Ken Youens-Clark <kyclark@gmail.com>
Purpose: Tetranucleotide frequency
"""

import argparse
import os


# --------------------------------------------------
def get_args():
    """Get command-line arguments"""

    parser = argparse.ArgumentParser(
        description='Tetranucleotide frequency',
        formatter_class=argparse.ArgumentDefaultsHelpFormatter)

    parser.add_argument('dna', metavar='str', help='Input DNA sequence') ①

    args = parser.parse_args() ②

    if os.path.isfile(args.dna): ③
        args.dna = open(args.dna).read() ④

    return args


# --------------------------------------------------
def main():
    """Make a jazz noise here"""

    args = get_args() ⑤
    count_a, count_c, count_g, count_t = 0, 0, 0, 0 ⑥
```

```
    for base in args.dna.lower(): ⑦
        if base == 'a':  ⑧
            count_a += 1
        elif base == 'c':
            count_c += 1
        elif base == 'g':
            count_g += 1
        elif base == 't':
            count_t += 1

    print(f'{count_a} {count_c} {count_g} {count_t}') ⑨


# --------------------------------------------------
if __name__ == '__main__':
    main()
```

① The DNA is defined as a positional string parameter.

② Intercept the arguments.

③ Test if the "dna" value is the name of a file.

④ Open and read the file to use the contents as the "dna" value.

⑤ Get the command-line arguments.

⑥ Initialize four variables to hold the counts for A, C, G, and T.

⑦ Iterate through each base in the lowercased sequence so that we don't have to check both upper- and lowercase bases.

⑧ Check for each wanted character, and increment the correct variable accordingly.

⑨ Print the values using an f-string.

Using str.count()

The `str` class has a method called `str.count()` that will safely count the number of times one string is found inside another string:

```
def main():
    args = get_args()
    dna = args.dna.lower() ①
    print('{} {} {} {}'.format(dna.count('a'), dna.count('c'), dna.count('g'),
                               dna.count('t'))) ②
```

① Since we need to use "dna" several times, copy it to a "dna" variable. Lowercase the value so that we only need to search lowercase bases.

② Use `dna.count()` to find the number of times each of A, C, G, and T are found.

Using a dictionary to count all the characters

```
def main():
    args = get_args()
    count = {} ①

    for base in args.dna.lower(): ②
        if not base in count:     ③
            count[base] = 0       ④
        count[base] += 1          ⑤

    print('{} {} {} {}'.format(count.get('a', 0), count.get('c', 0),  ⑥
                               count.get('g', 0), count.get('t', 0)))
```

① Initialize an empty dictionary to hold the counts.

② Use a "for" loop to iterate through the lowercased bases.

③ Check if the base does not yet exist in the dictionary.

④ Initialize the value for this base to 0.

⑤ Increment the count for this base by 1.

⑥ Use the `dict.get()` method to get each base's count or the default of 0. Usr a format string to create the output to print.

Only count the desired bases

The previous solution will count every character. This solution will only count those found in the dictionary which we initialize with values of 0.

```
def main():
    args = get_args()
    count = {'a': 0, 'c': 0, 'g': 0, 't': 0} ①
    for base in args.dna.lower(): ②
        if base in count:         ③
            count[base] += 1      ④

    print(' '.join([str(count.get(base, 0)) for base in 'acgt'])) ⑤
```

① Initialize the count dictionary with the four bases each with a count of 0.

② Iterate through each lowercased base.

③ Check if the base is found as a key in the `count`.

④ Increment the count for this base by 1.

⑤ Use a list comprehension to iterate through each of the bases A, C, G, and T and create a new list

with the values from the `count`. Join this list on spaces to print.

Using a defaultdict()

The `defaultdict()` is great.

```
def main():
    args = get_args()
    count = defaultdict(int) ①

    for base in args.dna.lower(): ②
        count[base] += 1          ③

    print(' '.join(map(lambda base: str(count.get(base, 0)), 'acgt'))) ④
```

① Initialize the count as a dictionary integer values which default to 0.

② Iterate through each lowercased base.

③ Increment the count for this base by 1.

④ Use a `map()` to express the same code as the list comprehension.

Use a Counter()

The `Counter()` does all this work for you! Our solution is now down to three lines of code:

```
def main():
    args = get_args()
    count = Counter(args.dna.lower()) ①
    print(' '.join(map(lambda base: str(count.get(base, 0)), 'acgt')))
```

① Count the frequency of all the characters in the lowercased input.