Computer Systems Modeling and Verification
(USEEN1)

Lists, Exceptions, Strings

Exercises taken from the *Python Programming Primer*[1] (and updated with type hints).

Textbook *Introduction to Python for Computational Science and Engineering* (2022), Chapters 4-6.

Corresponding lecture slides from *Computational Science and Engineering in Python*:

- Sequences
- Loops
- Exceptions

Exercises. Define the following functions.

1. Write a function `count_vowels(s: str) -> int` that returns the number of letters `'a'`, `'e'`, `'i'`, `'o'`, `'u'`, `'A'`, `'E'`, `'I'`, `'O'`, `'U'` in a given string `s` (the return value is of type integer).

   Examples:

   ```
   >>> count_vowels('This is a test')
   4

   >>> count_vowels('aoeui')
   5

   >>> count_vowels('aoeuiAOEUI')
   10

   >>> count_vowels('NO vOw3ls @t @ll ln thls strlng')
   0
   ```

2. Write a function `product3(a: Sequence[int], b: Sequence[int]) -> list[int]` that takes two sequences of numbers. Both sequence `a` and sequence `b` have three elements. With inputs `a=[ax, ay, az]` and `b=[bx, by, bz]`, the function should return a list which contains the vector product of 3d-vectors `a` and `b`, i.e. the return value is the list:

   `[ay * bz - az * by, az * bx - ax * bz, ax * by - ay * bx]`.

   Examples:

   ```
   >>> product3([1, 0, 0], [0, 1, 0])
   [0, 0, 1]

   >>> product3([1, 2, 4], [3, 5, 6])
   [-8, 6, -1]
   ```

   Change the function to use tuples instead of lists.

3. Write a function `seq_mult_scalar(a: Sequence[int], s: int) -> list[int]` which takes a list of numbers `a` and a scalar (i.e. a number) `s`. For the input `a=[a0, a1, a2, ..., an]` the function should return `[s * a0, s * a1, s * a2, ..., s * an]`.

1. *Python Programming Primer*, Hans Fangohr *et al.* University of Southampton (2016)

Example:

```
>>> seq_mult_scalar([-4, 9, 1], 10)
[-40, 90, 10]
```

4. A function `powers(n: float, k: int) -> list[float]` that returns the list `[1, n, n**2, n**3,..., n**k]` where `k` is an integer. Note that there should be `k+1` elements in the list.

   Examples:

   ```
   >>> powers(2, 3)
   [1, 2, 4, 8]

   >>> powers(0.5, 2)
   [1.0, 0.5, 0.25]
   ```

5. A function `traffic_light(load: float) -> str` that takes a floating point number `load`. The function should return the string:

   - `'green'` for values of `load` below 0.7.
   - `'amber'` for values of `load` equal to or greater than 0.7 but smaller than 0.9
   - `'red'` for values of `load` equal to 0.9 or greater than 0.9

   Example:

   ```
   >>> traffic_light(0.5)
   'green'
   ```