Homework: Python Data Structures, Control Flow, and Generators

This homework is covering Python basics, control flow, functions, data structures, iterators, generators, and strings.

Instructions

1. Submit a single notebook (.ipynb) when finished.
2. Answers are graded on correctness of output, not code length or style.
3. You may use the **starter notebook** to complete this assignment or create a new notebook.
4. If you decide to create a new notebook, make sure your code can pass the assertion tests included in the starter code.

Part 1: Data-Generating Functions

Define the following functions. Each function should return the specified data type. Do not print inside the function.

1. Write a function make_numbers_list(n) that returns a list of integers from 0 to n-1.
2. Write a function make_numbers_tuple(n) that returns a tuple of integers from 0 to n-1.
3. Write a function make_squares_dict(n) that returns a dictionary mapping each integer i to i^2 for i in 0 to n-1.
4. Write a function make_unique_mod_set(n, k) that returns a set containing i % k for i in 0 to n-1.

Hint

The % is the modulus operator in python. a % b returns the remainder of the division of a by b.

Part 2: Assigning and Manipulating Data

5. Using n = 10,000, call each function from Part 1 and assign the results to variables.
6. From the list, create a new list containing only even numbers using a list comprehension.
7. From the tuple, compute the sum of values greater than 5,000.
8. From the dictionary, create a list of keys where the value (square) is divisible by 3.
9. From the set, determine how many unique values it contains and explain why.

Part 3: Strings and String Methods

10. Write a function make_sentence(words) that takes a list of strings and returns a single sentence joined by spaces.
11. Given the string paragraph provided, write code that counts how many times each word appears (case-insensitive). Store the result in a dictionary.
12. From the same string paragraph, create a list of all words longer than 4 characters after stripping punctuation.

Part 4: Control Flow and Iteration

13. Iterate over the list of numbers you created in question 5, and count how many are divisible by both 2 and 5.
14. Iterate over the dictionary you created in question 5, and compute the average of all values whose keys are odd.
15. Using an if / elif / else block, classify numbers from 0 to 20 as 'small' (<10), 'medium' (10–15), or 'large' (>15). Store the result in a dictionary.

Part 5: Generators vs Iterables

16. Write a generator function gen_even_squares(n) that yields squares of even numbers from 0 to n-1.
17. Write an equivalent list-based version list_even_squares(n).
18. Use n = 1,000,000 and time how long it takes to compute the sum of values produced by each approach.

Hint

Using the time module in python, you can store the time before the operation to an object, then similarly record the time once the operation is finished. The elapsed time is the difference between the two. We have implemented this in the starter code already.

19. Briefly explain (2–4 sentences) why the generator version is more memory efficient.

Hint

Python generators produce values lazily, meaning values are created only when needed. Functions like sum() can consume generators without storing all values in memory.