**Python Assessment - 1**

**Question 1: Square Dictionary**

**Problem Statement:**

Write a function square\_dict(nums) that takes a list of integers nums as input and returns a dictionary where the keys are the integers from the list, and the values are the squares of the integers. Use dictionary comprehension to solve the problem.

**Function Signature:**

def square\_dict(nums: list) -> dict:

pass

**Example:**

square\_dict([1, 2, 3, 4])

# Output: {1: 1, 2: 4, 3: 9, 4: 16}

---

**Question 2: Age Classification**

**Problem Statement:**

Write a Python program that prompts the user to input their age. Based on the input, the program should output whether the person is a "Minor" (under 18), "Adult" (18-65), or "Senior Citizen" (above 65). If an invalid input (e.g., negative number or non-numeric input) is given, continue prompting the user until a valid input is received.

**Function Signature:**

def classify\_age():

pass

**Example:**

classify\_age()

# Input: 25

# Output: Adult

---

**Question 3: Sum or Product of List**

**Problem Statement:**

Write a function calculate(nums, operation) that takes two arguments:

nums: a list of integers.

operation: a string that can be either "sum" or "product".

The function should return the sum of the numbers if the operation is "sum", and the product of the numbers if the operation is "product". If no list is provided, the function should assume an empty list. If no operation is provided, the function should default to "sum".

**Function Signature:**

def calculate(nums: list = [], operation: str = "sum") -> int:

pass

**Example:**

calculate([1, 2, 3], "sum")

# Output: 6

calculate([1, 2, 3], "product")

# Output: 6

---

**Question 4: Prime Numbers with List Comprehension**

**Problem Statement:**

Write a Python program that generates a list of all prime numbers between 1 and 100 using list comprehension. Define a helper function is\_prime(n) that returns True if a number n is prime, and False otherwise.

**Function Signature:**

def generate\_primes() -> list:

pass

def is\_prime(n: int) -> bool:

pass

**Example:**

generate\_primes()

# Output: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]

---

**Question 5: Word Frequency**

**Problem Statement:**

Write a function word\_frequency(sentence) that takes a string sentence as input and returns a dictionary where the keys are unique words from the sentence, and the values are their frequency. Ignore case and punctuation.

**Function Signature:**

def word\_frequency(sentence: str) -> dict:

pass

**Example:**

word\_frequency("Hello world! Hello")

# Output: {"hello": 2, "world": 1}

---

**Question 6: Fibonacci Sequence**

Problem Statement:

Write a function fibonacci(n) that takes an integer n and returns the first n numbers of the Fibonacci sequence. The Fibonacci sequence is defined as:

F(0) = 0, F(1) = 1

F(n) = F(n-1) + F(n-2) for n > 1

**Function Signature:**

def fibonacci(n: int) -> list:

pass

**Example:**

fibonacci(5)

# Output: [0, 1, 1, 2, 3]

---

**Question 7: Palindrome Checker**

Problem Statement:

Write a function is\_palindrome(word) that checks whether a given word (string) is a palindrome. A palindrome is a word that reads the same backward as forward (ignoring case and spaces).

**Function Signature:**

def is\_palindrome(word: str) -> bool:

pass

**Example:**

is\_palindrome("Madam")

# Output: True

is\_palindrome("Hello")

# Output: False

---

**Question 8: Count Vowels and Consonants**

Problem Statement:

Write a function count\_vowels\_consonants(word) that takes a string word as input and returns a tuple (vowel\_count, consonant\_count). The function should ignore case and non-alphabetic characters.

**Function Signature:**

def count\_vowels\_consonants(word: str) -> tuple:

pass

**Example:**

count\_vowels\_consonants("Hello World")

# Output: (3, 7)

---

**Question 9: Merge Two Dictionaries**

**Problem Statement:**

Write a function merge\_dicts(dict1, dict2) that takes two dictionaries dict1 and dict2 as input and merges them into one dictionary. If both dictionaries have the same key, the value from dict2 should overwrite the value from dict1.

**Function Signature:**

def merge\_dicts(dict1: dict, dict2: dict) -> dict:

pass

**Example:**

merge\_dicts({'a': 1, 'b': 2}, {'b': 3, 'c': 4})

# Output: {'a': 1, 'b': 3, 'c': 4}

---

**Question 10: Find the Largest Element**

Problem Statement:

Write a function find\_largest(nums) that takes a list of integers nums and returns the largest element in the list without using the built-in max() function.

**Function Signature:**

def find\_largest(nums: list) -> int:

pass

**Example:**

find\_largest([1, 5, 3, 9, 2])

# Output: 9

—

**Summary of 10 Questions:**

1. Square Dictionary (dictionary comprehension)

2. Age Classification (conditional statements & loops)

3. Sum or Product of List (functions with default parameters)

4. Prime Numbers with List Comprehension (list comprehension)

5. Word Frequency (string manipulation & dictionaries)

6. Fibonacci Sequence (recursive function & loops)

7. Palindrome Checker (string manipulation)

8. Count Vowels and Consonants (string manipulation & loops)

9. Merge Two Dictionaries (dictionaries & loops)

10. Find the Largest Element (loops & comparisons)

—