Programming for Artificial Intelligence (Python)
Homework 3
Due: March 29 before class

We have six questions in this homework. The first one is an analysis question, you can type your answer in a cell by commenting it:

```
# I think the logic is ...
```

The other questions are coding questions.

1 Question 1 (answer in words)

We frequently need to understand how functions or loops work in Python. One way to do so is to write out the loop manually. For example, below we define a `sum1` function:

```
def sum1(list):
    s = 0
    for i in list:
        s += i
    return s
sum1([1, 2, 3])
```

The logic evaluation of the function is

1. start `sum1([1,2,3])`: s = 0
2. i = 1, s = 1
3. i = 2, s = 3

4. i = 3, s = 6

5. return: 6

Write out the logic evaluation of the following function. Note: you need to keep track of **all variables**.

```
    def naive_by2(lst, by):
        n = len(lst) // by
        res = []
        for i in range(n):
            res.append([lst[2 * i], lst[2 * i+1]])
        return res
    nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    naive_by2(nums, 2)
```

2 Question 2 (code)

In class, we created a grouper to split `nums=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` into `[[1, 2], [3, 4], [5, 6], [7, 8], [9, 10]]`. However, we wanted to split `nums` into `[[1, 3, 5, 7, 9], [2, 4, 6, 8, 10]]`. Modify the following function to achieve this goal. You do not need to use `iterator`s or `generator`s.

```
    def naive_by2(lst, by):z
        n = len(lst) // by
        res = []
        for i in range(n):
            res.append([lst[2 * i], lst[2 * i+1]])
        return res
```

3 Question 3 (code)

Write a generator that gives out squared numbers like 1, 4, 9, 16, ... in two ways: the function way and the comprehension way. For the comprehension, you can stop at 100.

4 Question 4 (code)

Write a Python generator function that generates Fibonacci numbers indefinitely. The Fibonacci sequence is defined by the recurrence relation: $F(n) = F(n-1) + F(n-2)$, with initial values $F(0) = 0$ and $F(1) = 1$. Test your generator by printing the first 10 Fibonacci numbers. **Hint: you can try the function way**:

```
def fibonacci():
    pass
    while True:
        pass
```

5 Question 5 (code)

You need to write a list comprehension to answer this question.

For a list of words, write a list comprehension that generates a list containing the lengths of words that start with a consonant (any letter except 'a', 'e', 'i', 'o', 'u') and end with a vowel ('a', 'e', 'i', 'o', 'u') from the given list.

For example, if the input list is `["apple", "banana", "orange", "kiwi", "grape"]`, the output list should be [6, 4, 5], since "banana", "kiwi", and "grape" meet the criteria.

Test your program with the input list `["apple", "banana", "orange", "kiwi", "grape", "elephant", "tiger", "lion", "mouse"]`. **(Hint: you can use the `___contains___()` and the `len()` function)**

6 Question 6 (code)

This question needs you to create a `namedtuple`:

```
from collections import namedtuple
...
```

Suppose you are managing a database of students, and each student record consists of the following fields: "name", "age", "grade", and "major". Make a `namedtuple` type to store these information. Here are the possible steps:

1. Create a namedtuple called “Student” with fields “name”, “age”, “grade”, and “major”.
2. Create a list of Student records representing the following students:
    - Alice, 20 years old, grade A, major Economics
    - Bob, 22 years old, grade B, major Mathematics
    - Charlie, 21 years old, grade B, major Physics
3. Print the details of all students in the list.