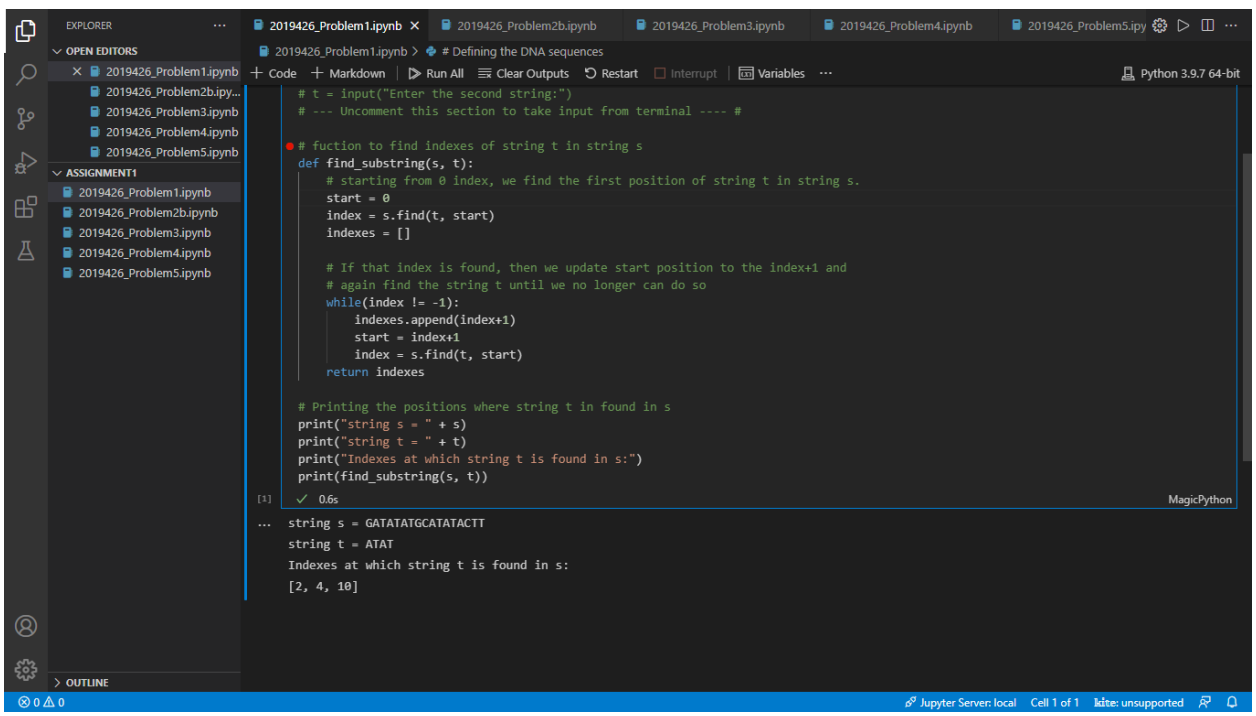
Note:

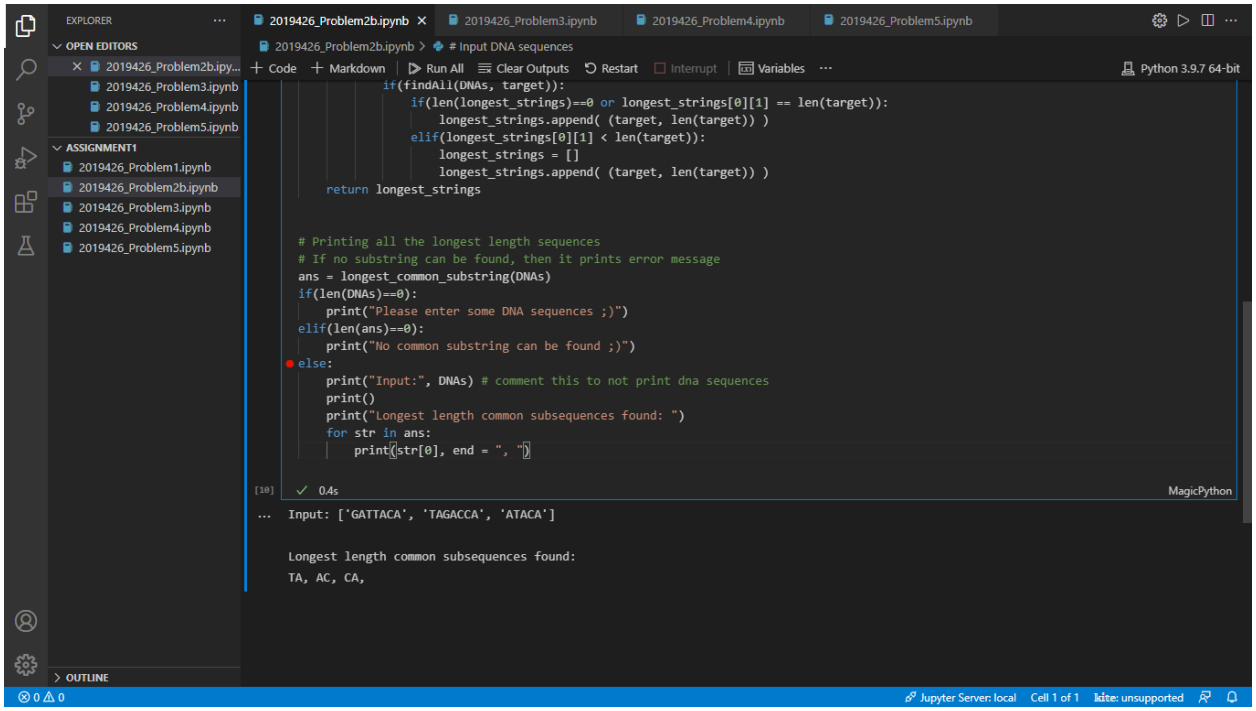
- Each question has a commented input section at the top. Uncomment it to take variable input/terminal input.
- Each input is case-sensitive, so AA and aa will behave differently.
- Each output has lines that you can comment for long input sequences to view output properly.

Screenshots:

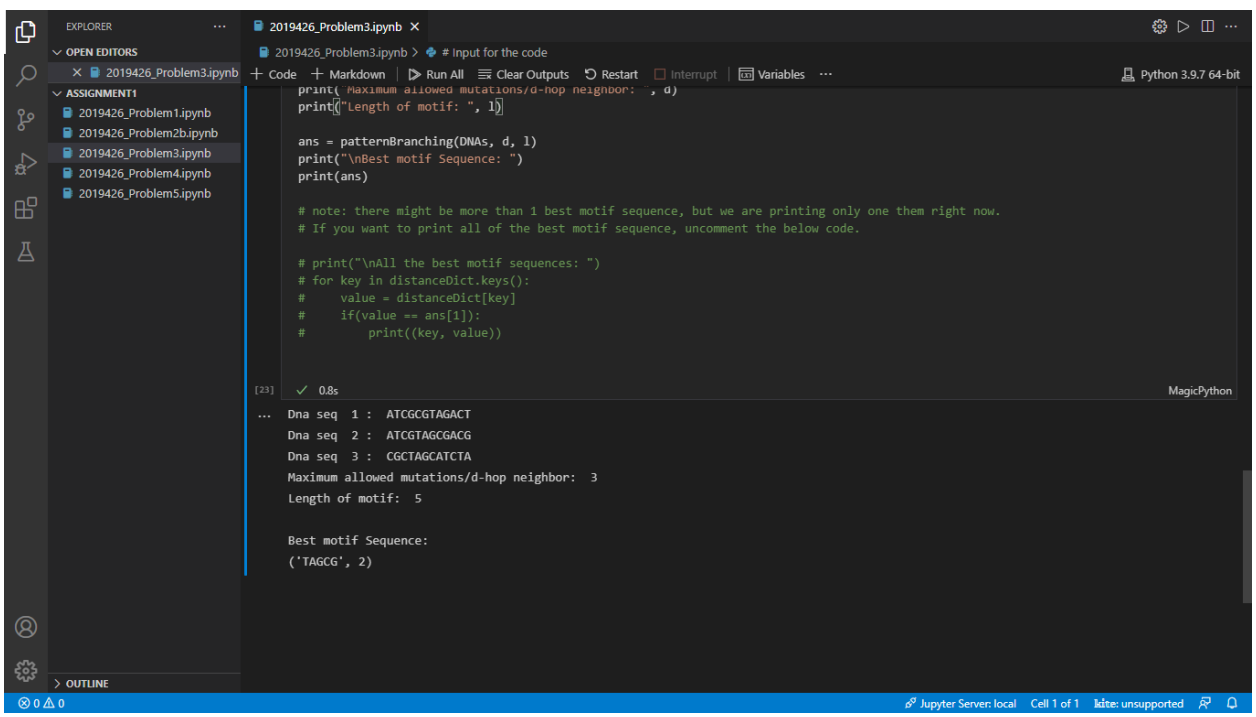
2019426_problem1



2019426_problem2b



2019426_problem3



2019426_problem4

```python
# --- Uncomment this section to take input from terminal ---- #
# s = input("Enter the first string:")
# t = input("Enter the second string:")
# --- Uncomment this section to take input from terminal ---- #

# function to calculate hamming distance
def D(s, t):
    if(len(s)!= len(t)):
        print("distances of sequences in not equal ;) Please enter equal length sequences")
        return -1

    # variable to count unmatched characters
    hamming_distance = 0
    # iterate over the sequence s and t simultaneously and if mismatch is found
    # increase hamming distance variable
    for i in range(0, len(s)-1):
        if(s[i] != t[i]):
            hamming_distance += 1
    return hamming_distance

# printing the hamming distance between s and t
print("string s = " + s)    # comment this to not print dna sequences
print("string t = " + t)    # comment this to not print dna sequences
print("Hamming distance between s and t:")
print(D(s, t))
```

```
string s = GAGCCTACTAACGGGAT
string t = CATCGTAATGACGGCCT
Hamming distance between s and t:
7
```

2019426_problem5

```python
    # For sequence of length 0, edit distance is be length of other sequnce.
    for i in range(0, m+1):
        dp[i][0] = i
    for j in range(0, n+1):
        dp[0][j] = j

    # using the dp recurence, we fill the value for each dp state
    for i in range(1, m+1):
        for j in range(1, n+1):
            if(s[i-1] == t[j-1]):
                dp[i][j] = dp[i-1][j-1]
            else:
                dp[i][j] = min(dp[i-1][j], dp[i][j-1], dp[i-1][j-1]) + 1

    # return final answer
    return dp[m][n]


# Printing the edit distance between s and t
print("string s = " + s)    # comment this to not print dna sequences
print("string t = " + t)    # comment this to not print dna sequences
print()
print("edit distance between s and t:")
print(D(s, t))
```

```
string s = INTENTION
string t = EXECUTION

edit distance between s and t:
5
```

2019426_problem2a

Check the attached answer on the next page.

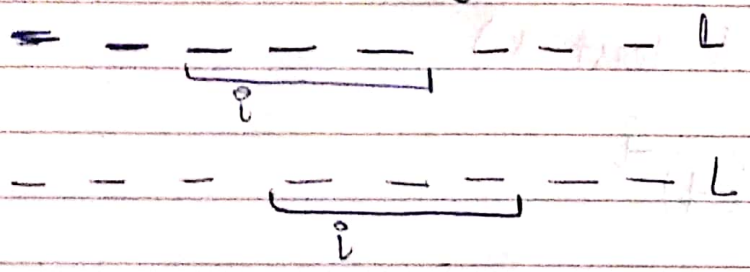
(2a) Given: length of each sequence A and B = L

P(common substring of length at least K nucleotides)

= P(common substring of length K) + ... + P(common substring of length L)

$$= \sum_{i=K}^{i=L} P(\text{common substring of length } i)$$

Probability of common substring of length i =



There are n-i+1 sequences of length i ~~an~~ in A and B. So, we have a total of (n-i+1)×(n-i+1) combinations to choose two i-length sequences.

P(common substring of length i) = no of ways to choose 2 strings of length i × P(substrings of length i to be equal)

$$= (n-i+1)(n-i+1)\, P(\text{equal substrings of length } i)$$

~~P(e~~ P(equal substrings of length i) = P(of one pos match) ^ no of pos

$$= (4/16)^{\wedge}(i) = (4/16)^{i} = (1/4)^{i}$$

~~# Ea You can~~

Reasoning behind last step is you have 4 bases and probability of each is 1/4. Once a base is fixed in first string, you only have 1 choice for base to have same base at that position.

— {A, T, G, C} // ~~{same as fi~~

— {same as upper string}

$$P(\text{common substring of } \underset{\text{len}}{i}) = (n-i+1) \times (n-i+1) \times (1/4)^i$$

$$= (n-i+1)^2 (4)^{-i}$$

P(common substring of length at least K)

$$= \sum_{i=K}^{i=L} (n-i+1)^2 (4)^{-i}$$