Andres Castellanos Carrillo

Video Link:
https://drive.google.com/file/d/1DUUyoRwQpE0ahlejgEyrHveFBsPv0T9g/view?usp=share_link

Code:

```
#question 1

def alignment_score(sequence1, sequence2, match=1, mismatch=-1,
gap=-1):
        ## check length is same or not, if not, report error and quit
    if len(sequence1) != len(sequence2):
        raise ValueError("There is an error")

    alignment_score = 0

    # Calculate the score for each position in the sequences
    for i in range(len(sequence1)):
        if sequence1[i] == sequence2[i]:
            alignment_score += match
        elif sequence1[i] != '-' and sequence2[i] != '-':
            alignment_score += mismatch
        else:
            alignment_score += gap

    # Print the sequences and the alignment score
    print("Sequence 1: ", sequence1)
    print("Sequence 2: ", sequence2)
    print("Alignment score: ", alignment_score)

    # Return the alignment score
    return alignment_score


alignment_score('AATATGATA','AAGTTCATA')
```

```
#question 1.5


def alignment_score(sequence1, sequence2, match=2, mismatch=-2,
gap=-2):
        ## check length is same or not, if not, report error and quit
    if len(sequence1) != len(sequence2):
        raise ValueError("There is an error")

    alignment_score = 0

    # Calculate the score for each position in the sequences
    for i in range(len(sequence1)):
        if sequence1[i] == sequence2[i]:
            alignment_score += match
        elif sequence1[i] != '-' and sequence2[i] != '-':
            alignment_score += mismatch
        else:
            alignment_score += gap

    # Print the sequences and the alignment score
    print("Sequence 1: ", sequence1)
    print("Sequence 2: ", sequence2)
    print("Alignment score: ", alignment_score)

    # Return the alignment score
    return alignment_score

alignment_score('AATATGATA','AAGTTCATA')

#question 2

def alignment_score(sequence1, sequence2, match=1, mismatch=-1,
gap=-1):
        ## check length is same or not, if not, report error and quit
    if len(sequence1) != len(sequence2):
        raise ValueError("There is an error")

    alignment_score = 0

    # Calculate the score for each position in the sequences
    for i in range(len(sequence1)):
        if sequence1[i] == sequence2[i]:
```

```
            alignment_score += match
        elif sequence1[i] != '-' and sequence2[i] != '-':
            alignment_score += mismatch
        else:
            alignment_score += gap

    # Print the sequences and the alignment score
    print("Sequence 1: ", sequence1)
    print("Sequence 2: ", sequence2)
    print("Alignment score: ", alignment_score)

    # Return the alignment score
    return alignment_score

alignment_score('A-TAT-ATA','AATTTC-TA')

#question 2.5

def alignment_score(sequence1, sequence2, match=2, mismatch=-2,
gap=-2):
        ## check length is same or not, if not, report error and quit
    if len(sequence1) != len(sequence2):
        raise ValueError("There is an error")

    alignment_score = 0

    # Calculate the score for each position in the sequences
    for i in range(len(sequence1)):
        if sequence1[i] == sequence2[i]:
            alignment_score += match
        elif sequence1[i] != '-' and sequence2[i] != '-':
            alignment_score += mismatch
        else:
            alignment_score += gap

    # Print the sequences and the alignment score
    print("Sequence 1: ", sequence1)
    print("Sequence 2: ", sequence2)
    print("Alignment score: ", alignment_score)

    # Return the alignment score
    return alignment_score
```

```
alignment_score('A-TAT-ATA','AATTTC-TA')


#question 3

def alignment_score(sequence1, sequence2, match=2, mismatch=-2,
gap=-2):
       ## check length is same or not, if not, report error and quit
    if len(sequence1) != len(sequence2):
        raise ValueError("There is an error")

    alignment_score = 0

    # Calculate the score for each position in the sequences
    for i in range(len(sequence1)):
        if sequence1[i] == sequence2[i]:
            alignment_score += match
        elif sequence1[i] != '-' and sequence2[i] != '-':
            alignment_score += mismatch
        else:
            alignment_score += gap

    # Print the sequences and the alignment score
    print("Sequence 1: ", sequence1)
    print("Sequence 2: ", sequence2)
    print("Alignment score: ", alignment_score)

    # Return the alignment score
    return alignment_score

alignment_score('AGCTGAA','AATTTCAGAGA')



#question 4

table_list = [[1,1,1],[2,2,2],[3,3,3]]

for i in range(len(table_list)):
    print("row{}: {}".format(i+1, table_list[i]))

#question 4.1
table_list = [[1,1,1],[2,2,2],[3,3,3]]
```

```
table_list[0] = [-1,-1,-1]

for i in range(len(table_list)):
    print("row{}: {}".format(i+1, table_list[i]))


#question 4.2

table_list = [[1,1,1],[2,2,2],[3,3,3]]

table_list[0] = [-1,-1,-1]

for i in range(len(table_list)):
    table_list[i][0] = -1

for i in range(len(table_list)):
    print("row{}: {}".format(i+1, table_list[i]))


#question 5

def alignment_table(sequence1, sequence2):
    length1 = len(sequence1)
    length2 = len(sequence2)

    # alignment table with M+1 rows and N+1 columns
    table = [[0] * (length2 + 1) for i in range(length1 + 1)]

    # print sequence 1 and sequence 2 with their lengths
    print("Sequence 1: {} with length {}".format(sequence1, length1))
    print("Sequence 2: {} with length {}".format(sequence2, length2))

    # print the alignment table with all values = 0
    print("The initial table with dimension 10 and 8 for the sequence
allignment is: ")

    for row in table:
        print(row)

alignment_table('AATTATATT', 'ACGTTAT')
```

```
#question 6

def alignment_table_updated(sequence1, sequence2):
  length1 = len(sequence1)
  length2 = len(sequence2)

  # alignment table with M+1 rows and N+1 columns
  table = [[0] * (length2 + 1) for i in range(length1 + 1)]

# initialize the first row
  for j in range(1, length2+1):
    table[0][j] = table[0][j-1] - 1

# print sequence 1 and sequence 2 with their lengths
  print("Sequence 1: {} with length {}".format(sequence1, length1))
  print("Sequence 2: {} with length {}".format(sequence2, length2))

# print the alignment table with all values = 0
  print("The initial table with dimension {} and {} for the sequence
alignment is: ".format(length1+1, length2+1))
  for row in table:
    print(row)

alignment_table_updated('AATTATATT', 'ACGTTAT')


#question 6.1

def alignment_table(sequence1, sequence2):
    length1 = len(sequence1)
    length2 = len(sequence2)

    # alignment table with M+1 rows and N+1 columns
    table = [[0] * (length2 + 1) for i in range(length1 + 1)]

    # initialize the first row and first column with gap penalty
    for i in range(1, length2+1):
        table[0][i] = -1 * i
    for i in range(1, length1+1):
        table[i][0] = -1 * i

    # print sequence 1 and sequence 2 with their lengths
```

```
    print("Sequence 1: {} with length {}".format(sequence1, length1))
    print("Sequence 2: {} with length {}".format(sequence2, length2))

    # print the alignment table with all values = 0
    print("The initial table with dimension {} and {} for the sequence
alignment is: ".format(length1+1, length2+1))

    for row in table:
        print(row)

alignment_table('AATTATATT', 'ACGTTAT')
```