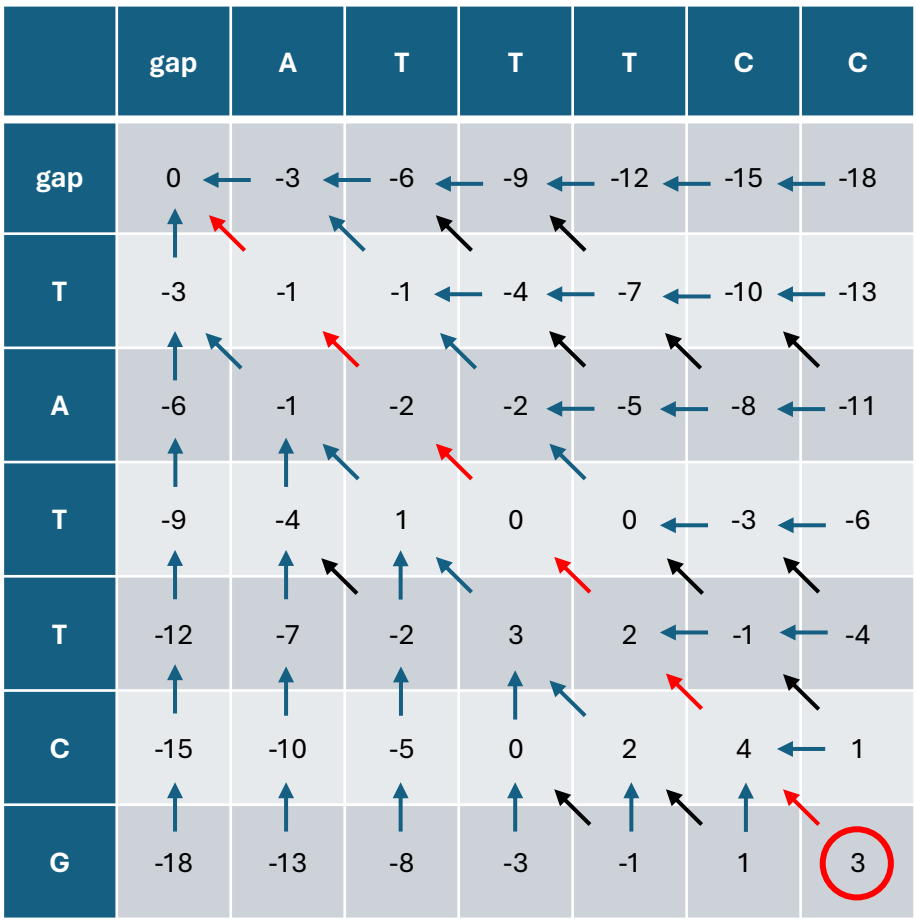
GLOBAL

Seq_2

Seq_1



Seq_1 ATTTCC

Seq_2 TATTCG

no multiple alignments

LOCAL

Seq_2

Seq_1

	gap	A	T	T	T	C	C
gap	0	0	0	0	0	0	0
T	0	0	2	2	2	0	0
A	0	2	0	1	1	1	0
T	0	0	4	2	3	0	0
T	0	0	2	6	4	2	0
C	0	0	0	3	5	6	4
G	0	0	0	0	2	4	5

Seq_1 TATTCG TATTCG

Seq_2 ATTTCC ATTTCC

two multiple alignments

```python
from sequence_alignments import (
    create_submat,
    needleman_wunsch,
    smith_waterman,
    recover_align,
    recover_align_local,
)

SEQ_1 = 'TATTCG'
SEQ_2 = 'ATTTCC'
MATCH = 2
MISMATCH = -1
GAP = -3

def get_alignments(
    seq_1: str = SEQ_1,
    seq_2: str = SEQ_2,
    match: int = MATCH,
    mismatch: int = MISMATCH,
    gap: int = GAP,
):
    '''
    Given two sequences and the match, mismatch, and gap scores,
    print all the relevant information about the global and local alignments.

    Args:
    - seq_1 (str): the first sequence to be aligned
    - seq_2 (str): the second sequence to be aligned
    - match (int): the score for a match
    - mismatch (int): the score for a mismatch
    - gap (int): the score for a gap
    '''
    alphabet = set(seq_1 + seq_2)

    # GLOBAL ALIGNMENT
    # Determine the score and traceback matrix.
    sm = create_submat(match, mismatch, alphabet)
    gsm, gtm = needleman_wunsch(seq_1, seq_2, sm, gap)
    print('Global score matrix:', gsm)
    print('Global traceback matrix:', gtm)

    # Determine the best score.
    gbs = gsm[-1][-1]
    print('Global alignment score:', gbs)

    # Retrieve the optimal sequence alignment.
    oga = recover_align(gtm, seq_1, seq_2)
    print('Optimal global alignment:', oga)

    # Explain if there are multiple best alignments.
    print('Multiple best global alignments:', 'Yes' if len(oga) > 1 else 'No')

    print()

    # LOCAL ALIGNMENT
    # Determine the score and traceback matrix.
    lsm, ltm, lbs = smith_waterman(seq_1, seq_2, sm, gap)
    print('Local score matrix:', lsm)
    print('Local traceback matrix:', ltm)

    # Determine the best score.
    print('Local alignment score:', lbs)

    # Retrieve the optimal sequence alignment.
    ola = recover_align_local(lsm, ltm, seq_1, seq_2)
    print('Optimal local alignment:', ola)

    # Explain if there are multiple best alignments.
    print('Multiple best local alignments:', 'Yes' if len(ola) > 1 else 'No')


if __name__ == '__main__':
    get_alignments()
```