# ch8_output

May 23, 2021

## 1 “translate” example output

[31]:
```
def translate(mrna, reading_frame=1):
    """
    Translates an mRNA sequence to a protein sequence by extracting codon␣
 ↪triplets
    from the desired `reading_frame`.

    :param mrna: mRNA sequence
    :type mrna: str
    :param reading_frame: Desired reading frame. Must be 1, 2, or 3. [Default: 1]
    :type reading_frame: int
    :return: Translation of the mRNA sequence at the desired `reading_frame).
    :rtype: str
    """

#   Values below 0 or above 3 are invalid, and yield an empty output (blank␣
 ↪line):

    if reading_frame > 3 or reading_frame < 1:
        return str()

    frame_offset = 1
    starting_frame = reading_frame - frame_offset

    protein = convert_sequence(mrna[starting_frame:], TO_AMINO_ACID)

    return protein
```

[21]:
```
mrna = 'CCGAUGUUGACUUAG'
peptide = translate(mrna)
print(peptide)

peptide = translate(mrna, 2)
print(peptide)
```

```
#Values below 0 or above 3 are invalid, and yield an empty output (blank line):

peptide = translate(mrna, reading_frame=5)
print(peptide)
```

```
P<Met>LT<STOP>
RC<STOP>L
```

[ ]:

```

```

## 2 "get_reading_frames" example output

[22]:

```
def get_reading_frames(mrna):
    """
    Produces the translation for all of the reading frames in an mRNA sequence.

    :param mrna: mRNA sequence
    :type mrna: str
    :return: Dictionary of translations in the following format:

        {
            'frame 1': 'translation for frame 1',
            'frame 2': 'translation for frame 2',
            'frame 3': 'translation for frame 3'
        }

    :rtype: dict

    """
    reading_frames = (1, 2, 3)
    result = dict()

    for frame in reading_frames:
        result[f"frame {frame}"] = translate(mrna, reading_frame=frame)

    return result
```

[23]:

```
mrna = 'CCGAUGUUGACUUAG'
frames = get_reading_frames(mrna)
print(frames)
```

```
{'frame 1': 'P<Met>LT<STOP>', 'frame 2': 'RC<STOP>L', 'frame 3': 'DVDL'}
```

[ ]:

```

```

## 3 “find_exons” with example output

```
[24]: def find_exons(seq):
          """
          Extracts all of the possible exons from a polypeptide sequence.

          An exon is defined as any sequence with at least 2 residues, which is
          located between a Methionine (M) residue, and a `<STOP>` marker.

          :param seq: Polypeptide sequence.
          :type seq: str
          :return: List of exons (string values) extracted from the polypeptide␣
       ↪sequence.
          :rtype: list

          """
          pattern = re.compile(r'(M\w+?)<STOP>', re.IGNORECASE)
          found = pattern.findall(seq)

          return found
```

```
[25]: mrna = 'CCGAUGUUGACUUAGCUGAUGUUUUUGUUGAUCGUGUAGGGG'
      peptide = translate(mrna)
      print(peptide)
      print('')

      peptide = peptide.replace('<Met>', 'M')
      exons = find_exons(peptide)
      print(exons)
      print('')
```

```
P<Met>LT<STOP>L<Met>FLLIV<STOP>G

['MLT', 'MFLLIV']
```

## 4 “get_sequence_exons” example output

```
[26]: def get_sequence_exons(seq):
          """
          Extracts all of the possible exons from a polypeptide sequence.

          An exon is defined as any sequence with at least 2 residues, which is
          located between a Methionine (M) residue, and a `<STOP>` marker.

          :param seq: Polypeptide sequence.
```

```
    :type seq: str
    :return: List of dictionaries containing the exon number, total number of␣
↪exons
             extracted from the sequence, length of the exon, and the exon␣
↪sequence,
             in the following format:

        {
            'exon number': int (1 or more),
            'total exons': int (1 or more),
            'length': int (2 or more),
            'sequence: str
        }

    :rtype: list
    """

    exons_list = find_exons(seq.replace('<Met>', 'M'))
    found_len = len(exons_list)

    result = list()

    for exon_number, exon in enumerate(exons_list, start=1):
        item = {
            'exon number': exon_number,
            'total exons': found_len,
            'length': len(exon),
            'sequence': exon,
        }

        result.append(item)

    return result
```

[28]:
```
mrna = 'CCGAUGUUGACUUAGCUGAUGUUUUUGUUGAUCGUGUAGGGG'

peptide = translate(mrna)
print(peptide)
print('')

exons = get_sequence_exons(peptide)
print(exons)
print('')
```

```
P<Met>LT<STOP>L<Met>FLLIV<STOP>G

[{'exon number': 1, 'total exons': 2, 'length': 3, 'sequence': 'MLT'}, {'exon
number': 2, 'total exons': 2, 'length': 6, 'sequence': 'MFLLIV'}]
```

## 5 "extract_exons" example output

[29]:
```python
def extract_exons(reading_frames):
    """
    Extracts all of the possible exons from each reading frame.

    An exon is defined as any sequence with at least 2 residues, which is
    located between a Methionine (`M`) residue, and a `<STOP>` marker.

    :param reading_frames: Dictionary of reading frames in the following format:

        {
            'frame 1': 'translation for frame 1',
            'frame 2': 'translation for frame 2',
            'frame 3': 'translation for frame 3'
        }

    :type reading_frames: dict
    :return: Dictionary of exons in the following format:

        {
            'frame 1': [
                # Output of `get_sequence_exons` for frame 1.
            ],
            'frame 2': [
                # Output of `get_sequence_exons` for frame 2.
            ],
            'frame 3': [
                # Output of `get_sequence_exons` for frame 3.
            ],
        }

    :rtype: dict

    """
    result = dict()

    for frame_name, translation in reading_frames.items():
        result[frame_name] = get_sequence_exons(translation)

    return result
```

[30]:
```python
mrna = 'CCGAUGUUGACUUAGCUGAUGUUUUUGUUGAUCGUGUAGGGG'
```

```
#    First, we extract the reading frames:

frames = get_reading_frames(mrna)
print(frames)
print('')


#    Now, we can use the reading frames dictionary to extract the exons:

exons = extract_exons(frames)
print(exons)
print('')
```

```
{'frame 1': 'P<Met>LT<STOP>L<Met>FLLIV<STOP>G', 'frame 2':
'RC<STOP>LS<STOP>CFC<STOP>SCR', 'frame 3': 'DVDLADVFVDRVG'}

{'frame 1': [{'exon number': 1, 'total exons': 2, 'length': 3, 'sequence':
'MLT'}, {'exon number': 2, 'total exons': 2, 'length': 6, 'sequence':
'MFLLIV'}], 'frame 2': [], 'frame 3': []}
```

[ ]:

[ ]: