main

This module describes the API of the Chromosome 12 Browser Implementation of the Browser as a Project of Biocomputing II

For the MSc. Bioinformatics, in Birkbeck College & UCL Taught by: Dr. Andrew C.R. Martin, UCL

Version 0.1b Beta ('Making it work')

Todo: * Finalise the testing

For more information about the API see doc/docs.md For the essay doc/essay.pdf

identify_coding_regions

```
identify_coding_regions(dna)
```

Wrapper of the database.db_API to return the coding regions of the input DNA sequence Args: DNA(str): The input DNA sequence Return: CDS(list): Coding Regions in the specified DNA sequence

generate_coding_sequence

```
generate_coding_sequence(dna)
```

Wrapper of the database.db_API to return the joined coding sequence of the input DNA sequence Args: DNA(str): The input DNA sequence Return: cseq(str): DNA coding sequence

count_codon_usage

```
count_codon_usage(gene)
```

Calculates the codon usage for a specified gene It can be used to calculate the usage either across a region or the whole See CUsage.py for the implementation of the Class

Args: gene(str): Gene to calculate CDS codon_loc(int): The location of the codon Return: codon_count(float): CDS

align

```
align(dna, protein, matrix, gap_penalty)
```

Implementation of a Waterman alignment See AlignSequence module for more

Args: dna(str): given dna to be aligned protein(str): protein sequence to be aligned matrix(list): a lookup matrix input gap_penalty(int): the value of the required gap_penalty Return: alignment (list[list]): an array including a list of the alignments

identify_re_sites

```
identify_re_sites(self, dna, re_sites)
```

Identify the restriction enzyme cut locations in the DNA Args: dna(str): DNA to calculate the RE cut locations Return: re_location(list): the location of the RE cut

re_unique

```
re_unique(dna, protein)
```

Generate a list of the restriction enzyme sites of the DNA Args: dna(str): The requested dna (list): The location of the codon Return: found(list): unique CDS

get_protein_info

```
get_protein_info(self, protein)
```

Database API wrapper to get the requested info for a specific protein Args: protein(str): ID of a protein Return: ac_number(int): protein accession number protein_id(int): the ID of the protein protein_name(str): the name of requested protein aa_seq(str): the aminoacid sequence of the requested protein