Programming Exercices

Series 1

Nuno D. Mendes

Exercices

1. **maketrans**

   The goal of this exercise is for you to implement the functionality that is given by the function `maketrans` of the `string` module.

   The `maketrans` function produces a 256-character long string which is interpreted as a translation table by the string method `translate`.

   Each position in the 256-character long string is the target translation for the corresponding ASCII character, indexed by the position in the sequence, e.g., the appropriate translation for the input character 'a' will be in position 97 of the translation string because 'a' is the 97th character of the ASCII table.

   The built-in functions `ord` and `chr` might be useful for this exercise.

   Make:

   ```
   import string
   help(string.maketrans)
   ```

   to obtain details of how `maketrans` should work.

2. **Primality Tester**

   Write a program such that, given a positive integer, determines whether it is a prime number.

3. **File statistics**

   The goal of this exercise is to produce a program that opens and reads a file and counts the occurrences of each different character in that file.

The name of the file to be analyzed should be passed as a command-line argument. If the given filename does not exist, the appropriate error message should be printed.

Upon determining the number of occurrences of each character, an ordered listing of all characters found alongside with their number of occurrences should be printed to a file named "report_<filename>", where <filename> should be the name of the file that was analyzed.

4. **FASTA**

Many genome sequences appear in FASTA format. Roughly, a FASTA file has the following appearance:

```
>gi|62484574|ref|NR_002251.1| Drosophila melanogaster mir-184S, miscRNA
CCTTATCATTCTCTCGCCCCG

>gi|62484573|ref|NR_002250.1| Drosophila melanogaster mir-184, miscRNA
TGGACGGAGAACTGATAAGGGC

>gi|57902744|gb|AY865942.1| Macaca nemestrina microRNA mir-184 gene, complete sequence
ACTTGCTGTAGGGACTTTATGAATGGCATGTGGGTGTGAGCTTAGCACAGAACCCTAGGGAGGAGCAGGA
CCTGCTGGGCCGGGGCCTCAGACCTGCGAGAGCCACTGGGTAAAGACTTCACTAACTTCGGCTTATTTAT
TAATTTTAAAATTTAAATTAATTTCTATTTTTAATTGATTTTTTTAAATGCAAAGAATCTACTACTTTCC
ATAGCTGTCCAGAGCTGCATGTTTGAATTTCTGTGTGCAGAAACATAAGTGACTCTCCAGGTGTCAGAGG
GAGAGACTGGGGCGAGAGGCCAGAGCAACGTAGAAGGGCACAGAGGGGCTGTGAATTTGAGGCAGGGGTG
GAGCTGCAGAGAGGGGGCGGGGAGGGCTCGCCAGGAAATCAAACGTCCGTTTACATCTTGTCCTGCAAAG
CTTCATCAAAACTTCTTTGCCGGTCAGTCACGTCCCCTTATCACTTTTCCAGCCCAGCTTTATGACTGTA
AGTGTTGGACGGAGAACTGATAAGGGTAGGTGATTGACACTCACAGCCTCCGGAACCCCCCGCACCGCCT
GCACCTGCATGATGGAGAAAACCTGGCGCTCCCGCTCTGGGTGCCCGAAGACAGCAGGGGATTCCAGGAG
GAGACCTTGGGCATATGGGGGCCCAGGTATGCGCCCCCTGCCTGAGGATGCTGGGGTAGCCT

>gi|57902743|gb|AY865941.1| Pongo pygmaeus microRNA mir-184 gene, complete sequence
TGGCCTGGGGCCAGCCTCTCTGGATGACCAATCCTGGTTGGAGGAGGAGACTTGCTGTAGGGACTTTATG
AATGGCATGTGGGTGTGGGCTTAGCACAGAACCCTGGGGAGGAGCAGGACCTGCTGAGCCCGGGCCTCAG
ACCTGCAAGAGCCACTGGGTAAAGACTTCACTAACTTCGGCTTATTTATTAATTTTAAAATTTAAATTAA
TTTCTATTTTTAATTGATTTTTTTAAATGCAAAGAATCCGCTACTTTCCATAGCTGTCCAGAGCTGCATG
TCTGAATTTCTGTGTGCAGAAACATAAGTGACTCTCCAGGTGTCAGAGGGAGAGACCGGGGCCAGAGGCC
AGAGCGAAGTAGAAGGGCACAGAGGGGCTCTGAATTTGAGGCAGAGGAGGAACTGCAGAGAGGGGGCGGG
GAGGGCTCGACGGGAAATCAAACGTCCATTTACATCTTGTCCTGCAGAGCTTCATCAAAACTTCTTTGCC
GGCCAGTCACGTCCCCTTATCACTTTTCCAGCCCAGCTTTGTGACTGTAAGTGTTGGACGGAGAACTGAT
AAGGGTAGGTGATTGACACTCACAGCCTCCGGAACCCCCGCGCCGCCTGCACCTGCGTGATGGGGAAAAC
CTGGCGCTCCCGCTCTGGCTGCCCGAGGAAAGCAGGGGATTCCAGGAGGAGACCTTGGGCATAGGGGGCC
CAGGTATGCGCCCCCTGCCTGAGGATGCTGGGGTAGCCT
```

Given a FASTA file:

(a) Count the number of sequences

(b) Determine the average size of the sequences

(c) Count the number of occurrences of each nucleotide

(d) Count the number occurrences of each dinucleotide (pair of nucleotides)

(e) Assuming that the given sequences refer to concatenated exons, that there are no phase shifts and the the first codon starts in the first position of the sequences, determine the corresponding sequence of aminoacids. (represent stop codons with '$*$', see http://www.chem.qmul.ac.uk/iupac/AminoAcid/AA212.html for the standard one-letter representation of aminoacids).

(f) Change the previous program, removing the assumption about the start position. Initiate translation only upon encountering an appropriate start codon. Similarly, stop translation upon encountering a standard stop codon.

5. **Very Simple Motif Finder**

   Download the FASTA file at http://algos.inesc-id.pt/∼ndm/file.fasta. Write a program that finds which 14-nucleotide long motif is common to all sequences.

6. **URM emulator**

   Write a program that emulates a URM. The program receives arguments in the command-line. The first argument should be the name of the URM script and the following n arguments should be the values to be placed in registers R_1 to R_n. The URM script should have this appearance:

```
1: T(3,1)
2: T(4,2)
3: J(3,2,9)
4: J(4,1,9)
5: S(3)
6: S(4)
7: S(0)
8: J(0,0,3)
9: HALT
```

   The emulator should be insensitive to:

   - Extra spaces
   - Empty lines
   - Capitalization of commands
   - Whether the script specifies a HALT or H command

   Upon running the URM script, the program should print the contents of R_0.

Notes

1. In order to complete these exercises you should read chapters 1 through 16 of the manual

2. Check with Python documentation if you have any doubts. You can find the language reference manual at http://docs.python.org/ref/ref.html and the Python Tutorial at http://docs.python.org/tut/tut.html. Remeber to use the `help` function on interactive mode, and the `pydoc` command in the command-line

3. Make sure your code is elegant and readable and that the appropriate error messages are printed every time the assumptions about program arguments are violated

4. Be lazy = be smart! Try to produce programs that require you to write the least amount of code

5. You are expected to complete this series of exercises until October, 20

6. To obtain comments, suggestions, to dissipate any doubts and to deliver your code you should write to `ndm@algos.inesc-id.pt`