ADMINISTRATIVE MATTERS

- All correspondence via pp1ex@rostlab.org
- All emails to other addresses will be silently ignored
- DO NOT WRITE EMAILS TO DR. KRUSCHE, PROF. BRUEGGE OR PRESIDENT OF THE UNIVERSITY
- Standard exercise group has been deleted
- Register for one of the groups “Group 1, ..., Group 4”
FIRST WEEK CONCLUSIONS

- Python 3
- **Strict** compliance with the naming scheme and input/output specification
- Check if your code has syntax errors (run it locally)

```bash
$ python main.py
File "main.py", line 9
    return ''.join(COMPLEMENTS[character]) for character in strand.upper()
                         ^
SyntaxError: invalid syntax
```

- Commit and **push**
FIRST WEEK CONCLUSIONS

- If a specification says that a function should take `arg` as an argument and return `return_value`, that means

```python
def some_func(arg):
    return return_value
```

and not

```python
def some_func():
    arg = input()
    print(return_value)
```
“collection failure” means the testing framework wasn’t even able to run your code (due to syntax errors or querying stdin, etc.)
ORFS (OPEN READING FRAMES)

- DNA sequence consisting of:
  1. **Start codon** (ATG; also encoding Methionine)
  2. Intermediate codons (encoding amino acids)
  3. **Stop codon** (TAA, TAG, TGA; do not encode amino acids)

- A single ORF can encode multiple protein sequences by
  - Not cutting some introns
  - Excluding some exons
  - This is of no concern for our exercise!
  - We always assume 1 ORF = 1 protein!
DNA “READING FRAMES”

- Since codons are triplets, we have three reading frames.
- ORFs stop at the first stop codon on the same frame.

```
„GCTATGAGGTCATG\textcolor{red}{GCTTCTG}TAGTAACG\textcolor{red}{TGAC}”
```

- F1: GCT \textcolor{red}{ATG} AGG TCA TGG CTT CTG \textcolor{red}{TAG} TAA CGT GAC
- F2: CTA TGA GGT CAT GGC TTC TGT AGT AAC GTG ac
- F3: TAT GAG GTC \textcolor{red}{ATG} GCT TCT GTA GTA ACG \textcolor{red}{TGA} c

- One ORF on F1: \textcolor{red}{ATG}AGGTCATGG\textcolor{red}{GCTTCTG}TAG
- One ORF on F3: \textcolor{red}{ATG}GCTTCTGTAGTAACG\textcolor{red}{TGA}
## Reverse Complementary Strand

> Complementary: A→T, T→A, G→C, C→G

> Reverse: ATGC → CGTA

<table>
<thead>
<tr>
<th>Strand</th>
<th>DNA sequence (read from 5’ to 3’)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary</td>
<td>5’ GCT\text{ATG}AGGT\text{CATG}GCTTCT\text{GTA}A\text{ACG}\text{TGAC} 3’</td>
</tr>
<tr>
<td>Rev. Comp.</td>
<td>3’ \text{CGAT}\text{ACTCCAGTA}CCGAAGACAT\text{CATTGC}\text{ACTG} 5’</td>
</tr>
<tr>
<td>Rev. Comp. (reading direction)</td>
<td>5’ \text{GTCACGTTACTACAGAAGCC}\text{ATGACCTCATAGC} 3’</td>
</tr>
</tbody>
</table>
HOW TO INDEX ORFS

- We start at index 0: first nucleotide (nt) in primary strand
- Rev. comp. strand is indexed according to primary strand
  - First nt in rev.comp. has same index as last nt in primary
  - Last nt in rev.comp. has same index as first nt in primary (i=0)
- Start of an ORF is position of first nt in start codon
- Stop of an ORF is position of last nt in stop codon
EXAMPLE

<table>
<thead>
<tr>
<th>Strand</th>
<th>DNA sequence (read from 5’ to 3’)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary</td>
<td>5’ GCTATGAGGTCATGGCTTCTGTAGTAACG TGA C 3’</td>
</tr>
<tr>
<td>Rev. Comp.</td>
<td>3’ CGATACGCCAGTACTCGAAGACATCATTGC ACTG 5’</td>
</tr>
<tr>
<td>Rev. Comp. (reading direction)</td>
<td>5’ GTCACGTTACTACAGAAGCCATGACCTCATAGC 3’</td>
</tr>
</tbody>
</table>

- **ATGAGGTCATGGCTTCTG**
  - ORF on primary F1; Start: 3, Stop: 23
- **ATGGCTTCTGTAGTAACG**
  - ORF on primary F3; Start: 11, Stop: 31
- **ATGACCTCATAG**
  - ORF on rev.comp. F3; Start: 12, Stop: 1
Task 2.2 (E) Genetic Code

- Implement the genetic code given in [https://en.wikipedia.org/wiki/Genetic_code#Standard_codon_tables](https://en.wikipedia.org/wiki/Genetic_code#Standard_codon_tables) as a dictionary where DNA triplets are keys and one letter amino acid code symbols are values. Remember that DNA in contrast to RNA contain T instead of U.

- Write a function called `translate` that takes a DNA sequence coding for an open reading frame (ORF). An ORF is a DNA/RNA sequence that precisely codes for a protein, i.e. no leading or trailing untranslated DNA/RNA residues.

- The function `translate` returns the corresponding amino acid sequence in one letter code as a string.
TASK 2.2 (E) GENETIC CODE

Dictionary:

```python
# Genetic code dictionary
code_dict = {
    'TTA': 'L',
    'TTG': 'L',
    ...
    'TGG': 'W'
}
```
Task 2.2 (E) Genetic Code

- `translate`

```python
def translate(orf):
    orf = orf.upper()
    codon_length = 3
    amino_acids = []
    for i in range(0, len(orf), codon_length):
        amino_acids.append(code_dict[orf[i:i+codon_length]])
    return ''.join(amino_acids)
```
write a function `aa_dist` that

- takes an amino acid sequence in one letter code
- returns the amino acid distribution as a Python dictionary with the single letter codes as keys and the frequencies as values
TASK 2.3 (E) AMINO ACID DISTRIBUTION

```python
from collections import Counter

def aa_dist(aa_seq):
    counted = Counter(aa_seq)
    for key in counted:
        counted[key] /= len(aa_seq)
    return counted
```
write a function `read_fasta` that

- takes a FASTA file name as an argument
- opens and reads one sequence
- returns a tuple consisting of the header and the sequence body
**TASK 2.4 (E) FASTA SEQUENCE FORMAT**

```python
# Exercise 2.4

def read_fasta(filename):
    f = open(filename, "r")

    # Save header
    first_line = f.readline()
    header = first_line[1:].strip()
```
# Read in sequence
aa_seq = ""
sequence_started = False
for line in f:
    if line.startswith(">") or line.startswith(';'):
        if sequence_started:
            break
        else:
            continue

    sequence_started = True

    aa_seq += line.strip()
PROTEIN PREDICTION 1 FOR COMPUTER SCIENTISTS

TASK 2.4 (E) FASTA SEQUENCE FORMAT

```python
# Remove trailing asterisk, if present
if aa_seq.endswith('*'):
    aa_seq = aa_seq[:-1]

return (header, aa_seq)
```
THANK YOU!

QUESTIONS?