Laboratory 5: beginning Python3
PCR primer annealing temperature calculations

This exercise is designed to teach simple Python regular expressions and basic built–in functions. Today you will create a script ('otm.py') that will calculate an optimal T_m for PCR using the nearest–neighbor algorithm of Borer et al. (1974) as modified by Rychlik et al. (1990) and corrected by Osborne (1992) or a modified version of the algorithm from Khandelwal & Bhyravabhotla (2010) that uses the Rychlik et al. (1990) combining formula. Both algorithms have modifications for use with polymorphic bases. The script uses the thermodynamic constants from Breslauer et al. (1986).

The instructions for creating the script assume that you are using a programming oriented text editor (e.g. Visual Studio Code) that provides line numbers (starting from 1). A complete script is provided at the end of this document for your reference. You should save and test your work often. Small errors can prevent the script from working properly and it is easiest to catch them if you try to run the script frequently.

Tasks

(1) Open a text editor (e.g. Visual Studio Code) and create a blank file.

(2) Enter the first nine lines of the script:

```
1  #!/usr/bin/env python3
2
3  ### IMPORTS
4  import getopt
5  import math
6  import os
7  import re
8  import sys
9  import textwrap
```

Answer question (1).

(3) Save the file as 'otm.py'.

(4) Make the script executable by typing `chmod 0755 otm.py` in the terminal.

(5) Test the script (it does nothing useful) by typing `./otm.py` in the terminal. Not getting an error message means all is well.

(6) Starting at line 11 (line 10 is intentionally blank), insert the following global constants:

```
10
11 ### GLOBAL CONSTANTS
12 BASE2NUMBER = {
13     'A': (0, ),
14     'C': (1, ),
15     'G': (2, ),
16     'T': (3, ),
17     'K': (2, 3),
18     'M': (0, 1),
19     'R': (0, 2),
```

```
    'S': (1, 2),
    'W': (0, 3),
    'Y': (1, 3),
    'B': (1, 2, 3),
    'D': (0, 2, 3),
    'H': (0, 1, 3),
    'V': (0, 1, 2),
    'N': (0, 1, 2, 3)
}
BASE2WEIGHT = {
    'B': 0.6666,
    'C': 1.0000,
    'D': 0.3333,
    'G': 1.0000,
    'H': 0.3333,
    'K': 0.5000,
    'M': 0.5000,
    'N': 0.5000,
    'R': 0.5000,
    'S': 1.0000,
    'V': 0.6666,
    'Y': 0.5000
}
DECIMALS = 1
DNA = re.compile('^[ABCDGHKMNRSTVWY]+$', re.IGNORECASE)
DNAPM = re.compile('^[0-5]{0,1}\.{0,1}[0-9]{0,2}$')
MONOMM = re.compile('^[1-9]|[1-9][0-9]|[1-9][0-9]{2,2}|[1-4][0-9]{3,3}|5000$')
OLIGOS = ('left', 'right', 'template')
WRAP = int(os.popen('stty size', 'r').read().split()[1])
```

(7) Test the script (there is still no useful output). Answer question (2).

(8) Starting at line 50 (line 49 is intentionally blank), insert the following global variables:

```

### GLOBAL USER SETTINGS AND DEFAULTS
settings = {}
settings['dna'] = 1.0 ### oligo DNA concentration (pM)
settings['left'] = '' ### left oligo sequence
settings['mono'] = 50.0 ### monovalent cation concentration (mM)
settings['right'] = '' ### right oligo sequence
settings['template'] = '' ### template sequence
settings['KB'] = False ### use the Khandelwal & Bhyravabhotla algorithm
```

(9) Test the script (there is still no useful output).

(10) Starting at line 59 (line 58 is intentionally blank), insert the following functions:

```

### FUNCTIONS
def eprint(*arguments, **keywordArguments):
    print(*arguments, file = sys.stderr, **keywordArguments)

def sequenceError(x):
    return wrap(f'{x.title()} sequence (required): -{x[0]} ACGT... | --{x}=ACGT...')

def wrap(string, columns = WRAP):
    return '\n'.join(textwrap.wrap(string, columns))

```

```
69  def oligoScore(oligo, matrix): ### modified to work with polymorphic bases
70     score = 0.0
71     for position in range(1, len(oligo)):
72        trailingBase = BASE2NUMBER[oligo[position-1]]
73        leadingBase = BASE2NUMBER[oligo[position]]
74        sum = 0.0
75        for trailing in trailingBase:
76           for leading in leadingBase:
77              sum += matrix[trailing][leading]
78        score += sum/(len(trailingBase)*len(leadingBase))
79     return score
80
81  def KB(oligo, mono = settings['mono'], dna = settings['dna']): ### Khandelwal & Bhyravabhotla
82     return 7.35*(oligoScore(oligo, (
83        (5.0, 10.0,  8.0,  7.0), ### AA,AC,AG,AT
84        (7.0, 11.0, 10.0,  8.0), ### CA,CC,CG,CT
85        (8.0, 13.0, 11.0, 10.0), ### GA,GC,GG,GT
86        (4.0,  8.0,  7.0,  5.0)  ### TA,TC,TG,TT
87     ))/len(oligo)) \
88     + 17.34*math.log(len(oligo), 10) \
89     + 4.96*math.log(mono/1000.0, 10) \
90     + 0.89*math.log(dna/1000000000000.0, 10) \
91     - 25.42
92
93  def RSR(oligo, mono = settings['mono']): ### Rychlik, Spencer, & Rhoads
94     dH = oligoScore(oligo, (
95        (9.1,  6.5,  7.8,  8.6), ### AA,AC,AG,AT
96        (5.8, 11.0, 11.9,  7.8), ### CA,CC,CG,CT
97        (5.6, 11.1, 11.0,  6.5), ### GA,GC,GG,GT
98        (6.0,  5.6,  5.8,  9.1), ### TA,TC,TG,TT
99     ))
100    dS = oligoScore(oligo, (
101       (24.0, 17.3, 20.8, 23.9), ### AA,AC,AG,AT
102       (12.9, 26.6, 27.8, 20.8), ### CA,CC,CG,CT
103       (13.5, 26.7, 26.6, 17.3), ### GA,GC,GG,GT
104       (16.9, 13.5, 12.9, 24.0)  ### TA,TC,TG,TT
105    ))
106    return (-1000.0*dH)/(-1.0*dS - 57.48) \
107    - 273.15 \
108    + 16.6*math.log(mono/1000.0, 10)
```

(11) Test the script after each function is added (there is still no useful output). Answer question (3).

(12) Insert the following lines, starting at 110 (line 109 is intentionally blank), to check the user input, output help information, and check that the required inputs have been given:

```
109
110 ### READ OPTIONS
111 try:
112    arguments, values = getopt.getopt(
113       sys.argv[1:],
114       'd:hkl:m:r:t:',
115       ['dna=', 'help', 'kb', 'left=', 'monovalent=', 'right=', 'template=']
116    )
117 except getopt.error as error:
118    eprint(str(error))
119    sys.exit(2)
120 for argument, value in arguments:
```

```
   if argument in ('-d', '--dna') and re.search(DNAPM, value) and float(value) > 0.0:
      settings['dna'] = float(value)
   elif argument in ('-h', '--help'):
      eprint('')
      eprint(wrap('A Python3 script for computing optimal PCR annealing temperatures using '
         'either the algorithm of Rychlik et al. [1] with Osborne's [2] corrections and '
         'additional modifications for polymorphic sequences (thermodynamic constants from '
         'Breslauer et al. [3]) or the algorithm of Khandelwal & Bhyravabhotla [4] modified '
         'for polymorphic sequences using the Rychlik et al. [1] combining formula (without '
         'Osborne's [2] corrections).'
      ))
      eprint('\nREFERENCES:')
      eprint(wrap('[1] Rychlik, W., W.J. Spencer, & R.E. Rhoads. 1990. Optimization of the '
         'annealing temperature for DNA amplification in vitro. Nucleic Acids Research 18: '
         '6409-6412. https://doi.org/10.1093/nar/18.21.6409'
      ))
      eprint(wrap('[2] Osborne, B.I. 1992. HyperPCR: a Macintosh Hypercard program for the '
         'determination of optimal PCR annealing temperature. CABIOS 8: 83. '
         'https://doi.org/10.1093/bioinformatics/8.1.83'
      ))
      eprint(wrap('[3] Breslauer, K.J., R. Frank, H. Blocker, & L.A. Marky. 1986. Predicting '
         'DNA duplex stability from the base sequence. Proceedings of the National Academy of '
         'Sciences of the United States of America 83: 3746-3750. '
         'https://doi.org/10.1073/pnas.83.11.3746'
      ))
      eprint(wrap('[4] Khandelwal, G. & J. Bhyravabhotla. 2010. A phenomenological model for '
         'predicting melting temperatures of DNA sequences. PLOS ONE 5: e12433. '
         'https://doi.org/10.1371/journal.pone.0012433'
      ))
      eprint('\nOPTIONS:')
      eprint(wrap('Oligo DNA concentration (pM; optional; default = '
         f"{settings['dna']:.{DECIMALS}f} pM): -d x.x | --dna=x.x"
      ))
      eprint(wrap('Use the Khandelwal & Bhyravabhotla [4] algorithm (optional; default = '
         f"{settings['KB']}): -k | --kb"
      ))
      eprint(sequenceError('left'))
      eprint(wrap('Monovalent cation concentration (mM; optional; default = '
         f"{settings['mono']:.{DECIMALS}f} mM): -m x.x | --monovalent=x.x"
      ))
      eprint(sequenceError('right'))
      eprint(f"{sequenceError('template')}\n")
      sys.exit(0)
   elif argument in ('-k', '--kb'):
      settings['KB'] = True
   elif argument in ('-l', '--left') and re.search(DNA, value):
      settings['left'] = value.upper()
   if argument in ('-m', '--monovalent') and re.search(MONOMM, value) and float(value) > 0.0:
      settings['mono'] = float(value)
   elif argument in ('-r', '--right') and re.search(DNA, value):
      settings['right'] = value.upper()
   elif argument in ('-t', '--template') and re.search(DNA, value):
      settings['template'] = value.upper()

### START OR END
for oligo in OLIGOS:
   if not settings[oligo]:
      eprint(sequenceError(oligo))
      sys.exit(2)
```

Test the script. There should now be useful output. Check that each of the user options (flags) works as expected. Answer question (4).

(13) Insert the final block of code starting on line 181 (line 180 is intentionally blank):

```
180
181 ### COMPUTE TM
182 print('\nOUTPUT:')
183 algorithm = 'Khandelwal & Bhyravabhotla' if settings['KB'] else 'Rychlik, Spencer, & Rhoads'
184 print(wrap(f'Tm calculated with a modified {algorithm} algorithm.'))
185 print(wrap(f"Monovalent cation concentration = {settings['mono']:.{DECIMALS}f} mM."))
186 print(wrap(f"Oligo DNA concentration = {settings['dna']:.{DECIMALS}f} pM."))
187
188 tm = {}
189 for oligo in OLIGOS[0:-1]:
190    tm[oligo] = KB(settings[oligo]) if settings['KB'] else RSR(settings[oligo])
191    print(wrap(f"{oligo.title()} oligo Tm = {tm[oligo]:.{DECIMALS}f}°C ({settings[oligo]})."))
192
193 gc = 0.0
194 for base, weight in BASE2WEIGHT.items():
195    gc += settings['template'].count(base)*weight
196 gc /= len(settings['template'])
197 tm['template'] = 41.0*gc \
198 + 16.6*math.log(settings['mono']) \
199 - 675/len(settings['template']) ### log base e contra Rychlik, Spencer, & Rhoads
200 print(wrap(f"Template Tm = {tm['template']:.{DECIMALS}f}°C."))
201
202 tm['optimal'] = min(tm['left'], tm['right'])
203 tm['optimal'] += 0.0 if settings['KB'] else 14.0
204 tm['optimal'] *= 0.3
205 tm['optimal'] += 0.7*tm['template']
206 tm['optimal'] -= 14.9 if settings['KB'] else 22.9
207 print(wrap(f"Optimal combined Tm = {tm['optimal']:.{DECIMALS}f}°C."))
208 print('')
209 sys.exit(0)
```

Test the script. Answer question (5).

(14) Check that the script is correctly calculating the individual and combined T_m values using this example from Rychlik et al. (1990): The oligos are lambda7131 (5′–GATGAGTTCGTGTCCGTACAAC-TGG–3′) and lambda7606 (5′–GGTTATCGAAATCAGCCACAGCGCC–3′) and the sequence is a portion of GenBank NC_001416.1 (5′–GATGAGTTCGTGTCCGTACAACTGGCGTAATCATGGCCCTTCGGGGCCATTGTTT-CTCTGTGGAGGAGTCCATGACGAAAGATGAACTGATTGCCCGTCTCCGCTCGCTGGGTGAACAACTGAACCGTGATGTCAGC-CTGACGGGGACGAAAGAAGAACTGGCGCTCCGTGTGGCAGAGCTGAAAGAGGAGCTTGATGACACGGATGAAACTGCCGGTC-AGGACACCCCTCTCAGCCGGGAAAATGTGCTGACCGGACATGAAAATGAGGTGGGATCAGCGCAGCCGGATACCGTGATTCT-GGATACGTCTGAACTGGTCACGGTCGTGGCACTGGTGAAGCTGCATACTGATGCACTTCACGCCACGCGGGATGAACCTGTG-GCATTTGTGCTGCCGGGAACGGCGTTTCGTGTCTCTGCCGGTGTGGCAGCCGAAATGACAGAGCGCGGCCTGGCCAGAATGC-AATAACGGGAGGCGCTGTGGCTGATTTCGATAACC–3′). At 0.05M monovalent cations, Rychlik et al. (1990) report lambda7131 T_m = 58.6°C; lambda7606 T_m = 65.2°C; and the optimal T_m = 59.8°C. Answer question (6).

Questions (https://forms.gle/hMXYFhpn7ivKicHi9)

(1) For task (2):

   (a) What does the first line do?
   (b) What do the 'import' statements do?

(2) For task (7):

   (a) What sort of data structure is BASE2NUMBER?
   (b) Can a running script modify the value of key 'A'?
   (c) Can the value of DECIMALS be modified by a running script?

(3) For task (11):

   (a) Why does the loop in the oligoScore function start from 1?
   (b) What does the for loop do?
   (c) What thermodynamic value is given to a polymorphic base?
   (d) What effect does monovalent cation concentration have on oligo T_m in the Rychlik et al. (1990) algorithm?

(4) For task (12):

   (a) Where will the help messages be printed?
   (b) How many bases of template sequence must be provided to be considered valid?
   (c) What would happen if the user input two valid oligos and 500 'X' characters as the template?
   (d) How would you add another user option?
   (e) What if a user specifies an invalid option (e.g. '-x')?
   (f) What is the minimum and maximum monovalent cation concentration that will be considered?

(5) For task (13):

   (a) How many digits after the decimal place will be output?
   (b) Why are some of the template counts weighted less than others?
   (c) What effect does the oligo T_m have on the optimal T_m?

(6) For task (14):

   (a) What T_m did your script calculate for each of the oligos?
   (b) What T_m did your script calculate for the template?
   (c) What T_m did your script calculate for the optimal T_m?
   (d) Are your values the same as those reported by Rychlik et al. (1990)? Explain.

Due at the start of class February 28.

Literature cited

Borer, P. N., B. Dengler, I. Tinoco & O. C. Uhlenbeck. 1974. Stability of ribonucleic acid double–stranded helices. Journal of Molecular Biology 86: 843–853.

Breslauer, K. J., R. Frank, H. Blocker & L. A. Marky. 1986. Predicting DNA duplex stability from the base sequence. Proceedings of the National Academy of Sciences of the United States of America 83: 3746–3750.

Khandelwal, G. & J. Bhyravabhotla. 2010. A phenomenological model for predicting melting temperatures of DNA sequences. PLOS ONE 5: e12433.

Osborne, B. I. 1992. HyperPCR: a Macintosh Hypercard program for the determination of optimal PCR annealing temperature. CABIOS 8: 83.

Rychlik, W., W. J. Spencer & R. E. Rhoads. 1990. Optimization of the annealing temperature for DNA amplification *in vitro*. Nucleic Acids Research 18: 6409–6412.

Final otm.py code

```
#!/usr/bin/env python3

### IMPORTS
import getopt
import math
import os
import re
import sys
import textwrap

### GLOBAL CONSTANTS
BASE2NUMBER = {
   'A': (0, ),
   'C': (1, ),
   'G': (2, ),
   'T': (3, ),
   'K': (2, 3),
   'M': (0, 1),
   'R': (0, 2),
   'S': (1, 2),
   'W': (0, 3),
   'Y': (1, 3),
   'B': (1, 2, 3),
   'D': (0, 2, 3),
   'H': (0, 1, 3),
   'V': (0, 1, 2),
   'N': (0, 1, 2, 3)
}
BASE2WEIGHT = {
   'B': 0.6666,
   'C': 1.0000,
   'D': 0.3333,
   'G': 1.0000,
   'H': 0.3333,
   'K': 0.5000,
   'M': 0.5000,
   'N': 0.5000,
   'R': 0.5000,
   'S': 1.0000,
   'V': 0.6666,
   'Y': 0.5000
}
DECIMALS = 1
DNA = re.compile('^[ABCDGHKMNRSTVWY]+$', re.IGNORECASE)
DNAPM = re.compile('^[0-5]{0,1}\.{0,1}[0-9]{0,2}$')
MONOMM = re.compile('^[1-9]|[1-9][0-9]|[1-9][0-9]{2,2}|[1-4][0-9]{3,3}|5000$')
OLIGOS = ('left', 'right', 'template')
WRAP = int(os.popen('stty size', 'r').read().split()[1])

### GLOBAL USER SETTINGS AND DEFAULTS
settings = {}
settings['dna'] = 1.0 ### oligo DNA concentration (pM)
settings['left'] = '' ### left oligo sequence
settings['mono'] = 50.0 ### monovalent cation concentration (mM)
settings['right'] = '' ### right oligo sequence
settings['template'] = '' ### template sequence
```

```
settings['KB'] = False ### use the Khandelwal & Bhyravabhotla algorithm

### FUNCTIONS
def eprint(*arguments, **keywordArguments):
   print(*arguments, file = sys.stderr, **keywordArguments)

def sequenceError(x):
   return wrap(f'{x.title()} sequence (required): -{x[0]} ACGT... | --{x}=ACGT...')

def wrap(string, columns = WRAP):
   return '\n'.join(textwrap.wrap(string, columns))

def oligoScore(oligo, matrix): ### modified to work with polymorphic bases
   score = 0.0
   for position in range(1, len(oligo)):
      trailingBase = BASE2NUMBER[oligo[position-1]]
      leadingBase = BASE2NUMBER[oligo[position]]
      sum = 0.0
      for trailing in trailingBase:
         for leading in leadingBase:
            sum += matrix[trailing][leading]
      score += sum/(len(trailingBase)*len(leadingBase))
   return score

def KB(oligo, mono = settings['mono'], dna = settings['dna']): ### Khandelwal & Bhyravabhotla
   return 7.35*(oligoScore(oligo, (
      (5.0, 10.0,  8.0,  7.0), ### AA,AC,AG,AT
      (7.0, 11.0, 10.0,  8.0), ### CA,CC,CG,CT
      (8.0, 13.0, 11.0, 10.0), ### GA,GC,GG,GT
      (4.0,  8.0,  7.0,  5.0)  ### TA,TC,TG,TT
   ))/len(oligo)) \
   + 17.34*math.log(len(oligo), 10) \
   + 4.96*math.log(mono/1000.0, 10) \
   + 0.89*math.log(dna/1000000000000.0, 10) \
   - 25.42

def RSR(oligo, mono = settings['mono']): ### Rychlik, Spencer, & Rhoads
   dH = oligoScore(oligo, (
      (9.1,  6.5,  7.8,  8.6), ### AA,AC,AG,AT
      (5.8, 11.0, 11.9,  7.8), ### CA,CC,CG,CT
      (5.6, 11.1, 11.0,  6.5), ### GA,GC,GG,GT
      (6.0,  5.6,  5.8,  9.1), ### TA,TC,TG,TT
   ))
   dS = oligoScore(oligo, (
      (24.0, 17.3, 20.8, 23.9), ### AA,AC,AG,AT
      (12.9, 26.6, 27.8, 20.8), ### CA,CC,CG,CT
      (13.5, 26.7, 26.6, 17.3), ### GA,GC,GG,GT
      (16.9, 13.5, 12.9, 24.0)  ### TA,TC,TG,TT
   ))
   return (-1000.0*dH)/(-1.0*dS - 57.48) \
   - 273.15 \
   + 16.6*math.log(mono/1000.0, 10)

### READ OPTIONS
try:
   arguments, values = getopt.getopt(
      sys.argv[1:],
      'd:hkl:m:r:t:',
      ['dna=', 'help', 'kb', 'left=', 'monovalent=', 'right=', 'template=']
```

```
    )
except getopt.error as error:
    eprint(str(error))
    sys.exit(2)
for argument, value in arguments:
    if argument in ('-d', '--dna') and re.search(DNAPM, value) and float(value) > 0.0:
        settings['dna'] = float(value)
    elif argument in ('-h', '--help'):
        eprint('')
        eprint(wrap('A Python3 script for computing optimal PCR annealing temperatures using '
            'either the algorithm of Rychlik et al. [1] with Osborne's [2] corrections and '
            'additional modifications for polymorphic sequences (thermodynamic constants from '
            'Breslauer et al. [3]) or the algorithm of Khandelwal & Bhyravabhotla [4] modified '
            'for polymorphic sequences using the Rychlik et al. [1] combining formula (without '
            'Osborne's [2] corrections).'
        ))
        eprint('\nREFERENCES:')
        eprint(wrap('[1] Rychlik, W., W.J. Spencer, & R.E. Rhoads. 1990. Optimization of the '
            'annealing temperature for DNA amplification in vitro. Nucleic Acids Research 18: '
            '6409-6412. https://doi.org/10.1093/nar/18.21.6409'
        ))
        eprint(wrap('[2] Osborne, B.I. 1992. HyperPCR: a Macintosh Hypercard program for the '
            'determination of optimal PCR annealing temperature. CABIOS 8: 83. '
            'https://doi.org/10.1093/bioinformatics/8.1.83'
        ))
        eprint(wrap('[3] Breslauer, K.J., R. Frank, H. Blocker, & L.A. Marky. 1986. Predicting '
            'DNA duplex stability from the base sequence. Proceedings of the National Academy of '
            'Sciences of the United States of America 83: 3746-3750. '
            'https://doi.org/10.1073/pnas.83.11.3746'
        ))
        eprint(wrap('[4] Khandelwal, G. & J. Bhyravabhotla. 2010. A phenomenological model for '
            'predicting melting temperatures of DNA sequences. PLOS ONE 5: e12433. '
            'https://doi.org/10.1371/journal.pone.0012433'
        ))
        eprint('\nOPTIONS:')
        eprint(wrap('Oligo DNA concentration (pM; optional; default = '
            f"{settings['dna']:.{DECIMALS}f} pM): -d x.x | --dna=x.x"
        ))
        eprint(wrap('Use the Khandelwal & Bhyravabhotla [4] algorithm (optional; default = '
            f"{settings['KB']}): -k | --kb"
        ))
        eprint(sequenceError('left'))
        eprint(wrap('Monovalent cation concentration (mM; optional; default = '
            f"{settings['mono']:.{DECIMALS}f} mM): -m x.x | --monovalent=x.x"
        ))
        eprint(sequenceError('right'))
        eprint(f"{sequenceError('template')}\n")
        sys.exit(0)
    elif argument in ('-k', '--kb'):
        settings['KB'] = True
    elif argument in ('-l', '--left') and re.search(DNA, value):
        settings['left'] = value.upper()
    if argument in ('-m', '--monovalent') and re.search(MONOMM, value) and float(value) > 0.0:
        settings['mono'] = float(value)
    elif argument in ('-r', '--right') and re.search(DNA, value):
        settings['right'] = value.upper()
    elif argument in ('-t', '--template') and re.search(DNA, value):
        settings['template'] = value.upper()

```

```
### START OR END
for oligo in OLIGOS:
   if not settings[oligo]:
      eprint(sequenceError(oligo))
      sys.exit(2)

### COMPUTE TM
print('\nOUTPUT:')
algorithm = 'Khandelwal & Bhyravabhotla' if settings['KB'] else 'Rychlik, Spencer, & Rhoads'
print(wrap(f'Tm calculated with a modified {algorithm} algorithm.'))
print(wrap(f"Monovalent cation concentration = {settings['mono']:.{DECIMALS}f} mM."))
print(wrap(f"Oligo DNA concentration = {settings['dna']:.{DECIMALS}f} pM."))

tm = {}
for oligo in OLIGOS[0:-1]:
   tm[oligo] = KB(settings[oligo]) if settings['KB'] else RSR(settings[oligo])
   print(wrap(f"{oligo.title()} oligo Tm = {tm[oligo]:.{DECIMALS}f}°C ({settings[oligo]})."))

gc = 0.0
for base, weight in BASE2WEIGHT.items():
   gc += settings['template'].count(base)*weight
gc /= len(settings['template'])
tm['template'] = 41.0*gc \
+ 16.6*math.log(settings['mono']) \
- 675/len(settings['template']) ### log base e contra Rychlik, Spencer, & Rhoads
print(wrap(f"Template Tm = {tm['template']:.{DECIMALS}f}°C."))

tm['optimal'] = min(tm['left'], tm['right'])
tm['optimal'] += 0.0 if settings['KB'] else 14.0
tm['optimal'] *= 0.3
tm['optimal'] += 0.7*tm['template']
tm['optimal'] -= 14.9 if settings['KB'] else 22.9
print(wrap(f"Optimal combined Tm = {tm['optimal']:.{DECIMALS}f}°C."))
print('')
sys.exit(0)
```