DNA Strand

1.0

Generated by Doxygen 1.8.16

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 DNAStrand Namespace Reference

Classes

- class DNAStrand

Functions

- def main (args=None)

  Main program for testing.

5.1.1 Detailed Description

Playing with string matching.

Author

of skeleton Paulo Roma, edited by Glauber Faria

Since

10/04/2019

See also

https://www.sciencedirect.com/topics/medicine-and-dentistry/dna-strand

5.1.2 Function Documentation

5.1.2.1 main()

```
def DNAStrand.main (
              args = None )
```

Main program for testing.

Parameters

args	two DNA strands.

```
245 def main(args=None):
246     if args is None:
247         args = sys.argv
248 
249     if len(args) == 5:
250         d = DNAStrand(args[1])
251         d2 = DNAStrand(args[2])
252         ls = int(args[3])
253         rs = int(args[4])
254     else:
255         d = DNAStrand("AGAGCAT")
256         d2 = DNAStrand("TCAT")
257         ls = 2
258         rs = 3
259 
260     print("Complement: %s" % d.createComplement())
261     print("Count A in %s: %d" % (d, d.letterCount('A')))
262     print("%s isValid: %r" % (d, d.isValid()))
263     print("Strand: %s" % d2)
264     print("RightShift: %s, %d = %s" % (d, rs, d2.findMatchesWithRightShift(d, rs)))
265     print("Left Shift: %s, %d = %s" % (d, ls, d2.findMatchesWithLeftShift(d, ls)))
266     print("Maximum Matches: %d" % d.findMaxPossibleMatches(d2))
267     print("Number of matches left shift: %s, %d = %s" % (d2, ls + rs, d.countMatchesWithLeftShift(d2, ls
      + rs)))
268 
269 
```

5.2 DNAStrandTest Namespace Reference

Classes

- class TestDNAStrand

5.2.1 Detailed Description

DNAStrand test class.

Author

Glauber Faria

Since

10/04/2020

Chapter 6

Class Documentation

6.1 DNAStrand.DNAStrand Class Reference

Public Member Functions

- def __init__ (self, givenData)

  Constructs a DNAStrand with the given string of data, normally consisting of characters 'A', 'C', 'G', and 'T'.
- def __str__ (self)

  Returns a string representing the strand data of this DNAStrand.
- def __len__ (self)
- def __add__ (self, other)
- def createComplement (self)

  Returns a new DNAStrand that is the complement of this one, that is, 'A' is replaced with 'T' and so on.
- def findBaseComplement (self, i)
- def findMatchesWithLeftShift (self, other, shift)

  Returns a string showing which characters in this strand are matched with 'other', when shifted left by the given amount.
- def findMatchesWithRightShift (self, other, shift)

  Returns a string showing which characters in this strand are matched with 'other', when shifted right by the given amount.
- def findMatchesOfSameLength (self, fragment1, fragment2)
- def findMaxPossibleMatches (self, other)

  Returns the maximum possible number of matching base pairs, when the given sequence is shifted left or right by any amount.
- def countMatchesWithLeftShift (self, other, shift)

  Returns the number of matching pairs, when 'other' is shifted to the left by 'shift' positions.
- def countMatchesWithRightShift (self, other, shift)

  Returns the number of matching pairs, when 'other' is shifted to the right by 'shift' positions.
- def isValid (self)

  Determines whether all characters in this strand are valid ('A', 'G', 'C', or 'T').
- def letterCount (self, ch)

  Counts the number of occurrences of the given character in this strand.
- def matches (self, c1, c2)

  Returns True if the two characters form a base pair ('A' with 'T' or 'C' with 'G').

Public Attributes

- strand

  Strand of this DNA, in upper case.

Static Public Attributes

- string symbols = 'ATCG'

  Valid DNA symbols.

6.1.1 Constructor & Destructor Documentation

6.1.1.1 __init__()

```
def DNAStrand.DNAStrand.__init__ (
              self,
              givenData )
```

Constructs a DNAStrand with the given string of data, normally consisting of characters 'A', 'C', 'G', and 'T'.

Raises a ValueError exception, in case of an invalid givenData strand.

Parameters

givenData	string of characters for this DNAStrand.

```
26     def __init__(self, givenData):
27
28         self.strand = givenData.upper()
29         #print(givenData)
30         #print(len(givenData))
31
32         # ...
33
```

6.1.2 Member Function Documentation

6.1.2.1 __add__()

```
def DNAStrand.DNAStrand.__add__ (
              self,
              other )
43     def __add__(self, other):
44         return self + other
45
46
```

6.1.2.2 __len__()

```
def DNAStrand.DNAStrand.__len__ (
                self )
```

```
39     def __len__(self):
40         return len(self.strand)
41
```

6.1.2.3 __str__()

```
def DNAStrand.DNAStrand.__str__ (
                self )
```

Returns a string representing the strand data of this DNAStrand.

```
35     def __str__(self):
36         return self.strand
37
```

6.1.2.4 countMatchesWithLeftShift()

```
def DNAStrand.DNAStrand.countMatchesWithLeftShift (
                self,
                other,
                shift )
```

Returns the number of matching pairs, when 'other' is shifted to the left by 'shift' positions.

Parameters

other	given DNAStrand to match with this strand.
shift	number of positions to shift other to the left.

Returns

number of matching pairs.

```
174     def countMatchesWithLeftShift(self, other, shift):
175
176         count = 0
177         shifted_strand = self.findMatchesWithLeftShift(other, shift)
178         count = sum(1 for char in shifted_strand if char.isupper())
179         return count
180
```

6.1.2.5 countMatchesWithRightShift()

```
def DNAStrand.DNAStrand.countMatchesWithRightShift (
                self,
                other,
                shift )
```

Returns the number of matching pairs, when 'other' is shifted to the right by 'shift' positions.

Parameters

other	given DNAStrand to be matched with this one.
shift	number of positions to shift other to the right.

Returns

number of matching pairs.

```
189     def countMatchesWithRightShift(self, other, shift):
190         count = 0
191         shifted_strand = self.findMatchesWithRightShift(other, shift)
192         count = sum(1 for char in shifted_strand if char.isupper())
193         return count
194
```

6.1.2.6 createComplement()

```
def DNAStrand.DNAStrand.createComplement (
                self )
```

Returns a new DNAStrand that is the complement of this one, that is, 'A' is replaced with 'T' and so on.

Returns

complement of this DNA.

```
53     def createComplement(self):
54         complement = ''
55         for i in self.strand:
56             complement += self.findBaseComplement(i)
57         return DNAStrand(complement)
58
```

6.1.2.7 findBaseComplement()

```
def DNAStrand.DNAStrand.findBaseComplement (
                self,
                i )
```

```
60     def findBaseComplement(self, i):
61         if i == 'A':
62             return 'T'
63         elif i == 'T':
64             return 'A'
65         elif i == 'C':
66             return 'G'
67         elif i == 'G':
68             return 'C'
69         else:
70             return ""
```

6.1.2.8 findMatchesOfSameLength()

```
def DNAStrand.DNAStrand.findMatchesOfSameLength (
                self,
                fragment1,
                fragment2 )
```

```
134     def findMatchesOfSameLength(self, fragment1, fragment2):
135         fragment1Match = ''
136         if len(fragment2) != len(fragment1):
137             return ''
138         for i in range(0, len(fragment1), 1):
139             isBaseMatch = self.matches(fragment1[i], fragment2[i])
140             if isBaseMatch == True:
141                 fragment1Match += fragment1[i: i+1]
142             else:
143                 fragment1Match += fragment1[i:i+1].lower()
144         return fragment1Match
145
```

6.1.2.9 findMatchesWithLeftShift()

```
def DNAStrand.DNAStrand.findMatchesWithLeftShift (
                self,
                other,
                shift )
```

Returns a string showing which characters in this strand are matched with 'other', when shifted left by the given amount.

Parameters

other	given DNAStrand.
shift	number of positions to shift other to the left.

Returns

a copy of this strand, where matched characters are upper case and unmatched, lower case.

```
80     def findMatchesWithLeftShift(self, other, shift):
81         matches = ''
82
83         if shift < 0:
84             return matches
85
86         shifted_strand = other.strand
87         original_strand_len = len(self.strand)
88         shifted_strand_len = len(shifted_strand)
89
90         while shifted_strand_len < original_strand_len + shift:
91             shifted_strand += ' '
92             shifted_strand_len = len(shifted_strand)
93
94         for index in range(original_strand_len):
95             if self.matches(self.strand[index], shifted_strand[index + shift]):
96                 matches += self.strand[index]
97             else:
98                 matches += self.strand[index].lower()
99
100         return matches
101
102
```

6.1.2.10 findMatchesWithRightShift()

```
def DNAStrand.DNAStrand.findMatchesWithRightShift (
                self,
                other,
                shift )
```

Returns a string showing which characters in this strand are matched with 'other', when shifted right by the given amount.

Parameters

other	given DNAStrand.
shift	number of positions to shift other to the right.

Returns

a copy of this strand, where matched characters are upper case and unmatched, lower case.

```
112     def findMatchesWithRightShift(self, other, shift):
113         matches = ''
114 
115         if shift < 0:
116             return matches
117 
118         shifted_strand = ' ' * shift + other.strand
119         original_strand_len = len(self.strand)
120         shifted_strand_len = len(shifted_strand)
121 
122         while shifted_strand_len < original_strand_len + shift:
123             shifted_strand += ' '
124             shifted_strand_len = len(shifted_strand)
125 
126         for index in range(original_strand_len):
127             if self.matches(self.strand[index], shifted_strand[index]):
128                 matches += self.strand[index]
129             else:
130                 matches += self.strand[index].lower()
131 
132         return matches
133 
```

6.1.2.11 findMaxPossibleMatches()

```
def DNAStrand.DNAStrand.findMaxPossibleMatches (
                self,
                other )
```

Returns the maximum possible number of matching base pairs, when the given sequence is shifted left or right by any amount.

Parameters

other	given DNAStrand to be matched with this one.

Returns

maximum number of matching pairs.

```
153     def findMaxPossibleMatches(self, other):
154         lenCompare = min(len(self.strand), len(other))
155         leftShiftMaxMatches = 0
156         rightShiftMaxMatches = 0
157 
158         for i in range(0, lenCompare-1, 1):
159             leftShiftMaxMatches = max(leftShiftMaxMatches, self.countMatchesWithLeftShift(other, i))
160             rightShiftMaxMatches = max(rightShiftMaxMatches, self.countMatchesWithRightShift(other, i))
161 
162         maxPossibleMatches = max(leftShiftMaxMatches, rightShiftMaxMatches)
163         return maxPossibleMatches
164 
165 
```

6.1.2.12 isValid()

```
def DNAStrand.DNAStrand.isValid (
                self )
```

Determines whether all characters in this strand are valid ('A', 'G', 'C', or 'T').

Returns

True if valid, and False otherwise.

```
201     def isValid(self):
202         for i in self.strand:
203             if i in self.symbols:
204                 valid = True
205             else:
206                 valid = False
207 
208         # ...
209 
210         return valid
211 
```

6.1.2.13 letterCount()

```
def DNAStrand.DNAStrand.letterCount (
                self,
                ch )
```

Counts the number of occurrences of the given character in this strand.

Parameters

ch	given character.

Returns

number of occurrences of ch.

```
218     def letterCount(self, ch):
219         count = 0
220         for i in self.strand:
221             if (i == ch):
222                 count += 1
223         return count
```

```
224
```

6.1.2.14 matches()

```
def DNAStrand.DNAStrand.matches (
                self,
                c1,
                c2 )
```

Returns True if the two characters form a base pair ('A' with 'T' or 'C' with 'G').

Parameters

c1	first character.
c2	second character.

Returns

True if they form a base pair, and False otherwise.

```
233     def matches(self, c1, c2):
234         if c1 == 'A' and c2 =='T' or c1 == 'T' and c2 =='A' or c1 == 'C' and c2 =='G' or c1 == 'G' and
        c2 =='C':
235             match = True
236         else:
237             match = False
238         return match
239
240
```

6.1.3 Member Data Documentation

6.1.3.1 strand

```
DNAStrand.DNAStrand.strand
```

Strand of this DNA, in upper case.

6.1.3.2 symbols

```
string DNAStrand.DNAStrand.symbols = 'ATCG'  [static]
```

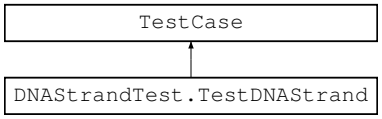
Valid DNA symbols.

The documentation for this class was generated from the following file:

- DNAStrand.py

6.2 DNAStrandTest.TestDNAStrand Class Reference

Inheritance diagram for DNAStrandTest.TestDNAStrand:



Public Member Functions

- def test_createComplement (self)
- def test_is_Valid (self)
- def test_countMatchesWithLeftShif (self)
- def test_findMatchesWithLeftShift (self)
- def test_countMatchesWithRightShift (self)
- def test_findMatchesWithRightShift (self)
- def test_findMaxPossibleMatches (self)
- def test_letterCoun (self)
- def test_matches (self)
- def test_matches_notvalid (self)

Static Public Attributes

- d1 = DNAStrand("TCAT")
- d2 = DNAStrand("AGAGCAT")
- string m = 'A'
- int ls = 2
- int rs = 3

6.2.1 Member Function Documentation

6.2.1.1 test_countMatchesWithLeftShif()

```
def DNAStrandTest.TestDNAStrand.test_countMatchesWithLeftShif (
                self )
37     def test_countMatchesWithLeftShif(self):
38         msg = 'countMatchesWithLeftShift() should return the value of matches in two DNA strands'
39         value1 = 1
40         value2 = 1
41         self.assertEqual(self.d1.countMatchesWithLeftShift(self.d2, 5), value1, msg)
42         self.assertEqual(self.d2.countMatchesWithLeftShift(self.d1, 3), value2, msg)
43
```

6.2.1.2 test_countMatchesWithRightShift()

```
def DNAStrandTest.TestDNAStrand.test_countMatchesWithRightShift (
                self )
```

```
51     def test_countMatchesWithRightShift(self):
52         msg = 'countMatchesWithRightShift() should return the value of matches in two DNA strands'
53         value1 = 1
54         value2 = 2
55         self.assertEqual(self.d1.countMatchesWithRightShift(self.d2, 3), value1, msg)
56         self.assertEqual(self.d2.countMatchesWithRightShift(self.d1, 0), value2, msg)
57
```

6.2.1.3 test_createComplement()

```
def DNAStrandTest.TestDNAStrand.test_createComplement (
                self )
```

```
25     def test_createComplement(self):
26         msg = 'createComplement() should return a DNAStrand complement'
27         dna_strand1 = 'AGTA'
28         dna_strand2 = 'TCTCGTA'
29         self.assertEqual(self.d1.createComplement().strand, dna_strand1, msg)
30         self.assertEqual(self.d2.createComplement().strand, dna_strand2, msg)
31
```

6.2.1.4 test_findMatchesWithLeftShift()

```
def DNAStrandTest.TestDNAStrand.test_findMatchesWithLeftShift (
                self )
```

```
44     def test_findMatchesWithLeftShift(self):
45         msg = 'findMatchesWithLeftShift() should return a dna string with capitalized matching
       characters'
46         dna_strand1 = 'Tcat'
47         dna_strand2 = 'agAgcat'
48         self.assertEqual(self.d1.findMatchesWithLeftShift(self.d2, 5), dna_strand1, msg)
49         self.assertEqual(self.d2.findMatchesWithLeftShift(self.d1, 1), dna_strand2, msg)
50
```

6.2.1.5 test_findMatchesWithRightShift()

```
def DNAStrandTest.TestDNAStrand.test_findMatchesWithRightShift (
                self )
```

```
58     def test_findMatchesWithRightShift(self):
59         msg = 'findMatchesWithRightShift() should return a dna string with capitalized matching
       characters'
60         dna_strand1 = 'tcaT'
61         dna_strand2 = 'agAGcAt'
62         self.assertEqual(self.d1.findMatchesWithRightShift(self.d2, 1), dna_strand1, msg)
63         self.assertEqual(self.d2.findMatchesWithRightShift(self.d1, 2), dna_strand2, msg)
64
```

6.2.1.6 test_findMaxPossibleMatches()

```
def DNAStrandTest.TestDNAStrand.test_findMaxPossibleMatches (
                self )
```

```
65     def test_findMaxPossibleMatches(self):
66         msg = 'findMaxPossibleMatches() should return the maximum value of matches in two DNA strands'
67         value = 3
68         self.assertEqual(self.d1.findMaxPossibleMatches(self.d2), value, msg)
69         self.assertEqual(self.d2.findMaxPossibleMatches(self.d1), value, msg)
70
```

6.2.1.7 test_is_Valid()

```
def DNAStrandTest.TestDNAStrand.test_is_Valid (
                self )
```

```
32     def test_is_Valid(self):
33         msg = 'isValid() should return True'
34         self.assertTrue(self.d1.isValid(), msg)
35         self.assertTrue(self.d2.isValid(), msg)
36
```

6.2.1.8 test_letterCoun()

```
def DNAStrandTest.TestDNAStrand.test_letterCoun (
                self )
```

```
71     def test_letterCoun(self):
72         msg = 'letterCount() should return the number of times that a letter appears in a DNA strand'
73         value1 = 1
74         value2 = 2
75         self.assertEqual(self.d1.letterCount('C'), value1, msg)
76         self.assertEqual(self.d2.letterCount('G'), value2, msg)
77
```

6.2.1.9 test_matches()

```
def DNAStrandTest.TestDNAStrand.test_matches (
                self )
```

```
78     def test_matches(self):
79         msg = 'matches() should return True'
80         self.assertTrue(self.d1.matches('A', 'T'), msg)
81         self.assertTrue(self.d2.matches('C', 'G'), msg)
82
```

6.2.1.10 test_matches_notvalid()

```
def DNAStrandTest.TestDNAStrand.test_matches_notvalid (
                self )
```

```
83     def test_matches_notvalid(self):
84         msg = 'matches() should return False'
85         self.assertFalse(self.d1.matches('A', 'C'), msg)
86         self.assertFalse(self.d2.matches('G', 'T'), msg)
87
88
```

6.2.2 Member Data Documentation

6.2.2.1 d1

```
DNAStrandTest.TestDNAStrand.d1 = DNAStrand("TCAT")  [static]
```

6.2.2.2 d2

```
DNAStrandTest.TestDNAStrand.d2 = DNAStrand("AGAGCAT")  [static]
```

6.2.2.3 ls

```
int DNAStrandTest.TestDNAStrand.ls = 2  [static]
```

6.2.2.4 m

```
string DNAStrandTest.TestDNAStrand.m = 'A'  [static]
```

6.2.2.5 rs

```
int DNAStrandTest.TestDNAStrand.rs = 3  [static]
```

The documentation for this class was generated from the following file:

- DNAStrandTest.py

Chapter 7

File Documentation

7.1 DNAStrand.py File Reference

Classes

- class DNAStrand.DNAStrand

Namespaces

- DNAStrand

Functions

- def DNAStrand.main (args=None)

  Main program for testing.

7.2 DNAStrandTest.py File Reference

Classes

- class DNAStrandTest.TestDNAStrand

Namespaces

- DNAStrandTest

Index