PROJEKT E 14. 11.2018

TERMIN ROZLICZENIA 26.11.2018

WYSZUKIWANIE MOTYWU:
ALGORYTM WYCZERPUJĄCY,
ALGORYTM ZACHŁANNY
ALGORYTM LOSOWY

PYTHON:
OBSŁUGA BŁĘDÓW I WYJĄTKÓW

Wyjątki(exceptions): zdarzenia modyfikujące przebieg programu.

```
"""
Przykłady poleceń powodujących wyjątki.
"""
# ZeroDivisionError :  dzielenie przez zero
print(23/0)

#ValueError :  niewłaściwa wartość
n=int(”3.14”)
print(n)

# IndexError :  odwołanie się do nieistniejącego elementu listy .
alist = []                  # pusta lista
print( alist[5])

# KeyError :  odwołanie się do nieistniejącego klucza w słowniku .
adict = {}                  # pusty słownik
print( adict["key"])

# IOError :  otwarcie do czytania nieistniejącego pliku .
afile = open("nie_istnieje.txt", "r")
```

Kiedy pojawia się błąd w czasie wykonywania programu, tworzony jest wyjątek. Zwykle program wtedy jest zatrzymywany, a Python wypisuje komunikat o błędzie. Tak działa domyślny program obsługi wyjątków.

Jeśli nie chcemy, by program zatrzymał się po wystąpieniu wyjątku, to należy opakować wywołanie w instrukcję

try/except/else

aby samodzielnie zarządzać przerwaniem.
Ponadto, jeżeli zależy nam na wykonaniu pewnych działań końcowych, niezależnych od wystąpienia wyjątku, to stosujemy instrukcję try/except/finally.

0_wyjatki.py

Wyjątki: obsługa

```
try:
    print(23/0)
    #print(23/1)
except ZeroDivisionError:
    print("dzielenie przez zero, pomijam")
else:
    print("dzielenie OK", end='\t')
print("pracuje dalej")
```

1_wyjatki.py

```
try:
    #afile = open("nie_istnieje.txt", "r")
    afile = open("0_wyjatki.py", "r")
    text=afile.read()
except IOError:
    print("nie ma takiego pliku")
    text=[]
finally:
    afile.close()
print(text)
```

2wyjatki.py

Ogólny format instrukcji :

try/except/else/finally

zawiera wiele opcjonalnych bloków z programami obsługi, choć musi pojawić się przynajmniej jeden.

```
try:
    instrukcje                          #podstawowe działanie instrukcji
except Exception1:                      #przechwytuje wskazany wyjątek
    instrukcje
except (Exception2, Exception3): #przechwytuje wymienione wyjątki
    instrukcje
except Exception4 as Value1:        # przechwytuje wyjątek i jego instancję
    instrukcje
except (Exception4, Exception5) as Value2 : #przechwytuje wyjątki i instancję
    instrukcje
except:                                 #przechwytuje wszystkie (pozostałe) wyjątki
    instrukcje
else:                                   #działania przy braku zgłoszenia wyjątku
    instrukcje
finally:                                #działania końcowe
    instrukcje
```

Algorytmy wyszukiwania motywu: wyczerpujący, zachłanny i losowy

BRUTEFORCEMOTIFSEARCH(DNA, t, n, l)
1 $bestScore \leftarrow 0$
2 **for each** $(s_1, \ldots, s_t)$ **from** $(1, \ldots, 1)$ **to** $(n-l+1, \ldots, n-l+1)$
3 **if** $Score(\mathrm{s}, DNA) > bestScore$
4 $bestScore \leftarrow Score(\mathrm{s}, DNA)$
5 $\mathrm{bestMotif} \leftarrow (s_1, s_2, \ldots, s_t)$
6 **return** bestMotif

GREEDYMOTIFSEARCH(*Dna*, *k*, *t*)
BestMotifs ← motif matrix formed by first *k*-mers in each string from *Dna*
for each *k*-mer *Motif* in the first string from *Dna*
$Motif_1 \leftarrow Motif$
for $i = 2$ to *t*
form *Profile* from motifs $Motif_1, \ldots, Motif_{i-1}$
$Motif_i \leftarrow$ *Profile*-most probable *k*-mer in the *i*-th string in *Dna*
$Motifs \leftarrow (Motif_1, \ldots, Motif_t)$
if SCORE(*Motifs*) < SCORE(*BestMotifs*)
BestMotifs ← *Motifs*
return *BestMotifs*

RANDOMIZEDMOTIFSEARCH(*Dna*, *k*, *t*)
randomly select *k*-mers $Motifs = (Motif_1, \ldots, Motif_t)$ in each string from *Dna*
BestMotifs ← *Motifs*
while forever
Profile ← PROFILE(*Motifs*)
Motifs ← MOTIFS(*Profile*, *Dna*)
if SCORE(*Motifs*) < SCORE(*BestMotifs*)
BestMotifs ← *Motifs*
else
return *BestMotifs*

Obliczenie jakości uzgodnionego najlepiej łańcucha(consensus)

2. Wektor pozycji startowych wstawek w łańcuchach DNA : **s**=(s1,s2,…,st)

```
                                        CGGGGCTATcCAgCTGGGTCGTCACATTCCCCTT...
                                TTTGAGGGTGCCCAATAAggGCAACTCCAAAGCGGACAAA
                                        GGATGgAtCTGATGCCGTTTGACGACCTA...
                                    AAGGAaGCAACcCCAGGAGCGCCTTTGCTGG...
                            AATTTTCTAAAAAGATTATAATGTCGGTCCtTGgAACTTC
                        CTGCTGTACAACTGAGATCATGCTGCATGCcAtTTTCAAC
                                TACATGATCTTTTGATGgcACTTGGATGAGGGAATGATGC
```

Przypomnienie z wykładu:

3. Macierz dopasowania dla danego **s**:

Alignment	A	T	C	C	A	G	C	T
	G	G	G	C	A	A	C	T
	A	T	G	G	A	T	C	T
	A	A	G	C	A	A	C	C
	T	T	G	G	A	A	C	T
	A	T	G	C	C	A	T	T
	A	T	G	G	C	A	C	T

4. Macierz profilu **P(s)** dla danego **s**:

$P(s)$

Profile	**A**	5	1	0	0	5	5	0	0
	T	1	5	0	0	0	1	1	6
	G	1	1	6	3	0	1	0	0
	C	0	0	1	4	2	0	6	1
Consensus		A	T	G	C	A	A	C	T

5. Uzgodniony łańcuch profilu dla **s**

$M_{P(s)}(j)$ **5 5 6 4 5 5 6 6**

$$Score(s, DNA) = \sum_{j=1,\ldots,l} M_{P(s)}(j)$$

PROJEKT E

14. 11.2018

Niech dany będzie zestaw 10 sekwencji DNA .

```
'tagtggtcttttgagtgtagatctgaagggaaagtatttccaccagttcggggtcacccagcagggcagggtgacttaat',
'cgcgactcggcgctcacagttatcgcacgtttagaccaaaacggagttggatccgaaactggagtttaatcggagtcctt',
'gttacttgtgagcctggttagacccgaaatataattgttggctgcatagcggagctgacatacgagtaggggaaatgcgt',
'aacatcaggctttgattaaacaatttaagcacgtaaatccgaattgacctgatgacaatacggaacatgccggctccggg',
'accaccggataggctgcttattaggtccaaaaggtagtatcgtaataatggctcagccatgtcaatgtgcggcattccac',
'tagattcgaatcgatcgtgttttctcccctctgtgggttaacgaggggtccgaccttgctcgcatgtgccgaacttgtaccc',
'gaaatggttcggtgcgatatcaggccgttctcttaacttggcggtgcagatccgaacgtctctggaggggtcgtgcgcta',
'atgtatactagacattctaacgctcgcttattggcggagaccatttgctccactacaagaggctactgtgtagatccgta',
'ttcttacacccttctttagatccaaacctgttggcgccatcttctttttcgagtccttgtacctccatttgctctgatgac',
'ctacctatgtaaaacaacatctactaacgtagtccggtctttcctgatctgccctaacctacaggtcgatccgaaattcg'
```

Znaleźć w nich motyw k-merowy, możliwie najdłuższy, metodami

- BruteForceMotifSearch(),
- GreedyMotifSearch(),
- RandomMotifSearch()

W implementacji wykorzystać załączoną funkcję Score(). Przy czym

- podać szczegółowe wyjaśnienia każdej instrukcji .
- zaproponować równoważne rozwiązanie uzyskane innymi operacjami Pythona
- opracować wiele (co najmniej 10) testów ujawniających przebieg działania funkcji

Wszystkie funkcje powinny być oddzielnymi modułami z dobrze przygotowaną sekcją testów.

uzupełnienia

```
def Score(s, DNA, k):
    """
    compute the consensus SCORE of a given k-mer alignment given
    offsets into each DNA string.
    s = list of starting indices.
    DNA = list of nucleotide strings.
    k = Target Motif length
    """

    score = 0
    for i in range(k):
        # loop over string positions
        cnt = dict(zip("acgt",(0,0,0,0)))
        for j, s_val in enumerate(s):
            #loop over DNA strands
            base = DNA[j][s_val+i]
            cnt[base] += 1
        score += max(cnt.values())
    return score
```