Oefententamen

Tentamencode: BFVP15INF2

Fenna Feenstra en Jurre Hageman

16/01/2017

Inloggen/bestanden

Voor dit tentamen moet je ingelogd zijn als gast (password gast). Op de Desktop staan de volgende bestanden/links die je nodig hebt of kunt gebruiken om dit tentamen te maken:

1. een link naar HTML PyDoc
2. de Python cheat sheet
3. het script submit_your_work waarmee je de gemaakte opgaven inlevert.
4. het bestand opgave2.py dat je moet verbeteren.

Opgaven

Het tentamen bevat een aantal opgaven. Iedere opdracht resulteert in een apart Python file. Begin elk script met de specificatie van de python3 interpreter (hashbang). Schrijf daarna je naam in het script, als metavariabele.

TIP: Programmeer per stap en kijk steeds wat er gebeurt. Gebruik IDLE3 of Python3 in de terminal om stukjes code uit te proberen.

Indienen

Het gemaakte werk kan ingeleverd worden door het script submit_your_work uit te voeren in de directory waar je het werk gemaakt hebt, als:

```
~/Desktop/submit_your_work STUDENTNR opg1.py opg2.py ...
```

Geef alle files gescheiden met een spatie. Hierna kun je bij de surveillant controleren of het indienen gelukt is.

Beoordeling

Bij elke opgave staat vermeld hoeveel punten maximaal te krijgen zijn. De opgaven worden beoordeeld op de volgende punten:

1. Werkt de code?
2. Doet de code wat het moet doen?
3. Is de code goed (leesbaar, effectief)?

Als een stuk code niet werk, maak er commentaar van en schrijf er in commentaar bij wat je bedoelt.

Het cijfer wordt als volgt bepaald: behaald aantal punten/totaal aantal punten*9+1

LEES DE OPGAVEN GOED DOOR! Succes!

Opgave 1 [30 punten]

Opgave 1 is te vinden als opgave1.py. Pas deze file aan op basis van onderstaande opgave. Let op: de output van de code hieronder begint steeds met ##.

a) Onderstaande functie maakt gebruik van een global statement om text te veranderen binnen de functie. Krijg de functie werkend zonder een global statement. Zorg er dus voor dat geen global statement gebruikt wordt. Hierbij moet de variabele text wel een globale variabele blijven en niet binnen de functie gedeclareerd worden.

```
text = "hello"


def modify_text():
    global text
    text = text + " you" #deze regel mag niet veranderd worden
    return text #deze regel mag niet veranderd worden

print(modify_text())
```

b) print zonder "loop" de inhoud van de sequences list. Gebruik maar 1 regel code. Gewenste output: ATGTTGTGCC ATGTTGTGCC ATGTTGTGTGCC ATGTTGCC

```
sequences = ["ATGTTGTGCC", "ATGTTGTGCC", "ATGTTGTGTGCC", "ATGTTGCC"]
```

c) Herschrijf onderstaande code die een file opent en sluit naar code die het zelfde doet maar dan het with statement gebruikt.

```
sequences = ["ATGTTGTGCC", "ATGTTGTGCC", "ATGTTGTGTGCC", "ATGTTGCC"]


fo = open('foo.txt', 'w')
for seq in sequences:
    fo.write(seq+ '\n')
fo.flush()
fo.close()

fo = open('foo.txt', 'r')
for line in fo:
    print(line)
```

```
fo.flush()
fo.close()
```

d) Verbeter de code volgens de docstring in de functie timestamp()

```
#IMPORTS
import time

#FUNCTIONS
def timestamp():
    """ get a time stamp according format dd-mm-yy hh:mm:ss """
    return time.ctime()

print(timestamp())

## Mon Jan 16 14:24:47 2017
```

e) Maak een lijst waarbij de elementen van onderstaande lijst omgezet worden in hoofdletters. Het resultaat moet zijn: ['AATC', 'ACGGG', 'AUTGC', 'CCC', 'ACUCC'] Het is de bedoeling dat je de lijst daadwerkelijk gebruikt en dat je map of filter gebruikt (kies zelf welk van deze twee functies geschikt is). Combineer dit met een lambda functie. De code hiervoor moet 1 regel beslaan.

```
nucleotide_list = ["aatc", "acggg", "autgc", "ccc", "acucc"]
new_list = "hier je antwoord"
print(new_list)
```

f) Maak een lijst waarbij de elementen van bovenstaande lijst (nucleotide_list) omgezet worden in hoofdletters. Daarnaast mogen elementen van de lijst met een u niet in de nieuwe lijst komen. Het resultaat moet zijn: ['AATC', 'ACGGG', 'CCC'] Het is de bedoeling dat je de lijst daadwerkelijk gebruikt en dat je nu een LIST COMPREHENSION gebruikt. De code hiervoor moet 1 regel beslaan.

```
nucleotide_list = ["aatc", "acggg", "autgc", "ccc", "acucc"]
new_list = "hier je antwoord"
print(new_list)
```

Opgave 2 [30 punten]

Opgave 2 is te vinden als opgave2.py. Pas deze file aan op basis van onderstaande opgave.

In de file opgave2.py vind je een programma dat een bisulfiet simulatie uitvoert op DNA. Door DNA met bisulfiet te behandelen is te onderzoeken of DNA gemethyleerd is.

Bisulfiet maakt van elke cytosine (C) een thymine (T) behalve als een cytosine gemethyleerd is geweest.

DNA wordt alleen gemethyleerd op CG dinucleotiden.

Voor simulatie van bisulfiet behandeling van ongemethyleerd DNA dienen dus alle C's (cytosines) omgezet te worden naar een T (Thymine).

Een voorbeeld: De volgorde ACGACA wordt als volgt omgezet: Indien NIET gemethyleerd: ATGATA Indien WEL gemethyleerd: ACGATA

Voor de simulatie van bisulfiet behandeling van gemethyleerd DNA wordt het volgende gedaan: - eerst wordt elke CG in de sequentie omgezet naar XG - dan wordt elke C omgezet naar T - daarna wordt elke X omgezet naar een C

De python file van opgave 2 werkt maar is niet netjes geschreven. Maak van dit script een net script.

- Gebruik docstrings waar nodig
- Roep documentatie aan als er geen sequentie als commandline argument gegeven is
- Zet de code in twee functies: get_bisulphite_unmethylated en get_bisulphite_methylated
- Zet deze functie in een aparte module en noem deze: bisulphiter. Importeer bisulphiter in je main module (opgave2.py).
- Definieer in de main module een main functie en laat deze alleen uitvoeren als de module daadwerkelijk als main module gerund wordt.

De code van opgave 2 vind je dus in opgave2.py maar is hier ook gegeven. Dit hoef je dus niet over te typen. Let op: de output van de code hieronder begint steeds met ##.

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


#get sequence and make upper case. Replace with code that catches command-
line arguments.
seq = "acgaca".upper()

#replace all C's for T's in unmethylated state
seq_unmethylated = seq.replace("C", "T")

#for the methylated state first change CG to XG
seq_methylated_mod_cg = seq.replace("CG", "XG")
#now change all remaining C's to T's
seq_methylated_mod_c = seq_methylated_mod_cg.replace("C", "T")
#Replace all X's back to C's
seq_methylated_mod_x = seq_methylated_mod_c.replace("X", "C")
#print the results
print("Original sequence:", seq)
print("Unmethylated DNA after bisulphite treatment:", seq_unmethylated)
print("Methylated DNA after bisulphite treatment:", seq_methylated_mod_x)

## ('Original sequence:', 'ACGACA')
## ('Unmethylated DNA after bisulphite treatment:', 'ATGATA')
## ('Methylated DNA after bisulphite treatment:', 'ACGATA')
```

Opgave 3 [30 punten]

Opgave 3 is te vinden als opgave3.py. Pas deze file aan op basis van onderstaande opgave. Maak de module af (zie docstring)

```
"""
a list of dna (variable length) is transcribed to rna by means of a
generator function
a print function prints each transcribed dna from the list (each rna string
on a new line)
in case of a StopIteration error it prints the following message
"calling the generator X times: out of range" where X is the length of the
list + 1
"""


def main():
    pass

main()
```