
**Софийски университет „Св. Кл. Охридски”**

Факултет по математика и информатика

*Магистърска програма*
*„Био- и медицинска информатика”*


# КУРСОВ ПРОЕКТ

*Линукс и езици за програмиране в биоинформатиката*

## Тема „RESTfull приложение на Python“

*Петър Андреев, ФN№ 26271*

*Специалност: Био- и медицинска информатика*

Февруари, 2021

София

# Съдържание

1. **Задача**. Да се разработи RESTFul приложение на Python, което да оперира върху данни от ensambl:

    1.1. Да приема id на gene и да връща като резултат цялата секвенция на гена и всички екзони като отделни свойства.

Пример:


```
{

"seq":"GTAGTTGATCTCCGGAGTTTCGCCATGCGGAACTTGGGGGCTTTCGCGGCCCGCGTCGGTGCGGAGTAGCTGCTTT",
    "exons":[
        {"start":38327340,"end":38327509,"id":"ENSE00003819676","seq":"TCCGGAGTTTC"},

{"start":38727340,"end":38327509,"id":"ENSE00003819676","seq":"GGTGCGGAGTAGC"},

{"start":38318829,"end":38318987,"id":"ENSE00001025712","seq":"GAACTTGGGGGCTTTCGCGGCC
CGCGTCGGTGC"}
    ]
}
```


    1.2. Да приема id на gene и да връща като резултат секвенцията с пресметнато процентното GC съдържание и разменени аминокиселини аденин с тимин

    1.3. Да приема ID и да връща секвенция в специфичен формат. Поддържаните формати трябва да са: fasta, multi-fasta, x-fasta

## 2. Материали и методи

Ensambl – геномен браузър (също публична база данни с геномна информация), научен проект на Европейския Институт по Биоинформатика

REST (Representational State Transfer) – неформален стандарт за софтуерна архитектура който осигурява оперативна съвместимост между компютърни системи които предоставят различни услуги

Python – интерпретативен език за програмиране от високо ниво с общо предназначение

Visual Studio Code – интегрирана среда за разработка

## 3. Резултати и дискусия

### 3.1. Секвенция на ген и списък на екзоните с техните секвенциии.

В ensemble не съществува REST endpoint който да може да изведе директно секвенциите на екзоните при зададено ID на ген. За да се получи изход подобен на описания в заданието, задачата трябва да се раздели на две части.

Първо, чрез обикновена get заявка от следния endpoint: https://rest.ensembl.org/sequence/id/geneID? се взема само пълната секвенция на гена по този начин:

```
geneID = "ENSG00000157764"

server = "https://rest.ensembl.org"
ext = "/sequence/id/" + geneID + "?"

r = requests.get(server+ext, headers={ "Content-Type" : "text/plain"})

if not r.ok:
  r.raise_for_status()
  sys.exit()
```

Като задаваме "Content-Type" : "text/plain", правим възможно извеждане само на секвенцията в текстов вид (възможно е задаване на x-fasta , като така се изпълняват частично изискванията по т. 2.3. от заданието).

Второ, по-надолу в изходния код на приложението разработваме логиката, която да изведе списъка с екзоните плюс тяхната секвенция както е в примера на заданието.

С втора get заявка на https://rest.ensembl.org/lookup/id/ и включена expand опция събираме IDs (ensemble идентификаторите) на всички екзони и транскрипти, по този начин:

```
def fetch_endpoint(server, request, content_type):

    r = requests.get(server+request, headers={ "Accept" : content_type})

    if not r.ok:
        r.raise_for_status()
        sys.exit()

    if content_type == 'application/json':
        return r.json()
    else:
```

```
        return r.text
```

Получават се списъци с ID на екзони и транскрипти (както и друга информация за съставните на съответния ген) в json format. При това информацията е така структурирана, че най-напред се получава списък от транскриптите на гена (всеки един е резултат от алтернативен сплайсинг на екзоните), а към всеки транскрипт има списък с ID на съставните му екзони. Поради тази причина, изхода от тази заявка не може да се използва директно, а трябва първо да се обходи с два for цикъла за да се съберат идентификаторите първо на транскриптите и след това на екзоните, които влизат в състава на всеки транскрипт:

```
# for each transcript get a list of the exon ids
for transcript in get_gene['Transcript']:
    print (">", transcript['id'])
    exons = []
    for exon in transcript['Exon']:
        exons.append(exon['id'])
```

След това, чрез метода dumps получаваме списък от json обекти само с идентификаторите на екзоните в дадения ген.

```
    # create a json dump of the exon ids
    exon_json = json.dumps({ "ids" : exons })
```

Този списък използваме след това за да извлечем секвенциите на всеки екзон, чрез пост метод по следния начин:

```
def fetch_endpoint_POST(server, request, data, content_type):

    r = requests.post(server+request,
                      headers={ "Accept" : content_type},
                      data=data )

    if not r.ok:
        r.raise_for_status()
        sys.exit()

    if content_type == 'application/json':
        return r.json()
    else:
        return r.text

server = "http://rest.ensembl.org/"
ext_gene = "lookup/id/" + geneID + "?expand=1"
con_json = "application/json"
```

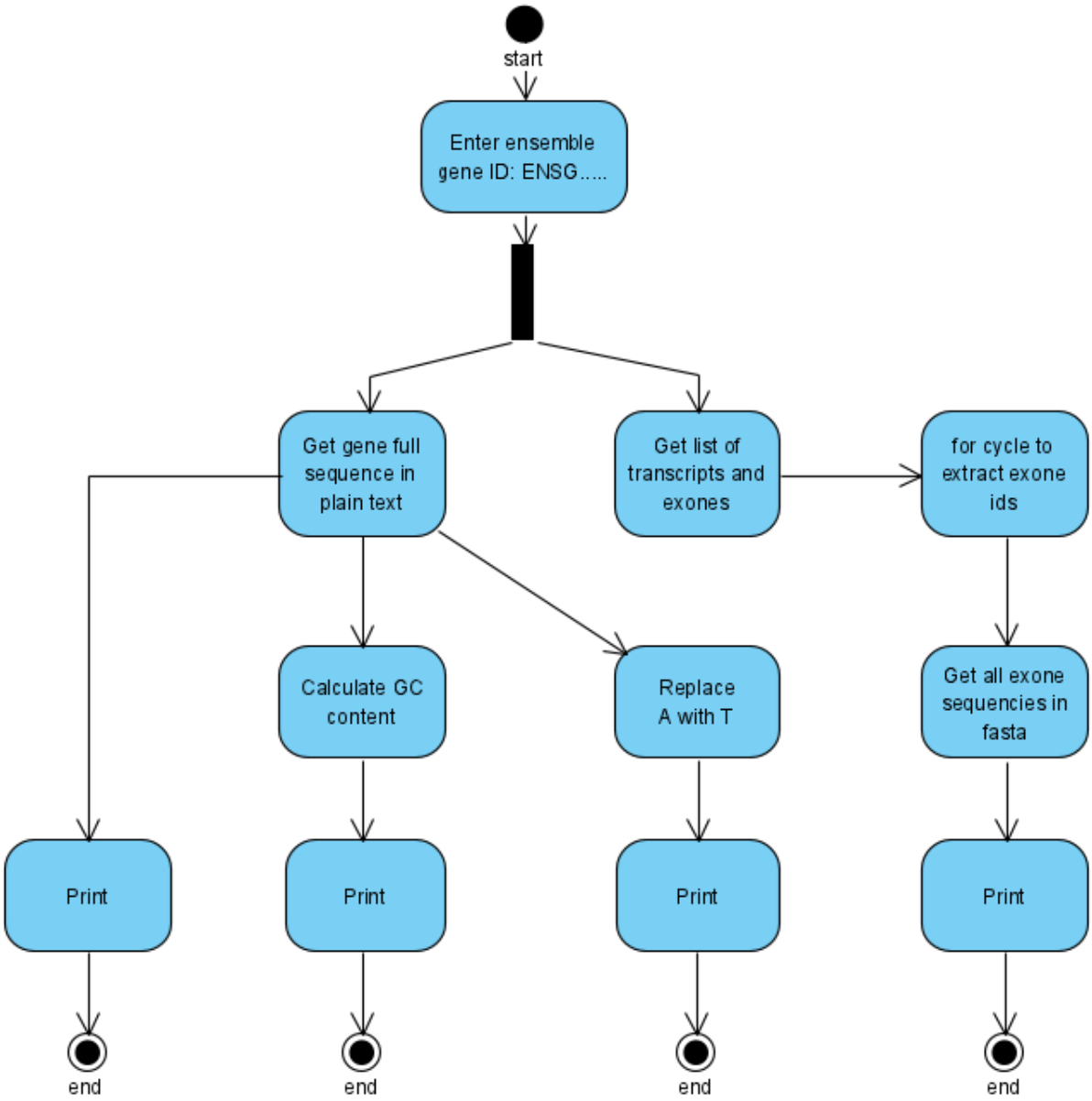
```
ext_sequence = "/sequence/id/"
con_seq = "text/x-fasta"
get_gene = fetch_endpoint(server, ext_gene, con_json)
```

3.2. Пресмятане на процентното GC съдържание на секвенция по зададено id и замяна на нуклеотиди аденин с тимин в секвенцията – работи с резултата от първата заявка с която е изведена секвенцията на гена в текстов вид (виж по-горе)

```
# calculates GC content

geneSequence = r.text

elements = Counter(geneSequence)

a = (((elements["G"] + elements["C"]) / (elements["G"] + elements["C"] + elements
["T"] + elements["A"]))*100)

print("GC content of", geneID, "is",  format(a, '.2f'), "%")

# modifies the sequenction

print (geneSequence.replace('A','T'))
```

3.3. Частично изпълнено чрез възможността за задаване на { "Content-Type" : "text/x-fasta"}

Схематично алгоритъма за работа на приложението може да се представи по следния начин:




Пълният изходен код на приложението е представен по-долу:

```
import requests, json
from collections import Counter


geneID = "ENSG00000157764"
```

```
# provides the sequence in text format

server = "https://rest.ensembl.org"
ext = "/sequence/id/" + geneID + "?"

r = requests.get(server+ext, headers={ "Content-Type" : "text/plain"})

if not r.ok:
  r.raise_for_status()
  sys.exit()

print(r.text)

geneSequence = r.text

# provides ids of transcripts and exons and processes the outcome

def fetch_endpoint(server, request, content_type):

    r = requests.get(server+request, headers={ "Accept" : content_type})

    if not r.ok:
        r.raise_for_status()
        sys.exit()

    if content_type == 'application/json':
        return r.json()
    else:
        return r.text

def fetch_endpoint_POST(server, request, data, content_type):

    r = requests.post(server+request,
                      headers={ "Accept" : content_type},
                      data=data )

    if not r.ok:
        r.raise_for_status()
        sys.exit()

    if content_type == 'application/json':
        return r.json()
    else:
        return r.text

server = "http://rest.ensembl.org/"
ext_gene = "lookup/id/" + geneID + "?expand=1"
```

```
con_json = "application/json"
ext_sequence = "/sequence/id/"
con_seq = "text/x-fasta"
get_gene = fetch_endpoint(server, ext_gene, con_json)


# for each transcript get a list of the exon ids
for transcript in get_gene['Transcript']:
    print (">", transcript['id'])
    exons = []
    for exon in transcript['Exon']:
        exons.append(exon['id'])

    # create a json dump of the exon ids
    exon_json = json.dumps({ "ids" : exons })
    sequences = fetch_endpoint_POST(server, ext_sequence, exon_json, con_seq)

    print(sequences)

# calculates GC content

elements = Counter(geneSequence)

a = (((elements["G"] + elements["C"]) / (elements["G"] + elements["C"] + elements
["T"] + elements["A"]))*100)

print("GC content of", geneID, "is",  format(a, '.2f'), "%")

# modifies the sequence

print (geneSequence.replace('A','T'))
```

Примерния изход на при зададено ensamble id на ген изглежда по следния начин:

CTTCCCCCAATCCCCTCAGGCTCGGCTGCGCCTCCGCCTCCCTTCCCCCTCCCCGCCCGACAG

CGGCCGCTCGGGCCCCGGCTCTCGGTTATAAGATGGCGGCGCTGAGCGGTGGCG…….

……

> ENST00000496384

>ENSE00003828230.1 chromosome:GRCh38:7:140924566:140924810:-1

CCCCGCTCCTCCGCCTCCGCCTCCGCCTCCGCCTCCCCCAGCTCTCCGCCTCCCCCC…….

>ENSE00003603715.1 chromosome:GRCh38:7:140850111:140850212:-1

GTGTGGAATATCAAACAAATGATTAAGTTGACCGCGGGCCGATGATTATCATAGT…….

……..

> ENST00000644969

>ENSE00001862791.3 chromosome:GRCh38:7:140924566:140924929:-1

CTTCCCCCAATCCCCTCAGGCTCGGCTGCGCCCGGGGCCGCGGGCCGGTACCTGAG…….

>ENSE00003603715.1 chromosome:GRCh38:7:140850111:140850212:-1

GTGTGGAATATCAAACAAATGATTAAGTTGACACAGGAACATATAGAGGCCCTATT…….

……

GC content of ENSG00000157764 is 38.00 %

CTTCCCCCTTTCCCCTCTGGCTCGGCTGCGCCCGGGGCCGCGGGCCGGTTCCTGTGGTGGCCCTG

GCGCCCTCCGCCCGCGGCGCCGCCCGGGCCGCTCCTCCCCGCGCCCCCCGCGCCCCCC…….

……

## 4. Заключение

Python, използван в комбинация с REST заявки към публично предоставени крайни точки за достъп на различни ресурси (като ensemble.org) е мощен инструмент за анализ и обработка на биологична информация.

## 5. Използвана литература

https://rest.ensembl.org/

https://www.ebi.ac.uk/training-beta/online/courses/ensembl-rest-api/

https://notebooks.gesis.org/binder/jupyter/user/ensembl-rest-api-jupyter-course-ljcaoh1d/tree/Python3

helpdesk@ensembl.org