

# Universidade Federal do Rio de Janeiro

## Centro de Ciências da Saúde

### Instituto de Biofísica Carlos Chagas Filho

**Disciplina:** CFB017 - Programação para Biociências

**Professor:** Dr. Vitor Lima Coelho

## Exercícios 2

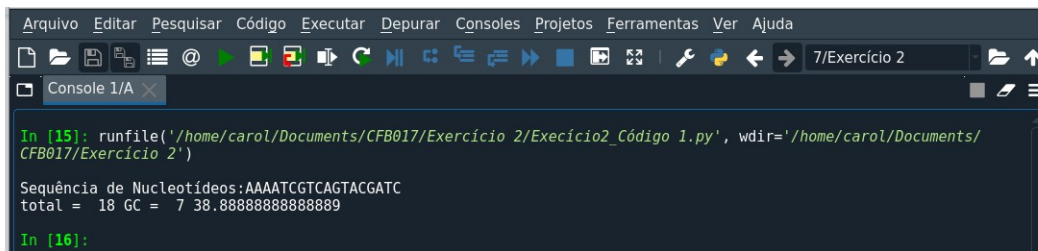
1 - Escreva um programa em Python que recebe uma sequência de DNA do usuário e calcula o conteúdo GC da sequência.

- Não utilize biopython para isto.

Código:

```
# guarda a sequência numa variável em formato de string
seq = str(input('Sequência de Nucleotídeos:'))
# contador de conteúdo GC
count = 0
for i in seq:
    # função upper para facilitar
    # 'or' booleano. Reescreve a variavel(x==y or x==z) ou utiliza elif
    if str(i).upper() == "G" or str(i).upper() == "C" :
        # contador recebe valor
        count = count+1
# conteúdo GC é em porcentagem
print('total = ', len(seq), 'GC = ', count, ((count/len(seq))*100))
```

Output:



```
Arquivo Editar Pesquisar Código Executar Depurar Consoles Projetos Ferramentas Ver Ajuda
7/Exercício 2
Console 1/A
In [15]: runfile('/home/carol/Documents/CFB017/Exercício 2/Exercício2_Código 1.py', wdir='/home/carol/Documents/CFB017/Exercício 2')
Sequência de Nucleotídeos:AAAATCGTCAGTACGATC
total = 18 GC = 7 38.88888888888889
In [16]:
```



2 - Escreva um programa Python que peça ao usuário uma sequência de aminoácido e imprima o percentual de cada aminoácido.

- Não utilize biopython para isto.

Código:

```
# recebendo a sequencia de AA do usuário
seq = str(input('Sequência de AA:')).upper()
# calculo do total de AA da sequência
total = len(seq)
# dicionário com contador dos AA existentes
aa_count = { 'A': 0, 'C': 0, 'D': 0, 'E': 0, 'F': 0, 'G': 0, 'H': 0, 'I': 0, 'K': 0, 'L': 0, 'M': 0, 'N': 0, 'P': 0, 'Q': 0, 'R': 0, 'S': 0, 'T': 0, 'V': 0, 'W': 0, 'Y': 0}
# contagem pra cada aminoácido
for i in seq:
# a cada item da sequencia que estiver no dicionário, vai acrescentar 1 à contagem
    aa_count[i] += 1
# calculo da porcentagem
for key, value in aa_count.items():
    if value != 0:
        percentage = float((value/total)*100)
# imprimir AAs e sua porcentagem
    print(key, percentage)
```

Output:

```
Applications ter, out 6 6:19 PM
Spyder (Python 3.7)
Arquivo Editar Pesquisar Código Executar Depurar Consoles Projetos Ferramentas Ver Ajuda
/home/carol/Documents/CFB017/Exercicio 2
Console 1/A
aa_count[i] += 1
KeyError: 'm'

In [10]: runfile('/home/carol/Documents/CFB017/Exercicio 2/Exercicio2_Código 2.py', wdir='/home/carol/Documents/CFB017/Exercicio 2')
Sequência de AA:MGHTAAAYWS
A 30.0
G 10.0
H 10.0
M 10.0
S 10.0
T 10.0
W 10.0
Y 10.0

In [11]: runfile('/home/carol/Documents/CFB017/Exercicio 2/Exercicio2_Código 2.py', wdir='/home/carol/Documents/CFB017/Exercicio 2')
Sequência de AA:mghtaayws
A 30.0
G 10.0
H 10.0
M 10.0
S 10.0
T 10.0
W 10.0
Y 10.0

In [12]: |
Editor Console IPython
conda: base (Python 3.7.6) Line 17, Col 30 ISO-8859-9 LF RW Mem 51%
```



3 - Escreva um programa Python que peça ao usuário duas sequências de DNA e imprima o complemento reverso de sua concatenação

- Não utilize biopython para isto.

Código:

```
seq_a = str(input('Sequência 1:'))
seq_b = str(input('Sequência 2:'))
# concatenação de strings
seq = seq_a + seq_b
# invertendo a ordem da sequência
rev_seq = seq[::-1]
complemento_reverso = ''
# essa parte pode ser feita com dicionário também
for i in rev_seq:
    if i.upper() == 'A':
# inserção de um novo caractere à string complemento reverso
        complemento_reverso += 'T'
    if i.upper() == 'T':
        complemento_reverso += 'A'
    if i.upper() == 'G':
        complemento_reverso += 'C'
    if i.upper() == 'C':
        complemento_reverso += 'G'
print(complemento_reverso)
```

Output:

```
Arquivo Editar Pesquisar Código Executar Depurar Consoles Projetos Ferramentas Ver Ajuda
7/Exercício 2
Console 1/A
In [17]: runfile('/home/carol/Documents/CFB017/Exercício 2/sem título1.py', wdir='/home/carol/Documents/CFB017/Exercício 2')
Sequência 1:AAACGT
Sequência 2:CTGAAA
TTTCAGACGTTT
In [18]:
```



4 - Escreva um programa em Python que recebe do usuário uma sequência de DNA que contém dois exons e um íntron; leia do usuário as coordenadas do primeiro exon (no formato **start1;end1**) e as coordenadas do segundo exon (**start2;end2**). Em seguida, imprima apenas a região exônica, isto é, as sub-sequências concatenadas do primeiro exon com o segundo exon.

- Não utilize biopython para isto.
- Sequência digitada pelo usuário:

```
ATCGATCGATCGATCGACTGACTAGTCATAGCTATGCATGTAGCTACTCGATCGATCG
ATCGATCGATCGATCGATCGATCGATCATGCTATCATCGATCGATATCGATGCATCGA
CTACTAT
```

- Coordenadas do exon 1 digitada pelo usuário:

```
1:20
```


- Coordenadas do exon 2 digitada pelo usuário:

```
62:123
```

Código:

```
# recebendo sequência de pré-mRNA
RNA = str(input('Digite sua sequência de RNA:'))
# recebendo coordenadas exônicas
## o split separa o input e gera uma lista com os itens que foram
separados
coordenada_a = str(input('coordenada 1:')).split(';')
coordenada_b = str(input('coordenada 2:')).split(';')
# fatiando a string sequência
## elementos da lista gerada pelo .split são transformadas em números
## em python a contagem se inicia do zero, logo, deve-se subtrair 1
das coordenadas
exon1 = RNA[int(coordenada_a[0])-1:int(coordenada_a[-1])-1]
exon2 = RNA[int(coordenada_b[0])-1:int(coordenada_b[-1])-1]
# concatenação dos exons
mRNA = exon1+exon2
print(mRNA)
```

Output:



```
In [12]: runfile('/home/carol/Documents/CFB017/Exercício 2/Execício2_Código 4.py', wdir='/home/carol/Documents/CFB017/Exercício 2')

Digite sua sequência de
RNA:ATCGATCGATCGATCGACTGACTAGTCATAGCTATGCATGTAGTACTCGATCGATCGATCGATCGATCGATCGATCGATCATGCTATCATCGATCGATATCGATGCA
TCGACTACTAT

coordenada 1:1;20

coordenada 2:62;123
ATCGATCGATCGATCGACTGACTGATCGATCGATCGATCGATCATGCTATCATCGATCGATATCGATGCA TCGACTACTAT
```