1 Interpret Pythonu

Interpret Pythonu sa typicky spustí volaním programu `python`, ale je lepšie využiť nejaké vývojové prostredie. Takýchto vývojových prostredí je voľne k dispozícii niekoľko. IDLE je relatívne jednoduché prostredie, ale funguje totožne pod UNIX-like systémami i Windows. Bohatšie prostredia sú `Eclipse` a `Eric4`. Na cvičení budeme používať IDLE. Na začiatku práce je vhodné nastaviť aktuálny adresár

```
import os
os.chdir('~/BioInf')
```

Tiež sa môže hodiť úprava cesty, podľa ktorej Python hľadá súbory pre **`import`**

```
import sys
sys.path        #vypise zoznam prehladavanych adresarov
```

Do cesty je možné pridať (na začiatok) adresár, kde budú naše moduly

```
sys.path.insert(0,'~/BioInf/Modules')
```

2 Jednoduché manipulácie s DNA sekvenciami

Biologické sekvencie je najjednoduchšie reprezentovať ako textové reťazce. Naprogramujte nasledujúce funkcie:

1. `DnaComplement(dna)` vráti reťazec DNA komplementárny k reťazcu `dna`.

   ```
   >>> DnaComplement('ACGTAATCGT')
   'CATGCCGATG'
   ```

2. `Reverse(seq)` vráti sekvenciu `seq` zapísanú odzadu.

   ```
   >>> Reverse('ACGTAATCGT')
   'TGCTAATGCA'
   ```

3. `DnaReverseComplement(dna)` vráti reťazec DNA komplementárny k obrátenému reťazcu `dna`.

   ```
   >>> DnaReverseComplement('ACGTAATCGT')
   'GTAGCCGTAC'
   ```

4. `MotifPos(mot,seq)` vráti zoznam pozícií, kde sa v sekvencii `seq` vyskytuje podreťazec `mot` (nazývaný motív, angl. motif).

   ```
   >>> MotifPos('CGT','ACGTAATCGT')
   [1, 7]
   ```

V Pyhtone je vytváranie reťazca postupným pridávaním po jednom znaku veľmi neefektívne. Pri pridaní každého znaku sa vytvorí nový reťazec. Reťazec je možné previesť na zoznam jednoznakových reťazcov a naspäť.

```
>>> s='aBc s'
>>> print s
aBc s
>>> l=list(s)
>>> print l
['a', 'B', 'c', ' ', 's']
>>> ss=''.join(l)
>>> print ss
aBc s
```

Okrem prevodu `string` → `zoznam` je možné urobiť cyklus priamo po znakoch reťazca:

```
for c in theString:
    processChar(c)
```

Tzv. 'list comprehension' tiež umožňuje efektívne prejsť všetky znaky reťazca (alebo prvky zoznamu):

```
results = [ processChar(c) for c in theString ]
```

Rovnako efektívne funguje tzv. mapovanie

```
results = map(processChar,theString)
```

Ak potrebujeme zistiť množinu znakov, ktoré sa v reťazci vyskytujú, tak to efektívne urobí prevod na množinu

```
>>> s=set('acgtctaatGgaCtdAcTTtgGAagtCcCCCcTGActActcta'.upper())
>>> print s
set(['A', 'C', 'T', 'G', 'D'])
>>> dnachars = set(['A','C','G','T'])
>>> if s-dnachars:
   print 'Nie je to DNA'
else:
   print 'Je to DNA'

Nie je to DNA
```

Rýchle funguje aj nahradzovanie v reťazcoch

```
>>> s='dAcTTtgGAagtCcCCCcTGAc'
>>> print s,'\n',s.replace('Cc','*')
dAcTTtgGAagtCcCCCcTGAc
dAcTTtgGAagt*CC*TGAc
```

Pre transformácie, kde sa jeden znak nahradzuje jedným znakom je výhodná funkcia `translate`, ktorá však vyžaduje tabuľku – vektor 256 znakov – ktorá udáva na *i*-tej pozícii znak, na ktorý sa má previesť znak s ordinálnou hodnotou *i*. Takú tabuľku vie pripraviť funkcia `maketrans`. Napríklad nahradenie všetkých znakov `'G'` na `'a'`, `'I'` na `'a'` a `'e'` na `'a'`.

```
>>> import string
>>> s='GbrIke,dGbrIke'
>>> tr = string.maketrans('GIe','aaa')
>>> string.translate(s,tr)
'abraka,dabraka'
```

Otočenie reťazca sa dá dosiahnuť efektívne rozšíreným operátorom indexovania

```
>>> s='abraka'
>>> s[::-1]
'akarba'
```

3 Kódovanie proteínov v DNA

Štandardný kód amínokyselín v DNA je nasledujúca tabuľka:

TTT→F	TTC→F	TTA→L	TTG→L	TCT→S
TCC→S	TCA→S	TCG→S	TAT→Y	TAC→Y
TGT→C	TGC→C	TGG→W	CTT→L	CTC→L
CTA→L	CTG→L	CCT→P	CCC→P	CCA→P
CCG→P	CAT→H	CAC→H	CAA→Q	CAG→Q
CGT→R	CGC→R	CGA→R	CGG→R	ATT→I
ATC→I	ATA→I	ATG→M	ACT→T	ACC→T
ACA→T	ACG→T	AAT→N	AAC→N	AAA→K
AAG→K	AGT→S	AGC→S	AGA→R	AGG→R
GTT→V	GTC→V	GTA→V	GTG→V	GCT→A
GCC→A	GCA→A	GCG→A	GAT→D	GAC→D
GAA→E	GAG→E	GGT→G	GGC→G	GGA→G
GGG→G				

K nej je treba ešte poznať množinu štartovacích kodónov {TAA, TAG, TGA}, ktorými môže gén začínať, a množinu stop-kodónov {TTG, CTG, ATG}, ktorými gén končí.

Ale existujú i ďalšie kódy. Napríklad v mitochondriálnej DNA stavovcov (čes. obratlovce) funguje tabuľka:

TTT→F	TTC→F	TTA→L	TTG→L	TCT→S
TCC→S	TCA→S	TCG→S	TAT→Y	TAC→Y
TGT→C	TGC→C	TGA→W	TGG→W	CTT→L
CTC→L	CTA→L	CTG→L	CCT→P	CCC→P
CCA→P	CCG→P	CAT→H	CAC→H	CAA→Q
CAG→Q	CGT→R	CGC→R	CGA→R	CGG→R
ATT→I	ATC→I	ATA→M	ATG→M	ACT→T
ACC→T	ACA→T	ACG→T	AAT→N	AAC→N
AAA→K	AAG→K	AGT→S	AGC→S	GTT→V
GTC→V	GTA→V	GTG→V	GCT→A	GCC→A
GCA→A	GCG→A	GAT→D	GAC→D	GAA→E
GAG→E	GGT→G	GGC→G	GGA→G	GGG→G

s množinou štartovacích kodónov {TAA, TAG, AGA, AGG} a množinou stop-kodónov {ATT, ATC, ATA, ATG, GTG}.

Takýchto tabuliek sa v bioinformatike používa viac než 10. Jedna z vhodných reprezentácií takejto tabuľky v Pythone je slovník. Štandardná tabuľka sa dá napríklad reprezentovať ako trojica

```
(slovnik_kodonov,start_kodony,stop_kodony).
```

Štandardná kódovacia tabuľka sa potom vytvorí nasledovne:

```
std_table = ({
          'TTT': 'F', 'TTC': 'F', 'TTA': 'L', 'TTG': 'L', 'TCT': 'S',
          'TCC': 'S', 'TCA': 'S', 'TCG': 'S', 'TAT': 'Y', 'TAC': 'Y',
          'TGT': 'C', 'TGC': 'C', 'TGG': 'W', 'CTT': 'L', 'CTC': 'L',
          'CTA': 'L', 'CTG': 'L', 'CCT': 'P', 'CCC': 'P', 'CCA': 'P',
          'CCG': 'P', 'CAT': 'H', 'CAC': 'H', 'CAA': 'Q', 'CAG': 'Q',
          'CGT': 'R', 'CGC': 'R', 'CGA': 'R', 'CGG': 'R', 'ATT': 'I',
          'ATC': 'I', 'ATA': 'I', 'ATG': 'M', 'ACT': 'T', 'ACC': 'T',
          'ACA': 'T', 'ACG': 'T', 'AAT': 'N', 'AAC': 'N', 'AAA': 'K',
          'AAG': 'K', 'AGT': 'S', 'AGC': 'S', 'AGA': 'R', 'AGG': 'R',
          'GTT': 'V', 'GTC': 'V', 'GTA': 'V', 'GTG': 'V', 'GCT': 'A',
          'GCC': 'A', 'GCA': 'A', 'GCG': 'A', 'GAT': 'D', 'GAC': 'D',
          'GAA': 'E', 'GAG': 'E', 'GGT': 'G', 'GGC': 'G', 'GGA': 'G',
          'GGG': 'G' },
          ('TAA', 'TAG', 'TGA'),
          ('TTG', 'CTG', 'ATG')
        )
```

Úlohy:

1. Napíšte funkciu `PrintCodonTab(table=std_table)`, ktorá vypíše tabuľku kódov amínokyselín pomocou trojíc báz v DNA a zoznamy štartovacích a stop-kodónov. Keď je parameter s tabuľkou vynechaný, tak má použiť štandardnú tabuľku. Tabuľku vypíšte prehľadne, príkaz

```
print table
```

nestačí!

2. Napíšte funkciu `Cod2Aa(codon,table=std_table)`, ktorá pre daný kodón `codon` vráti buď jednopísmenkový kód amínokyseliny alebo reťazec `'Stop'`, ak sa jedná o stop-kodón. Keď je parameter s tabuľkou vynechaný, tak má použiť štandardnú tabuľku.

3. Napíšte funkciu `Aa2Cod(amino,table=std_table)`, ktorá pre daný jednopísmenkový kód amínokyseliny `amino` vráti zoznam kodónov, ktoré ju kódujú a pre `amino='Stop'` vráti zoznam stop-kodónov. Keď je parameter s tabuľkou vynechaný, tak má použiť štandardnú tabuľku.

4. Napíšte funkciu `Dna2Cod(dna,pos=0,table=std_table)`, ktorá pre daný reťazec DNA `dna` vráti reťazec proteínu, ktorý začína na pozícii `pos` a končí buď nejakým stop-kodónom alebo koncom reťazca DNA. Keď je parameter s tabuľkou vynechaný, tak má použiť štandardnú tabuľku.

Poznámky: Na manipuláciu so slovníkmi sa hodia nasledujúce funkcie

- `slovnik.has_key(kluc)` vráti `True`, keď slovník obsahuje dvojicu s kľúčom `kluc`,
- `slovnik.keys()` vráti zoznam všetkých kľúčov v slovníku `slovnik`,
- `slovnik[kluc]` vráti hodnotu uloženú s kľúčom `kluc`.

4 Načítanie bioinformatických sekvencií

Formát súborov FASTA je veľmi jednoduchý. Jedná sa o textový súbor. Zápis sekvencie začína vždy hlavičkou. Hlavička je jednoriadkový popis začínajúci znakom > nasledovaným identifikátorom sekvencie (až do medzery alebo konca riadku). Za identifikátorom môže nasledovať ľubovoľný popis až do konca riadku. Po hlavičke nasleduje ľubovoľný počet riadkov obsahujúcich sekvenciu rozsekanú na kusy maximálne 80 znakov (často sa maximálna dĺžka riadku nedodržuje). Sekvencia končí buď s koncom súboru alebo hlavičkou ďalšej sekvencie.

Príklad súboru vo FASTA-formáte:

```
>AtMg00665 nad5b nad5.2
GATATGATGATTGGTTTAGGTA
>AtMg00513 nad5a nad5.1
ATGTATCTACTTATCGTATTTTTGCCCCTGCTCGGTAGTTCCGTAGCAGGTTTTTTCGGACGTTTTCTAGGATCA
GAAGGAAGCGCTATAATGACCACTACGTGCGTTTCATTCTCTTCGATCTTATCTTTGATTGCTTTTTATGAAGTC
GCACCGGGAGCTAGTGCTTGCTATCTAAGAATTGCTCCATGGATCTCATCGGAAATGTTTGATGCTTCTTGGGGC
TTCTTGTTCGATAGCCCGACCGTAGTGATGTTAATTGTGGTTACATCCATAAGTAGCTTGGTCCATCTTTATTCC
ATTTCATATATGTCCGAGGATCCGCATAGCCCTCGATTTATGTGTTATTTATCCATTCTTACTTTTTTTATGCCA
...
```

Základ spracovania textového súboru v Pythone môže byť takto jednoduchý:

```
data = open('Arabidopsis.fasta').read()
```

Celý textový súbor je načítaný do jediného reťazca. S takýmto reťazcom sa však dobre nepracuje. Podobne sa dá prečítať naraz celý binárny súbor.

```
data = open('Subor.bin','rb').read()
```

Bezpečnejšie je však definovať inštanciu objektu pre súbor a ten po skončení práce zatvoriť

```
fileObj = open('Arabidopsis.fasta')
try:
    text = fileObj.read()
finally:
    fileObj.close()
```

Načítanie textového súboru sa robí obvykle pomocou metódy `readlines()`, ktorá vráti zoznam reťazcov obsahujúcich jednotlivé riadky súboru. Pozor, každý takto prečítaný riadok končí symbolom konca riadku `'\n'`. Ak súbor vo formáte FASTA obsahuje jedinú sekvenciu, tak sa táto sekvencia dá načítať nasledovne

```
fileObj = open('Arabidopsis.fasta')
try:
    lines = fileObj.readlines()
finally:
    fileObj.close()
dna= ''.join(lines[1:])
dna = dna.replace('\n','')
```

Na postupné spracovanie textového súboru po riadkoch sa hodí cyklus cez prvky (riadky) súbora

```
fileObj = open('Arabidopsis.fasta')
try:
    for line in fileObj:
        Process(line)
finally:
    fileObj.close()
...
```

Režim práce so súborom – druhý parameter funkcie `open` môže obsahovať napríklad

`'rU'` čítanie textového súboru s univerzálnym rozpoznávaním konca riadku – správne budú rozpoznané konce riadkov z DOSu/Windows (’
r
n’) ale i UNIXu (’
r’).

`'w'` zápis do textového súbora.
..

`'wb'` zápis do binárneho súbora.

Naprogramujte nasledujúce funkciu:

`ReadFasta(FName,verbose=1)` vráti slovník obsahujúci všetky sekvencie zo súboru s menom `FName`. Kľúčom v slovníku budú identifikátory sekvencií. Pre `verbose = 0` funkcia nič nepíše. Inak vypíše identifikátory načítaných sekvencií a ich dĺžky.