

# RNA FRABASE Adapter

---

BIOINFORMATYKA STRUKTURALNA

Paweł Kaleciński  
UNIwersytet Jagielloński

## Spis Treści:

Informacje wstępne.....	3
Opis Projektu.....	3
Możliwości użytkownika.....	4
Hierarchia klas.....	4
Opis Main.....	4
Opis Helper.....	4
Wymagania.....	7
Opis bibliotek.....	7
Wdrożenie.....	7
Obsługa błędów oraz Obsługa wejścia.....	8
Zrzuty Ekranu.....	9
Podsumowanie.....	10

## Informacje Wstępne:

Celem mojego projektu było stworzenie adaptera do **RNA FRABASE**. Jest to silnik z bazą danych służący do wyszukiwania trójwymiarowych fragmentów w strukturach 3D RNA przy pomocy sekwencji i/lub struktury drugorzędowej w notacji dot-bracket. Baza ta posiada dwie opcje wyszukiwania: wyszukiwanie proste oraz wyszukiwanie zaawansowane. W swoim projekcie zająłem się wykorzystaniem wyszukiwania prostego, jednak nie wykluczam w przyszłości rozszerzenia funkcjonalności o wyszukiwanie zaawansowane. Dokumentacja ta ma na celu opisać program, by ułatwić użytkowanie oraz służyć, jako pomoc dla osób, które będą go modyfikować, bądź rozwijać w przyszłości.

## Opis Projektu:

Projekt umożliwia użytkownikowi korzystanie z internetowej bazy **RNA FRABASE** z poziomu Pythona. Przy pomocy niniejszego programu użytkownik może uzyskać następujące informacje o sekwencji:

- PDB ID
- NDB ID
- Struktura drugorzędowa
- Sekwencja
- Łańcuch, jego początek oraz koniec
- Metoda
- Klasa
- Data depozycji w PDB
- Rozdzielczość
- Modele

## Możliwości Użytkownika:

Użytkownik ma następujące możliwości:

- Wpisanie sekwencji ręcznie
- Wczytanie sekwencji z pliku
- Wyszukiwanie Proste

## Hierarchia Klas:

W projekcie znajdują się dwie główne klasy:

- Main
- Helper

## Opis Klasy Main:

Zawiera rdzeń naszego programu. Tutaj znajdują się główny mechanizm połączenia z bazą danych

## Opis Klasy Helper:

Klasa Helper, jak może sugerować nazwa jest klasą pomoczną. Posiada ona funkcję sprawdzającą wejście do naszego programu oraz funkcję umożliwiającą parsowanie i pobieranie danych z dokumentu HTML.

Metody występujące w tej klasie to:

```
def check_characters(self, seq):  
    i = 0  
    chars = ('A', 'C', 'G', 'U', 'R', 'Y', 'M', 'K', 'W', 'S', 'B', 'D',  
            'H', 'V', 'N', '.', '(', ')', '[', ']', '{', '}', '<', '>', '?', '\n')  
    while i < len(seq):  
        if seq[i].upper() not in chars:  
            print("Invalid characters in sequence")  
            return True  
        i += 1
```

Metoda ta umożliwia sprawdzenie czy we wpisanej sekwencji nie występuje niepożądany znak. Jej parametrem jest **seq**, czyli sekwencja, która jest sprawdzana.

```
def check_numbers(self, seq):
    if re.findall('\d+', seq):
        print("Invalid characters in sequence")
        return True
```

Metoda sprawdzająca czy we wpisanej sekwencji nie pojawiły się liczby. Jeżeli tak zwraca True. Wykorzystuję ten sam parametr co poprzednia metoda.

```
def check_length(self, seq):
    chars = ('A', 'C', 'G', 'U', 'R', 'Y', 'M', 'K', 'W', 'S', 'B',
'D', 'H', 'V', 'N')
    braces = ('(', ')', '[', ']', '{', '}')
    if len(seq) < 11 and (seq.count(seq[0]) == len(seq)) and all(x ==
seq[0] for x in seq) and seq[0] == ".":
        print("Sequence too short")
        return True
    elif len(seq) < 4 and seq in chars:
        print("Sequence too short")
        return True
    elif len(seq) < 6 and (seq.count(seq[0]) == len(seq)) and all(x ==
seq[0] for x in seq) and seq[0] in braces:
        print("Sequence too short")
        return True
```

Metoda odpowiadająca za sprawdzenie długości wprowadzanej sekwencji. Jeżeli sekwencja składa się z samych kropek musi mieć ona co najmniej 11 znaków, jeżeli składa się z liter musi mieć co najmniej 4 znaki, a jeżeli składa się z nawiasów musi mieć minimum 6 znaków. Wykorzystuje ten sam parametr co poprzednie metody

```
def check_identity(self, seq, flag, list1):
    if seq in list1 and flag > 1 and not (seq == list1[len(list1)-1]):
        print("Incorrect pattern definition")
        return True
    else:
        return False
```

Metoda sprawdzająca czy aktualnie wpisywana sekwencja nie jest identyczna z sekwencją wpisaną wcześniej. Jako argumenty przyjmuję aktualną sekwencję(**seq**), flagę(**flag**), która jest zmienną pomocniczą, gdy wprowadzamy kilka sekwencji oraz listę(**list1**), w której sprawdzamy czy występuje dana sekwencja oraz czy nie jest ona na ostatnim miejscu, co oznaczałoby, że jest to aktualnie wpisywana sekwencja.

```

def main_algorithm(self, answer, soup, counter, sequence, names):
    if answer.lower() == "yes":
        root = Tk()
        root.fileName = filedialog.askopenfilename()
        root.destroy()
        file = open(root.fileName, 'a', encoding='utf-8')
        for row in (soup.find_all(attrs={"class": ["row_table1",
"row_table2"]})):
            for i in row.find_all("td"):
                if counter == 14:
                    counter = 0
                if counter % 13 == 0 and counter != 0:
                    sequence += " "
                    counter += 1
                if answer.lower() == "yes":
                    file.write(sequence + "\n")
                else:
                    print(sequence)
                    sequence = ""
                elif not i.text == "" or (names[counter] == "Å") or
sequence == " ":
                    sequence += names[counter] + " " + i.get_text('\n' +
names[counter] + " ").strip()
                    counter += 1
                if answer.lower() == "yes":
                    file.write(sequence + "\n")
                    sequence = ""
                else:
                    print(sequence)
                    sequence = ""
            if answer.lower == "yes":
                file.write("\n")
                file.close()

```

Najdłuższa funkcja, serce program. Przyjmuję ona 5 argumentów:

- answer – parametr odpowiadający za decyzję czy będziemy dokonywać zapisu do pliku czy też nie
- soup – parametr umożliwiający parsowanie dokumentu HTML
- counter – licznik, zmienna pomocnicza
- sequence – dana sekwencja, która zostanie wypisana, bądź zapisana do pliku
- names – krotka, umożliwiająca opisanie poszczególnych wierszy, wypisywanych przez program

Funkcja ta parsuje i wydobywa informacje z naszymi wynikami, umożliwia również zapis do pliku.

## Wymagania:

By korzystać z projektu wymagane są następujące elementy:

- Python w wersji 3.\*
- PhantomJS w wersji 2.1.1
- biblioteka Selenium
- biblioteka bs4
- biblioteka tkinter

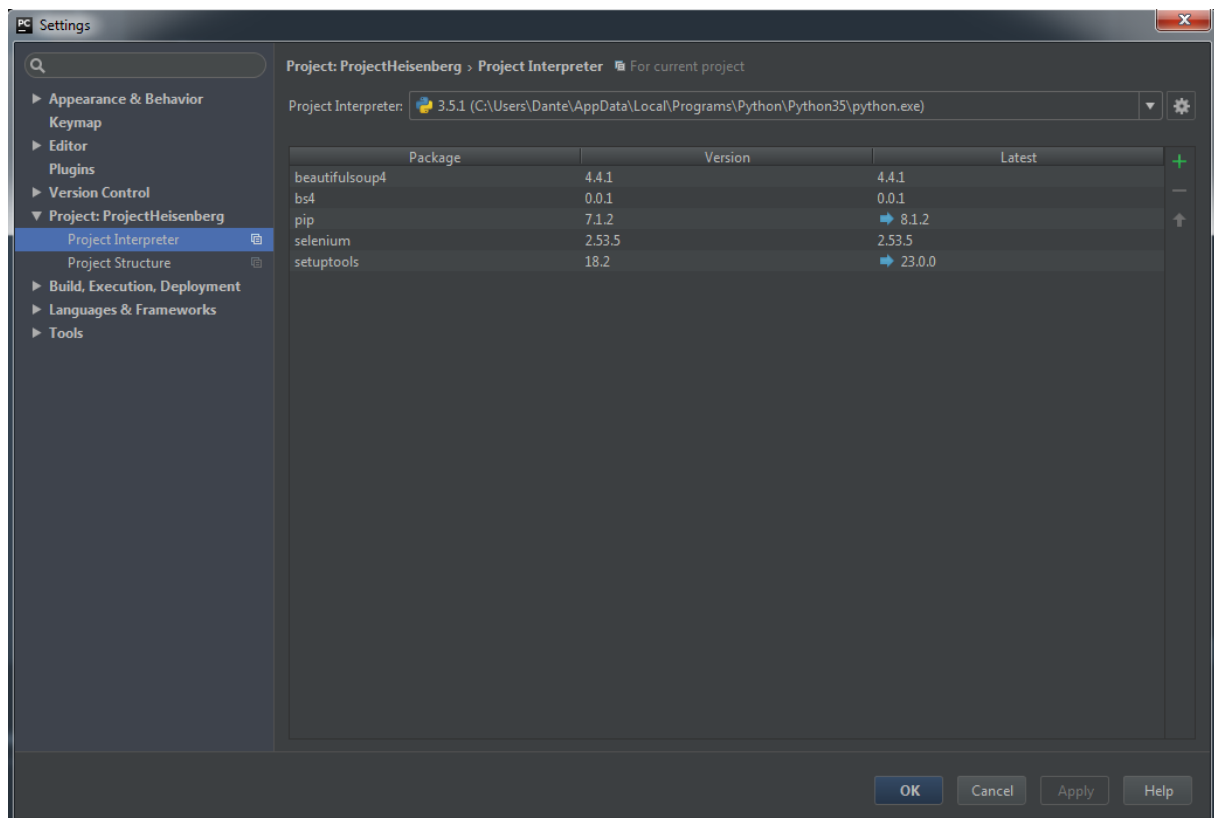
## Opis bibliotek:

- Selenium: biblioteka umożliwiająca dostanie się do bazy, oraz pobieranie z niej pożądaných informacji
- bs4: biblioteka, umożliwiająca Parsowanie dokumentów HTML i wydobywanie z nich potrzebnych nam informacji
- tkinter: biblioteka graficzna, wykorzystana w celu pojawienia się okienek dialogowych umożliwiających wskazanie lokalizacji dla pliku wejściowego oraz pliku wyjściowego

## Wdrożenie:

By móc korzystać z projektu musimy upewnić się, że posiadamy wymienione powyżej biblioteki. Jeżeli nie musimy je zainstalować, możemy zrobić to za pomocą narzędzia **pip**, następującą komendą **pip install [tu wpisz nazwę wymaganej biblioteki]** lub jeżeli posiadamy **Pycharma**, jego twórcy ułatwili nam pracę i umożliwili pobranie potrzebnych bibliotek z poziomu **Pycharma**. Robimy to w następujący sposób:

Klikamy **File -> Settings**, ukaże nam się nowe okienko, tam wybieramy po lewej stronie **Project: [nazwa projektu]** i naszym oczom ukazują się następujący widok:



Naciskamy na zielony plus, wpisujemy nazwę pożądanego pakietu i instalujemy.

Gdy już mamy potrzebne biblioteki należy w projekcie ustawić ścieżkę do PhantomJS, robimy to w następującym miejscu w kodzie:

```
browser = webdriver.PhantomJS("Tu wpisz swoją ścieżkę")
```

Gdy już to zrobimy program jest gotowy do użycia.



## Obsługa błędów oraz Obsługa wejścia:

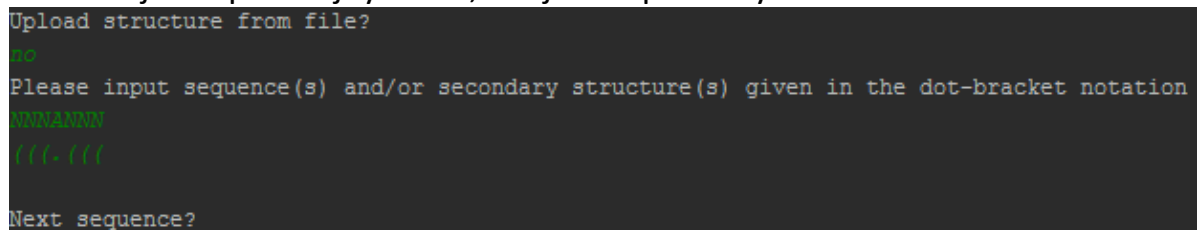
Mylić się jest rzeczą ludzką, dlatego też program jest wyposażony w odpowiedni sposób sprawdzania poprawności wpisanych danych. Mamy wymienione wyżej funkcję, a jeżeli one zawiodą to nic się nie stało. Program i tak nie pozwoli nam wykonać niedozwolonych operacji, jeżeli coś przejdzie przez funkcję po prostu jego akcja zostanie zakończona, a my będziemy poinformowani odpowiednim komunikatem oraz zostaniemy zmuszeni do ponownego odpalenia projektu. Jeżeli pomylimy się przy wpisywaniu drugiej sekwencji, niestety będziemy zmuszeni wpisać ponownie cały nasz input, łącznie z poprzednimi sekwencjami. Jeżeli chodzi o wczytywanie z pliku to format jest następujący:

```
#tRNA(Phe), yeast (Saccharomyces Scerevisiae)
>strand1
gCGGAUUUAgCUCAGuuGGGAGAGCgCCAGAcUgAAgAucUGGAGgUCcUGUGuuCGaUCCACAGAAUUCGCACC
```

Pierwsza linia nie jest brana pod uwagę, druga oznacza numer sekwencji, a trzecia to już sama sekwencja. Format dla dwóch sekwencji wyglądałby następująco:

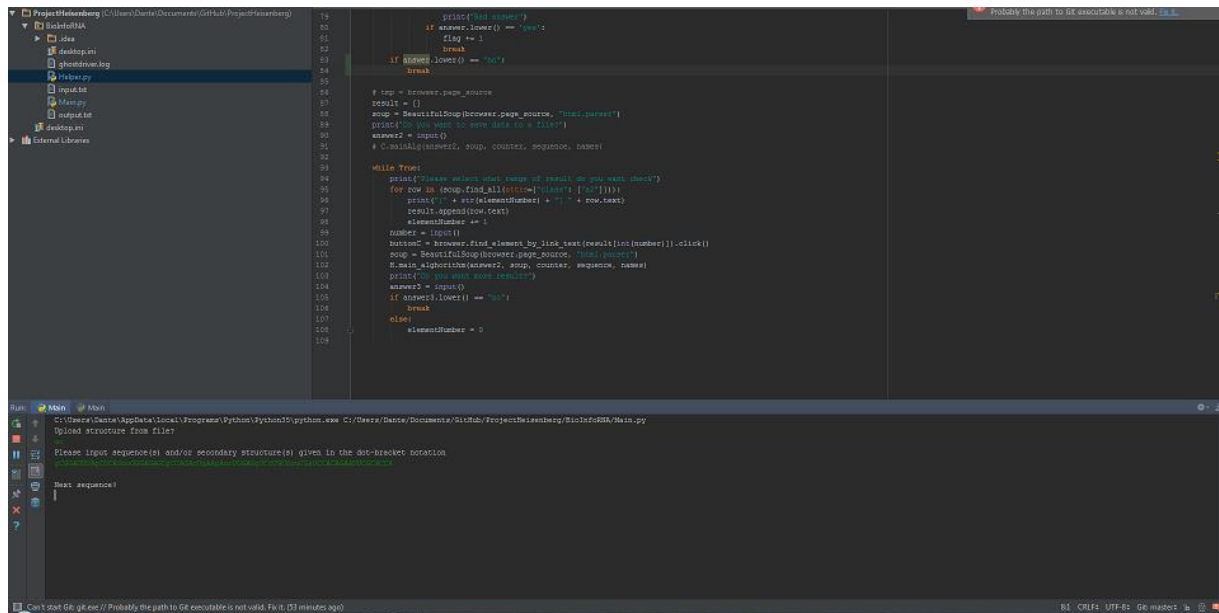
```
#Bulge
>strand1
NNNANNN
(((.(((
>strand2
NNNNNN
))))))
```

Wpisując ręcznie musimy jedynie podawać samą sekwencję, reszta uzupełniana jest automatycznie(>strand). Użytkownik musi pamiętać, by po wpisaniu sekwencji dać podwójny enter, tak jak na poniższym obrazku:



```
Upload structure from file?
no
Please input sequence(s) and/or secondary structure(s) given in the dot-bracket notation
NNNANNN
(((.(((
Next sequence?
```

## Zrzuty Ekranu:



```
No. 47
PDB id 3DEG
NDB id
Sequence GCGGAUUUAgCUCAGuuGGGAGAGCGCCAGAcUgAAgAucUGGAGgUCCUGUGuuCGaUCCACAGAAUUCGCACCA
Secondary Structure ((((((...(((.....[...]))).(((.....)))).....((((...[...]))))))))....
Chain A
Start 1
End 76
Method Electron Microscopy
Class RIBOSOME
PDB deposition 2008-06-10
Å
Models 1

No. 48
PDB id 3E1A
NDB id
Sequence gCGGAUUUAgCUCAGuuGGGAGAGCGCCAGAcUgAAgAucUGGAGgUCCUGUGuuCGaUCCACAGAAUUCGCACCA
Secondary Structure ..(((((((...(((.....[...]))).(((.....[...])))).....((((...[...]))))))))....
Chain A
Start 1
End 76
Method Electron Microscopy
Class RIBOSOME
PDB deposition 2008-08-03
Å 9
Models 1
```

```
Upload structure from file?
no
Please input sequence(s) and/or secondary structure(s) given in the dot-bracket notation
gCGGAUUUAgCUCAGuuGGGAGAGCGCCAGAcUgAAgAucUGGAGgUCCUGUGuuCGaUCCACAGAAUUCGCACCA
Next sequence?
no
Do you want to save data to a file?
no
Please select what range of result do you want check
[0] 1-50
[1] 51-59
|
```

**Podsumowanie:**

Projekt posiada spore możliwości modyfikacji, na pewno w przyszłości będę chciał wprowadzić możliwość korzystania z wyszukiwania zaawansowanego, a także wprowadzić jakąś przyjemną oprawę graficzną. Wiem, że mogą pojawiać się błędy, dlatego też będę wdzięczny za wszelkie komentarze i uwagi. Można je kierować na mój adres meil:

[pawelkalecinski94@gmail.com](mailto:pawelkalecinski94@gmail.com) .

Chętnie będę widział również wszelkie poprawki i modyfikacje na GitHubie, ponieważ jest to projekt OpenSource. Mam nadzieję, że korzystanie z mojego programu będzie owocne, a ta dokumentacja ułatwi jego obsługę