

Projekt\_peptydy

Wygenerowano przez Doxygen 1.8.13

## Spis treści

<b>1</b>	<b>Indeks przestrzeni nazw</b>	<b>2</b>
1.1	Pakiety . . . . .	2
<b>2</b>	<b>Indeks klas</b>	<b>2</b>
2.1	Lista klas . . . . .	2
<b>3</b>	<b>Indeks plików</b>	<b>2</b>
3.1	Lista plików . . . . .	2
<b>4</b>	<b>Dokumentacja przestrzeni nazw</b>	<b>2</b>
4.1	Dokumentacja przestrzeni nazw program . . . . .	2
4.1.1	Dokumentacja funkcji . . . . .	3
4.1.2	Dokumentacja zmiennych . . . . .	5
<b>5</b>	<b>Dokumentacja klas</b>	<b>7</b>
5.1	Dokumentacja klasy program.atom . . . . .	7
5.1.1	Opis szczegółowy . . . . .	7
5.1.2	Dokumentacja konstruktora i destruktor . . . . .	8
5.1.3	Dokumentacja funkcji składowych . . . . .	8
5.1.4	Dokumentacja atrybutów składowych . . . . .	9
5.2	Dokumentacja klasy program.monomer . . . . .	10
5.2.1	Opis szczegółowy . . . . .	11
5.2.2	Dokumentacja konstruktora i destruktor . . . . .	11
5.2.3	Dokumentacja funkcji składowych . . . . .	12
5.2.4	Dokumentacja atrybutów składowych . . . . .	14
5.3	Dokumentacja klasy program.peptide . . . . .	14
5.3.1	Opis szczegółowy . . . . .	14
5.3.2	Dokumentacja funkcji składowych . . . . .	15
5.3.3	Dokumentacja atrybutów składowych . . . . .	16
5.4	Dokumentacja klasy program.vector . . . . .	16
5.4.1	Opis szczegółowy . . . . .	17
5.4.2	Dokumentacja konstruktora i destruktor . . . . .	17
5.4.3	Dokumentacja funkcji składowych . . . . .	18
5.4.4	Dokumentacja atrybutów składowych . . . . .	19

<b>6 Dokumentacja plików</b>	<b>19</b>
6.1 Dokumentacja pliku program.py . . . . .	19

## 1 Indeks przestrzeni nazw

### 1.1 Pakiety

Oto lista pakietów wraz z krótkim opisem (o ile jest dostępny):

<b>program</b>	<b>2</b>
----------------	----------

## 2 Indeks klas

### 2.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

<b>program.atom</b> <b>Atomy</b>	<b>7</b>
<b>program.monomer</b> <b>Monomery</b>	<b>10</b>
<b>program.peptide</b> <b>Peptydy</b>	<b>14</b>
<b>program.vector</b> <b>Wektory w układzie kartezjańskim</b>	<b>16</b>

## 3 Indeks plików

### 3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

<b>program.py</b>	<b>19</b>
-------------------	-----------

## 4 Dokumentacja przestrzeni nazw

### 4.1 Dokumentacja przestrzeni nazw program

#### Komponenty

- class **atom**  
*Atomy.*
- class **monomer**  
*Monomery.*
- class **peptide**  
*Peptydy.*
- class **vector**  
*Wektory w układzie kartezjańskim.*

## Funkcje

- def **transpose** (m)  
*Transpozycja macierzy.*
- def **dihedral\_angle** (b1, b2, b3)  
*Wyznaczanie kąta torsyjnego.*
- def **upload\_data** ( table)  
*Szuka atomów i wiązań istotnych dla tworzenia wiązania.*
- def **make\_output** (peptyd, i, j, n)  
*Tworzy plik w formacie pdb.*

## Zmienne

- **file** = open('Components-pub.cif')  
*Tworzy monomery białkowe.*
- int **i** = 0
- list **table** = []
- int **number\_of\_lines** = 0
- dictionary **monomers\_list** = {}
- **words** = re.split("\s+", line.strip())
- def **monomer\_i** = **upload\_data**( table)
- **data** = open(sys.argv[1])  
*Tworzy peptyd o zadanych kątach omega, fi, psi.*
- string **words2** = ""
- list **table2** = []
- **aa** = **peptide**()
- **f** = open('output.pdb', 'w')  
*Drukuje stworzony peptyd do pliku output.pdb.*
- int **n** = 0

## 4.1.1 Dokumentacja funkcji

## 4.1.1.1 dihedral\_angle()

```
def program.dihedral_angle (
    b1,
    b2,
    b3 )
```

Wyznaczanie kąta torsyjnego.

## Parametry

<i>b1, b2, b3</i>	trzy wektory pomiędzy, którymi liczony jest kąt torsyjny
-------------------	--

## Zwraca

wartość kąta torsyjnego

**Autor**

Bernadeta Nowosielska

**4.1.1.2 make\_output()**

```
def program.make_output (
    peptyd,
    i,
    j,
    n )
```

Tworzy plik w formacie pdb.

**Parametry**

<i>peptyd</i>	obiekt klasy peptyd - gotowy produkt łączenia aminokwasow
<i>i</i>	ilość aminokwasow w peptydzie
<i>j</i>	ilość atomow w kolejnych aminokwasach
<i>n</i>	numery porządkowe kolejnych aminokwasow

**Autor**

Barbara Gruza

**4.1.1.3 transpose()**

```
def program.transpose (
    m )
```

Transpozycja macierzy.

**Parametry**

<i>m</i>	maciwerz, która ma zostać zmieniona
----------	-------------------------------------

**Zwraca**

macierz transponowana

**Autor**

Bernadeta Nowosielska

## 4.1.1.4 upload\_data()

```
def program.upload_data (
    table )
```

Szuka atomów i wiązań istotnych dla tworzenia wiązania.

## Parametry

<i>table</i>	tablica z danymi wczytanymi z pliku cif
--------------	---

## Zwraca

obiekty klasy monomer

## 4.1.2 Dokumentacja zmiennych

## 4.1.2.1 aa

```
program.aa = peptide()
```

## 4.1.2.2 data

```
program.data = open(sys.argv[1])
```

Tworzy peptyd o zadanych kątach omega, fi, psi.

Program wczytuje dane z pliku data.txt. W pliku wejściowym w pierwszej kolumnie musi znajdować się trzyliterowy kod monomeru. W kolejnych trzech kolumnach mogą znajdować się wartości kątów omega, fi, psi (w takiej kolejności) podane w stopniach. Jeśli liczba kolumn w pliku jest różna od 4 (plik nie zawiera dokładnie trzech wartości kątów), zostają automatycznie ustalone następujące wartości: omega=180 fi=(-60) psi=(-45) /author Barbara Gruza

## 4.1.2.3 f

```
program.f = open('output.pdb', 'w')
```

Drukuje stworzony peptyd do pliku output.pdb.

Korzystając z funkcji make\_output tworzy plik zawierający stworzony obiekt klasy peptyd w formacie .pdb /author Barbara Gruza

## 4.1.2.4 file

```
program.file = open('Components-pub.cif')
```

Tworzy monomery białkowe.

Program wczytuje dane z pliku Components-pub.cif do tablicy 'table'. Wybiera L-peptydy (oraz glicynę). Korzystając z funkcji 'upload\_data' tworzy listę obiektów klasy monomer. Jeśli monomer nie zawiera poprawnie zdefiniowanych nazw oraz współrzędnych atomów grup aminowej, karboksylowej oraz węgla alfa nie zostanie załadowany do tablicy /author Barbara Gruza

**4.1.2.5 i**

```
int program.i = 0
```

**4.1.2.6 monomer\_i**

```
def program.monomer_i = upload_data( table)
```

**4.1.2.7 monomers\_list**

```
dictionary program.monomers_list = {}
```

**4.1.2.8 n**

```
int program.n = 0
```

**4.1.2.9 number\_of\_lines**

```
int program.number_of_lines = 0
```

**4.1.2.10 table**

```
list program.table = []
```

**4.1.2.11 table2**

```
list program.table2 = []
```

**4.1.2.12 words**

```
program.words = re.split("\s+",line.strip())
```

**4.1.2.13 words2**

```
program.words2 = ""
```

## 5 Dokumentacja klas

### 5.1 Dokumentacja klasy program.atom

Atomy.

#### Metody publiczne

- def **\_\_init\_\_** (\_\_self\_\_, a, b, c, **ID**, **element**, **znacznik**)  
*Konstruktor.*
- def **scalar\_prod** (\_\_self\_\_, p)  
*Iloczyn skalarny.*
- def **rotate** (\_\_self\_\_, m)  
*Obrót.*
- def **translation** (\_\_self\_\_, v)  
*Przesunięcie.*
- def **vector\_prod** (\_\_self\_\_, p)  
*Iloczyn wektorowy.*
- def **\_\_str\_\_** (\_\_self\_\_)  
*Pozwala na wydrukowanie atomu.*

#### Statyczne atrybuty publiczne

- int **x** = 0
- int **y** = 0
- int **z** = 0
- string **ID** = ""
- string **element** = ""
- string **znacznik** = ""

#### 5.1.1 Opis szczegółowy

Atomy.

#### Autor

Bernadeta Nowosielska

#### Parametry

<i>x,y,z</i>	współrzędne w układzie kartezjańskim
<i>ID</i>	unikalny identyfikator
<i>element</i>	pierwiastek chemiczny
<i>znacznik</i>	atomy "funkcyjne"



## 5.1.2 Dokumentacja konstruktora i destruktora

### 5.1.2.1 `__init__()`

```
def program.atom.__init__ (
    __self__,
    a,
    b,
    c,
    ID,
    element,
    znacznik )
```

Konstruktor.

#### Parametry

<i>a,b,c</i>	współrzędne
<i>ID</i>	unikalny identyfikator
<i>element</i>	pierwiastek chemiczny
<i>znacznik</i>	dla atomów funkcyjnych

## 5.1.3 Dokumentacja funkcji składowych

### 5.1.3.1 `__str__()`

```
def program.atom.__str__ (
    __self__ )
```

Pozwala na wydrukowanie atomu.

### 5.1.3.2 `rotate()`

```
def program.atom.rotate (
    __self__,
    m )
```

Obrót.

#### Parametry

<i>m</i>	macierz obrotu
----------	----------------

Zmienia współrzędne atomu

#### 5.1.3.3 scalar\_prod()

```
def program.atom.scalar_prod (
    __self__,
    p )
```

Iloczyn skalarny.

##### Parametry

$p$	atom lub wektor
-----	-----------------

##### Zwraca

wynik iloczynu skalarnego

#### 5.1.3.4 translation()

```
def program.atom.translation (
    __self__,
    v )
```

Przesunięcie.

##### Parametry

$v$	wektor
-----	--------

Zmienia współrzędne atomu

#### 5.1.3.5 vector\_prod()

```
def program.atom.vector_prod (
    __self__,
    p )
```

Iloczyn wektorowy.

##### Parametry

$p$	atom lub wektor
-----	-----------------

##### Zwraca

obiekt klasy wektor będący wynikiem iloczynu wektorowego

#### 5.1.4 Dokumentacja atrybutów składowych

#### 5.1.4.1 element

```
string program.atom.element = "" [static]
```

#### 5.1.4.2 ID

```
string program.atom.ID = "" [static]
```

#### 5.1.4.3 x

```
int program.atom.x = 0 [static]
```

#### 5.1.4.4 y

```
int program.atom.y = 0 [static]
```

#### 5.1.4.5 z

```
int program.atom.z = 0 [static]
```

#### 5.1.4.6 znacznik

```
string program.atom.znacznik = "" [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- **program.py**

## 5.2 Dokumentacja klasy program.monomer

Monomery.

## Metody publiczne

- def **\_\_init\_\_** (\_\_self\_\_, ID, noa, atoms, nob, bonds, wazne\_atomy)  
*Konstruktor.*
- def **replicate** (\_\_self\_\_)  
*Tworzy wierną kopię monomeru.*
- def **rotate\_all** (\_\_self\_\_, m)  
*Obrót wszystkich atomu monomeru.*
- def **translation\_all** (\_\_self\_\_, v)  
*Przesunięcie wszystkich atomów monomeru.*
- def **system\_C** (\_\_self\_\_)  
*Tworzy macierz układu współrzędnych na C.*
- def **system\_N** (\_\_self\_\_)  
*Tworzy macierz układu współrzędnych na C.*
- def **remove** (\_\_self\_\_, ID)  
*Usuwa atom.*
- def **\_\_str\_\_** (\_\_self\_\_)  
*Pozwala na wydrukowanie monomru.*

## Statyczne atrybuty publiczne

- string **ID** = ""
- int **number\_of\_atoms** = 0
- int **number\_of\_bonds** = 0

## 5.2.1 Opis szczegółowy

Monomery.

Autor

Bernadeta Nowosielska

## Parametry

<i>ID</i>	nazwa monomeru
<i>number_of_atoms</i>	liczba atomów w monomerze
<i>number_of_bonds</i>	liczba wiązań w monomerze

## 5.2.2 Dokumentacja konstruktora i destruktor

## 5.2.2.1 \_\_init\_\_()

```
def program.monomer.__init__ (  
    __self__,
```

```
ID,  
noa,  
atoms,  
nob,  
bonds,  
wazne_atomy )
```

Konstruktor.

#### Parametry

<i>ID</i>	nazwa
<i>noa</i>	liczba atomów w monomerze
<i>atoms</i>	tablica zawierająca obiekty klasy atom
<i>nob</i>	liczba wiązań w monomerze
<i>bonds</i>	tablica wiązań
<i>wazne_atomy</i>	tablica atomów funkcyjnych

### 5.2.3 Dokumentacja funkcji składowych

#### 5.2.3.1 \_\_str\_\_()

```
def program.monomer.__str__ (  
    __self__ )
```

Pozwala na wydrukowanie monomru.

#### 5.2.3.2 remove()

```
def program.monomer.remove (  
    __self__,  
    ID )
```

Usuwa atom.

#### Parametry

<i>ID</i>	ID atomu który będzie usunięty
-----------	--------------------------------

#### 5.2.3.3 replicate()

```
def program.monomer.replicate (  
    __self__ )
```

Tworzy wierną kopję monomeru.

#### 5.2.3.4 rotate\_all()

```
def program.monomer.rotate_all (
    __self__,
    m )
```

Obrót wszystkich atomu monomeru.

##### Parametry

<i>m</i>	macierz obrotu
----------	----------------

#### 5.2.3.5 system\_C()

```
def program.monomer.system_C (
    __self__ )
```

Tworzy macierz układu współrzędnych na C.

##### Zwraca

###### Macierz układu

Tworzy macierz układu współrzędnych na karbonylowym atomie węgla. Tak aby oś x znajdowała się na wiązaniu C-OH, a oś y była prostopadła do grupy.

#### 5.2.3.6 system\_N()

```
def program.monomer.system_N (
    __self__ )
```

Tworzy macierz układu współrzędnych na C.

##### Zwraca

###### Macierz układu

Tworzy macierz układu współrzędnych na karbonylowym atomie węgla. Tak aby oś x znajdowała się na wiązaniu N-X, a oś y była prostopadła do atomów H i węgla alpha.

#### 5.2.3.7 translation\_all()

```
def program.monomer.translation_all (
    __self__,
    v )
```

Przesunięcie wszystkich atomów monomeru.

## Parametry

$v$	wektor, o który następuje przesunięcie
-----	--

## 5.2.4 Dokumentacja atrybutów składowych

## 5.2.4.1 ID

```
string program.monomer.ID = "" [static]
```

## 5.2.4.2 number\_of\_atoms

```
int program.monomer.number_of_atoms = 0 [static]
```

## 5.2.4.3 number\_of\_bonds

```
int program.monomer.number_of_bonds = 0 [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- **program.py**

## 5.3 Dokumentacja klasy program.peptide

Peptydy.

## Metody publiczne

- def **start** (\_\_self\_\_, monomer1)  
*Dodaje pierwszy monomer do peptydu.*
- def **add** (\_\_self\_\_, monomer3, omega\_i, fi\_i, psi\_i)  
*Dodaje kolejne monomery.*
- def **\_\_str\_\_** (\_\_self\_\_)  
*Pozwala na wydrukowanie peptydu.*
- def **rotate\_da** (\_\_self\_\_, type\_, angle, N, C, m3)  
*Obraca o kąt torsyjny.*

## Statyczne atrybuty publiczne

- list **coord** = []
- list **A** = []

## 5.3.1 Opis szczegółowy

Peptydy.

Autor

Bernadeta Nowosielska

## Parametry

<i>coord</i>	tablica zawierająca tablicę, która zawiera ID monomeru, jego licznik oraz tablicę obiektów klasy atom.
--------------	--

## 5.3.2 Dokumentacja funkcji składowych

5.3.2.1 `__str__()`

```
def program.peptide.__str__ (
    __self__ )
```

Pozwala na wydrukowanie peptydu.

5.3.2.2 `add()`

```
def program.peptide.add (
    __self__,
    monomer3,
    omega_i,
    fi_i,
    psi_i )
```

Dodaje kolejne monomery.

## Parametry

<i>monomer3</i>	dodawany monomer
<i>omega_i, fi_i, psi_i</i>	kąty torsyjne, które zostaną ustawione

5.3.2.3 `rotate_da()`

```
def program.peptide.rotate_da (
    __self__,
    type_,
    angle,
    N,
    C,
    m3 )
```

Obraca o kąt torsyjny.



**Parametry**

<i>type</i> ↔ —	wybrany kąt torsyjny "omega", "fi" lub "psi"
<i>angle</i>	wartość o którą nastąpi obrót
<i>N,C</i>	końcowy i początkowy atom wiązania, na którym będzie następować obrót (obiekt klasy atom)
<i>m3</i>	monomer, który będzie obracany

**5.3.2.4 start()**

```
def program.peptide.start (
    __self__,
    monomer1 )
```

Dodaje pierwszy monomer do peptydu.

**Parametry**

<i>monomer1</i>	obiekt klasy monomer
-----------------	----------------------

**5.3.3 Dokumentacja atrybutów składowych****5.3.3.1 A**

```
list program.peptide.A = [] [static]
```

**5.3.3.2 coord**

```
list program.peptide.coord = [] [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- **program.py**

**5.4 Dokumentacja klasy program.vector**

Wektory w układzie kartezjańskim.

## Metody publiczne

- `def __init__ (__self__, a, b, c)`  
*Konstruktor.*
- `def leng (__self__)`  
*Długość wektora.*
- `def vector_prod (__self__, p)`  
*Iloczyn wektorowy.*
- `def scalar_prod (__self__, p)`  
*Iloczyn skalarny.*
- `def __str__ (__self__)`  
*Pozwala na wydrukowanie wektora.*

## Statyczne atrybuty publiczne

- `int x = 0`
- `int y = 0`
- `int z = 0`

## 5.4.1 Opis szczegółowy

Wektory w układzie kartezjańskim.

## Autor

Bernadeta Nowosielska

## Parametry

<code>x,y,z</code>	współrzędne w układzie kartezjańskim
--------------------	--------------------------------------

## 5.4.2 Dokumentacja konstruktora i destruktora

5.4.2.1 `__init__()`

```
def program.vector.__init__ (
    __self__,
    a,
    b,
    c )
```

Konstruktor.

## Parametry

<code>a,b,c</code>	pobierane współrzędne
--------------------	-----------------------

### 5.4.3 Dokumentacja funkcji składowych

#### 5.4.3.1 `__str__()`

```
def program.vector.__str__ (
    __self__ )
```

Pozwala na wydrukowanie wektora.

#### 5.4.3.2 `leng()`

```
def program.vector.leng (
    __self__ )
```

Długość wektora.

#### Zwraca

długość wektora

#### 5.4.3.3 `scalar_prod()`

```
def program.vector.scalar_prod (
    __self__,
    p )
```

Iloczyn skalarny.

#### Parametry

$p$	atom lub wektor
-----	-----------------

#### Zwraca

wynik iloczynu skalarnego

#### 5.4.3.4 `vector_prod()`

```
def program.vector.vector_prod (
    __self__,
    p )
```

Iloczyn wektorowy.

## Parametry

$p$	atom lub wektor
-----	-----------------

## Zwraca

obiekt klasy wektor

## 5.4.4 Dokumentacja atrybutów składowych

5.4.4.1  $x$ 

```
int program.vector.x = 0 [static]
```

5.4.4.2  $y$ 

```
int program.vector.y = 0 [static]
```

5.4.4.3  $z$ 

```
int program.vector.z = 0 [static]
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- **program.py**

## 6 Dokumentacja plików

### 6.1 Dokumentacja pliku program.py

## Komponenty

- class **program.vector**  
*Wektory w układzie kartezjańskim.*
- class **program.atom**  
*Atomy.*
- class **program.monomer**  
*Monomery.*
- class **program.peptide**  
*Peptydy.*

## Przestrzenie nazw

- **program**

## Funkcje

- **def program.transpose (m)**  
*Transpozycja macierzy.*
- **def program.dihedral\_angle (b1, b2, b3)**  
*Wyznaczanie kąta torsyjnego.*
- **def program.upload\_data (table)**  
*Szuka atomów i wiązań istotnych dla tworzenia wiązania.*
- **def program.make\_output (peptyd, i, j, n)**  
*Tworzy plik w formacie pdb.*

## Zmienne

- **program.file** = open('Components-pub.cif')  
*Tworzy monomery białkowe.*
- **int program.i** = 0
- **list program.table** = []
- **int program.number\_of\_lines** = 0
- **dictionary program.monomers\_list** = {}
- **program.words** = re.split("\s+", line.strip())
- **def program.monomer\_i** = upload\_data(table)
- **program.data** = open(sys.argv[1])  
*Tworzy poptyd o zadanych kątach omega, fi, psi.*
- **string program.words2** = ""
- **list program.table2** = []
- **program.aa** = peptide()
- **program.f** = open('output.pdb', 'w')  
*Drukuje stworzony peptyd do pliku output.pdb.*
- **int program.n** = 0