

---

# **RapidPeptidesGenerator Documentation**

*Release 1.2.4*

**Nicolas Maillet**

**May 25, 2021**



---

## Contents:

---

<b>1</b>	<b>Overview</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
<b>3</b>	<b>Usage</b>	<b>7</b>
<b>4</b>	<b>User and Developer Guides</b>	<b>9</b>
4.1	User Guide . . . . .	9
4.1.1	Overview . . . . .	9
4.1.2	Installation . . . . .	9
4.1.2.1	From pip . . . . .	9
4.1.2.2	From source code . . . . .	9
4.1.2.3	Using without installation . . . . .	10
4.1.3	Classical use . . . . .	10
4.1.3.1	Getting help . . . . .	10
4.1.3.2	Listing enzymes . . . . .	10
4.1.3.3	Performing digestion . . . . .	10
4.1.3.4	Adding a new enzyme . . . . .	11
4.1.4	Options . . . . .	11
4.1.5	Digestion modes . . . . .	12
4.1.6	Miscleavage . . . . .	13
4.1.7	Non-interactive mode . . . . .	13
4.1.8	Output . . . . .	13
4.1.9	Random names . . . . .	14
4.1.10	Verbosity . . . . .	14
4.1.11	Creating a new enzyme . . . . .	15
4.1.11.1	Definition of rules . . . . .	16
4.1.11.2	Definition of exceptions . . . . .	18
4.1.11.3	Easily writing complex enzymes . . . . .	19
4.1.11.4	Example of enzymes . . . . .	21
4.1.12	Deleting user-defined enzymes . . . . .	21
4.2	Enzyme definitions . . . . .	21
4.2.1	Available enzymes . . . . .	22
4.2.1.1	Arg-C . . . . .	22
4.2.1.2	Asp-N . . . . .	22
4.2.1.3	BNPS-Skatole . . . . .	23
4.2.1.4	Bromelain . . . . .	23

4.2.1.5	Caspase 1	23
4.2.1.6	Caspase 2	23
4.2.1.7	Caspase 3	24
4.2.1.8	Caspase 4	24
4.2.1.9	Caspase 5	24
4.2.1.10	Caspase 6	24
4.2.1.11	Caspase 7	25
4.2.1.12	Caspase 8	25
4.2.1.13	Caspase 9	25
4.2.1.14	Caspase 10	25
4.2.1.15	Chymotrypsin high specificity	26
4.2.1.16	Chymotrypsin low specificity	26
4.2.1.17	Clostripain	26
4.2.1.18	CNBr	27
4.2.1.19	Enterokinase	27
4.2.1.20	Factor Xa	27
4.2.1.21	Ficin	27
4.2.1.22	Formic acid	27
4.2.1.23	Glu-C	28
4.2.1.24	Glutamyl endopeptidase	28
4.2.1.25	Granzyme B	28
4.2.1.26	Hydroxylamine	28
4.2.1.27	Iodosobenzoic acid	28
4.2.1.28	Lys-C	29
4.2.1.29	Lys-N	29
4.2.1.30	Neutrophil elastase	29
4.2.1.31	NTCB	29
4.2.1.32	Papain	29
4.2.1.33	Pepsin pH 1.3	30
4.2.1.34	Pepsin pH >=2	30
4.2.1.35	Proline-endopeptidase	31
4.2.1.36	Proteinase K	31
4.2.1.37	Staphylococcal peptidase I	31
4.2.1.38	Thermolysin	31
4.2.1.39	Thrombin (PeptideCutter)	32
4.2.1.40	Thrombin SG	32
4.2.1.41	Tobacco etch virus protease	32
4.2.1.42	Trypsin	33
4.3	rpg	33
4.3.1	rpg package	33
4.3.1.1	Submodules	33
4.3.1.2	rpg.RapidPeptidesGenerator module	33
4.3.1.3	rpg.context module	34
4.3.1.4	rpg.core module	34
4.3.1.5	rpg.digest module	36
4.3.1.6	rpg.enzyme module	39
4.3.1.7	rpg.enzymes_definition module	39
4.3.1.8	rpg.rule module	39
4.3.1.9	rpg.sequence module	42
4.3.1.10	Module contents	43
4.4	CHANGELOG	43





Rapid Peptides Generator (RPG) is a software dedicated to predict proteases-induced cleavage sites on amino acid sequences.

**note** RPG is tested with Gitlab Ci for the following Python version: 3.6 to 3.9

**issues** Please use <https://gitlab.pasteur.fr/nmaillet/rpg>

**publication** To cite RPG, please refer to <https://doi.org/10.1093/nargab/lqz004>





Rapid Peptides Generator (RPG), is a standalone software dedicated to predict proteases-induced cleavage sites on sequences.

RPG is a python tool taking a (multi-)fasta/fastq file (gzipped or not) of proteins as input and digest each of them. The digestion mode can be either 'concurrent', i.e. all enzymes are present at the same time during digestion, or 'sequential'. In sequential mode, each protein will be digested by each enzyme, one by one.

The resulting peptides contain informations about positions of cleavage site, peptide sequences, length, mass as-well as an estimation of isoelectric point (pI) of each peptide. Shortly, the isoelectric point is the pH at which a peptide carries no net electrical charge and a good approximation can be computed on small molecules. Results are outputted in multi-fasta, CSV or TSV file.

Currently, 42 enzymes and chemicals are included in RPG. The user can easily design new enzymes, using a simple yet powerful grammar. This grammar allows the user to design complex enzymes like trypsin or thrombin, including many exceptions and different cleavage sites. User-defined enzymes are then interpreted by RPG and included in the local installation of the software.

RPG follows the standards for software development with continuous integration on Gitlab (<https://gitlab.pasteur.fr/nmaillet/rpg>) and automatic on-line documentation (<https://rapid-peptide-generator.readthedocs.io>).



## CHAPTER 2

---

### Installation

---

In order to install RPG, you can use **pip**:

```
pip3 install rpg
```

This command installs RPG and its Python dependencies.



## CHAPTER 3

---

### Usage

---

From the command line:

```
rpg --help
```



### 4.1 User Guide

#### 4.1.1 Overview

You can run **Rapide Peptides Generator** using the standalone version called:

```
rpg
```

You can obtain help by using:

```
rpg --help
```

#### 4.1.2 Installation

##### 4.1.2.1 From pip

The suggested way of installing the latest **RPG** version is through **pip**:

```
pip3 install rpg
```

Then you can use:

```
rpg --help
```

##### 4.1.2.2 From source code

**RPG** is coded in Python. To manually install it from source, get the source and install **RPG** using:

```
git clone https://gitlab.pasteur.fr/nmaillet/rpg/  
cd rpg  
python setup.py install
```

### 4.1.2.3 Using without installation

You can download the source code from Pasteur's **Gitlab**: <https://gitlab.pasteur.fr/nmaillet/rpg/>.

In order to directly run **RPG** from source, you need to copy file `tests/context.py` into `rpg` folder.

Then, uncomment line 42 of `rpg/RapidePeptideGenerator.py`. Modify:

```
#from context import rpg
```

To:

```
from context import rpg
```

Then, from the main **RPG** directory, use:

```
python3 rpg/RapidePeptidesGenerator.py -\\-help
```

**Warning:** Using without installation is not recommended, as you need all requirements of `requirements.txt` installed locally and you may encounter issues with Sphinx autodoc or other unwanted behaviors.

## 4.1.3 Classical use

Here are some typical examples of **RPG** usage.

### 4.1.3.1 Getting help

To access build-in help, use:

```
rpg -\\-help
```

### 4.1.3.2 Listing enzymes

To list all available enzymes, use:

```
rpg -l
```

### 4.1.3.3 Performing digestion

There are two digestion modes in **RPG**. In sequential mode, each protein will be digested by each enzyme, one by one. In concurrent mode, all enzymes are present at the same time during digestion. See *Digestion modes* for more information.



### Sequential digestion of one sequence

To perform sequential digestion of the sequence “QWSDORESDF” with enzymes 2 and 5 and store results in *output\_file.fasta*, use:

```
rpg -s QWSDORESDF -o output_file.fasta -e 2 5
```

### Sequential digestion of a (multi)fasta file

To perform sequential digestion of *input\_file.fasta* with enzymes 2 and 5 and store results in *output\_file.fasta*, use:

```
rpg -i input_file.fasta -o output_file.fasta -e 2 5
```

### Concurrent digestion of a (multi)fasta file

To perform concurrent digestion of *input\_file.fasta* with enzymes 2 and 5 and store results in *output\_file.fasta*, use:

```
rpg -i input_file.fasta -o output_file.fasta -e 2 5 -d c
```

#### 4.1.3.4 Adding a new enzyme

To extend the list of the available enzymes and add a new one, use:

```
rpg -a
```

See *Creating a new enzyme* for more information.

### 4.1.4 Options

Here are all available options in **RPG**:

- h, --help:** Show this help message and exit.
- a, --addenzyme:** Add a new user-defined enzyme. See *Creating a new enzyme* for more information.
- d, --digest:** Digestion mode. Either ‘s’, ‘sequential’, ‘c’ or ‘concurrent’ (default: s). See *Digestion modes* for more information.
- e, --enzymes:** Enzyme(s) id number to use (*i.e.* -e 0 5 10 to use enzymes 0, 5 and 10). Use -l first to get enzyme ids. See *Enzyme definitions* for more information.
- f, --fmt:** Output file format. Either ‘fasta’, ‘csv’, or ‘tsv’ (default: fasta). See *Output* for more information.
- i, --inputdata:** Input file, in (multi)fasta / fastq format (gzipped or not). See *Sequential digestion of a (multi)fasta file* for example.
- s, --sequence:** Input a single protein sequence without commentary. See *Sequential digestion of one sequence* for example.
- l, --list:** Display the list of available enzymes.
- m, --miscleavage:** Percentage of miscleavage, between 0 and 100, by enzyme(s). It should be in the same order as the `-enzymes` options (*i.e.* -m 15 5.2 10). It works only for sequential digestion (default: 0). See *Miscleavage* for more information.

**-n, --noninteractive:** Non-interactive mode. No standard output, only error(s) (`--quiet` enable, overwrite `-v`). If output filename already exists, output file will be overwritten. See *Non-interactive mode* for more information.

**-o, --outputfile:** Result file to output resulting peptides (default `./peptides.xxx` depending of `-fmt`).

**-p, --pka:** Define pKa values. Either `ipc2`, `stryer` or `ipc` (default: `ipc2`). IPC2 values come from *IPC\_peptide* in *Supplementary Table S1*, Stryer values from *Biochemistry Stryer*, 7th edition and IPC values from *IPC\_peptide*.

**-r, --randomname:** Random (not used) output name file. See *Random names* for more information.

**-c, --processes:** Number of parallel processes to use (default: 1)

**-q, --quiet:** No standard output, only error(s).

**-v, --verbose:** Increase output verbosity. `-vv` will increased more than `-v`. See *Verbosity* for more information.

**--version:** Show program's version number and exit.

### 4.1.5 Digestion modes

There are two digestion modes in **RPG**. In `'sequential'` mode, each protein will be digested by each enzyme, one by one. Launching 3 times **RPG** on the same protein with 3 different enzymes or launching one time **RPG** on the protein with the 3 enzymes in `'sequential'` mode leads to exactly the same result.

In concurrent mode, all enzymes are present at the same time during digestion and exposure time is supposed to be infinite, *i.e.* all possible cleavages **will** occur (there is no miscleavage). In this mode, the cleavage of a first enzyme can make available the cleavage site of another enzyme.

Let's define two enzymes. The first is called `'afterP'` (id 28) and cleaves after P. The second is called `'afterK'` (id 29) and cleaves after K if there is no P just before. Digesting `'PKPKPKPK'` using those two enzymes in sequential mode gives the following result (see *Output* for more information):

```
$ rpg -s PKPKPKPK -e 28 29
>Input_0_afterP_1_1_115.13198_5.54
P
>Input_1_afterP_3_2_243.30608_9.4
KP
>Input_2_afterP_5_2_243.30608_9.4
KP
>Input_3_afterP_7_2_243.30608_9.4
KP
>Input_4_afterP_8_1_146.18938_9.4
K
>Input_0_afterK_0_8_919.17848_11.27
PKPKPKPK
```

`'afterP'` cleaves as expected and `'afterK'` is not able to cleave anything.

Digesting `'PKPKPKPK'` using those two enzymes in concurrent mode gives the following result:

```
$ rpg -s PKPKPKPK -e 28 29 -d c
>Input_0_afterP-afterK_1_1_115.13198_5.54
P
>Input_1_afterP-afterK_2_1_146.18938_9.4
K
>Input_2_afterP-afterK_3_1_115.13198_5.54
P
>Input_3_afterP-afterK_4_1_146.18938_9.4
K
>Input_4_afterP-afterK_5_1_115.13198_5.54
```

(continues on next page)

(continued from previous page)

```
P
>Input_5_afterP-afterK_6_1_146.18938_9.4
K
>Input_6_afterP-afterK_7_1_115.13198_5.54
P
>Input_7_afterP-afterK_8_1_146.18938_9.4
K
```

Here, we have to understand that ‘afterP’ cleaves at the same positions as in sequential mode and the products (mostly ‘KP’) are then cleaved by ‘afterK’. Indeed, there is no more P before K, making ‘afterK’ able to cleave.

Default mode is ‘sequential’. Reminder: you can input miscleavage values only for this mode.

### 4.1.6 Miscleavage

Sometimes an enzyme does not cleave at a given position even if requirements are fulfilled. This event is called miscleavage and can have biological, chemical or physical origins. To take into account this behavior in **RPG**, one can assign a miscleavage value to an enzyme, expressed as a **percentage**.

For example, using:

```
rpg -s QWSDORESDF -e 1 2 3 -m 1.4 2.6
```

will assign a miscleavage probability of *1.4%* to enzyme *1*, a miscleavage probability of *2.6%* to enzyme *2* and a miscleavage probability of *0%* to enzyme *3* (default behavior). For enzyme *1*, each cleavage will then have a probability of 0.014 to **not** occur.

### 4.1.7 Non-interactive mode

Option **-n**, **-noninteractive** force **RPG** to not print any standard output, only error(s) are displayed in the shell. It enable ‘-quiet’ option and overwrites **-verbose** option. If output filename already exists, the output file will be systematically overwritten. This option is mostly used in cluster or pipeline when user does not want **RPG** to wait for input or display anything but errors.

### 4.1.8 Output

Output of **RPG** contains the following information in one line for each generated peptide, in this order:

- Header of original sequence or ‘Input’ if the sequence is directly inputed in **RPG**, *i.e.*, **-s**
- Sequential numbering (starting from 0) of out-coming peptides for each of original sequence
- Enzyme name used to obtain this peptide
- Cleavage position on the original sequence (0 if no cleavage occurs)
- Peptide size
- Peptide molecular weight estimation
- Peptide isoelectric point estimation (pI)

Then, on the next line:

- Peptide sequence

Peptide molecular weight approximation is computed as the addition of average isotopic masses of each amino acid present in the peptide. Then the average isotopic mass of one water molecule is added to it. Molecular weight values are given in Dalton (Da). It does not take into consideration any digestion-induced modifications.

The isoelectric point is computed by solving Henderson–Hasselbalch equation using binary search. It is based on Lukasz P. Kozlowski work (<http://isoelectric.org/index.html>).

The default output is in multi-fasta format. The header then summarizes all this information. For example, on the following multi-fasta result:

```
>Input_0_Asp-N_3_3_419.43738_5.54
QWS
>Input_1_Asp-N_8_5_742.78688_4.16
...
```

we can see that a sequence was directly inputted in **RPG** (*Input*), the first peptide (*0*) was obtained with *Asp-N* and this enzyme cleaved after the *3rd* amino acid in the original sequence. The peptide has a size of 3 amino acids, a molecular weight estimated at *419.43738* Da and a theoretical isoelectric point of *5.54*. The full sequence is then written (*QWS*). The output of the remaining peptides follows in the same format.

More information can be outputted using *Verbosity* option.

### 4.1.9 Random names

Option **-r**, **-randomname** force **RPG** to use a random name for output file. When using it, **RPG** will not ask user output file name **nor** location. The output file will be created in the working directory. This option is generally used for testing or automatic tasks.

### 4.1.10 Verbosity

Verbosity can be increased or decreased. The output file is not affected by **-v** or **-q** options.

With default verbosity level (no **-v** nor **-q** option), the output is:

```
$ rpg -s QWSDORESDF -e 1
>Input_0_Asp-N_3_3_419.43738_5.54
QWS
>Input_1_Asp-N_8_5_742.78688_4.16
DORES
>Input_2_Asp-N_10_2_280.28048_3.6
DF
```

Increasing verbosity by one, *i.e.* using **-v**, adds information about used options. For example:

```
$ rpg -s QWSDORESDF -e 1 -v
Warning: File 'peptides.fasta' already exist!
Overwrite it? (y/n)
y
Input: QWSDORESDF
Enzyme(s) used: ['Asp-N']
Mode: sequential
miscleavage ratio: [0]
Output file: /Users/nmaillet/Prog/RPG/peptides.fasta
>Input_0_Asp-N_3_3_419.43738_5.54
QWS
>Input_1_Asp-N_8_5_742.78688_4.16
```

(continues on next page)

(continued from previous page)

```
DORES
>Input_2_Asp-N_10_2_280.28048_3.6
DF
```

Increasing verbosity by two, *i.e.* using **-vv**, also adds statistics about each of the digested proteins. For example:

```
$ rpg -s QWSDORESDF -e 1 -vv
Warning: File 'peptides.fasta' already exist!
Overwrite it? (y/n)
y
Input: QWSDORESDF
Enzyme(s) used: ['Asp-N']
Mode: sequential
miscleavage ratio: [0]
Output file: /Users/nmaillet/Prog/RPG/peptides.fasta

Number of cleavage: 2
Cleavage position: 3, 8
Number of miscleavage: 0
miscleavage position:
miscleavage ratio: 0.00%
Smallest peptide size: 2
N terminal peptide: QWS
C terminal peptide: DF
>Input_0_Asp-N_3_3_419.43738_5.54
QWS
>Input_1_Asp-N_8_5_742.78688_4.16
DORES
>Input_2_Asp-N_10_2_280.28048_3.6
DF
```

Decreasing verbosity, *i.e.* using **-q** option, removes all information but errors. For example:

```
$ rpg -s QWSDORESDF -e 1 -q
Warning: File 'peptides.fasta' already exist!
Overwrite it? (y/n)
y
```

#### 4.1.11 Creating a new enzyme

Option **-a**, **--addenzyme** allows the user to define new enzymes. An enzyme contains one or several rules and exceptions.

In the following, nomenclature of [Schechter and Berger](#) is used. Amino acids before the cleavage site are designated as *P1*, *P2*, *P3*, etc in the N-terminal direction, and as *P1'*, *P2'*, *P3'*, etc in the C-terminal direction. For example, with cleavage site represented as 'I':

```
...P3-P2-P1-I-P1'-P2'-P3'...
```

In **RPG**, this nomenclature is represented as:

```
... (P3) (P2) (P1) (,) (P1') (P2') (P3') ...
```

#### 4.1.11.1 Definition of rules

A rule specifies which amino acid is targeted by the enzyme, the cleavage position (*i.e.* **before** or **after** the targeted amino acid) and optionally the surrounding context. Each amino acid must be included in parentheses, *i.e.* ‘(’ and ‘)’ and the cleavage position is represented by a comma, *i.e.* ‘,’. The comma must always be directly before or after a closing or opening parenthesis, respectively.

For example, to define a cleavage occurring **before** A, one must input:

```
(,A)
```

To define a cleavage occurring **after** B, one must input:

```
(B,)
```

The surrounding context is specified by adding other amino acids, before or after the targeted one. For example, to define a cleavage occurring **before** A, position *P1*, preceded by B in position *P1*, C in position *P3* and followed by D in position *P2*, one must input:

```
(C)() (B) (,A) (D)
```

Note that this enzyme will only cleave if it finds the motif C\*BAD, where \* could be **any** amino acid. It will **not** cleave BAD, nor C\*BA, BA, etc. For example, creating and using enzyme *rpg\_example\_userguide* (enzyme id 43):

```
$ rpg -a
Name of the new enzyme?
rpg_example_userguide
Create a cleaving rule (c) or an exception (e)? (q) to quit:
c
Write your cleaving rule, (q) to quit:
(C)() (B) (,A) (D)
Create a cleaving rule (c) or an exception (e)? (q) to quit:
q
Add another enzyme? (y/n)
n

$ rpg -s CWBADE -e 43
>Input_0_rpg_example_userguide_3_3_307.36728_5.46
CWB
>Input_1_rpg_example_userguide_6_3_333.29818_3.4
ADE

$ rpg -s FAD -e 43
>Input_0_rpg_example_userguide_0_3_351.35928_3.6
FAD
```

In order for this enzyme to also cleave before AD (before A in *P1* followed by D in *P2*), on top of the previous rule, one has to define one more rule in **RPG**:

```
(,A) (D)
(C)() (B) (,A) (D)
```

It is important to note that for each enzyme, it is enough that one of the rule is broken for the cleavage to not occur. In this example, the defined enzyme will **not** cleave BAD, as it is specified that it will cleave before A preceded by B in *P1* if there is C in ‘*P3*’. Identically, it will **not** cleave C\*BA\*, as D is required in *P2* for both rules.

```

$ rpg -a
Name of the new enzyme?
rpg_example_userguide
Create a cleaving rule (c) or an exception (e)? (q) to quit:
c
Write your cleaving rule, (q) to quit:
(,A) (D)
Create a cleaving rule (c) or an exception (e)? (q) to quit:
c
Write your cleaving rule, (q) to quit:
(C) ( ) (B) (,A) (D)
Create a cleaving rule (c) or an exception (e)? (q) to quit:
q
Add another enzyme? (y/n)
n

$ rpg -s CWBADE -e 43
>Input_0_rpg_example_userguide_3_3_307.36728_5.46
CWB
>Input_1_rpg_example_userguide_6_3_333.29818_3.4
ADE

$ rpg -s FAD -e 43
>Input_0_rpg_example_userguide_1_1_165.19188_5.54
F
>Input_1_rpg_example_userguide_3_2_204.18268_3.6
AD

$ rpg -s BAD -e 43
>Input_0_rpg_example_userguide_0_3_204.18268_3.6
BAD

```

The order of inputted rules is not relevant. In other words, this enzyme:

```
(,A) (D)
(C) ( ) (B) (,A) (D)
```

and this second one:

```
(C) ( ) (B) (,A) (D)
(,A) (D)
```

are identical.

It is possible to define none-related cleavage rules for the same enzyme, for example:

```
(G, ) (G)
(P) (W, ) (E) (T)
```

This enzyme will cleave after G (position *PI*) followed by G in *PI'* and also after W (*PI*) preceded by P in *P2* and followed by E in *PI'* and T in *P2'*.

Note that each rule must concern only **one** cleavage site. It is not possible to input rule like:

```
(A, ) (B, )
```

This would define an enzyme cleaving after A in *PI* followed by B in *PI'* but also cleaving after B in *PI* preceded by A in *P2*. The proper way to input this is by using two separate rules:

```
(A, ) (B)
(A) (B, )
```

However, it is possible to write rules in a more efficient way as explained in *Easily writing complex enzymes*.

#### 4.1.11.2 Definition of exceptions

An exception specifies when a cleavage should **not** occur. **Exceptions must always be linked to a rule.**

For example, to define a cleavage occurring **before** A ( $P1'$ ), one must input:

```
(, A)
```

Exceptions can then be inputted. For example, to define “a cleavage occurs before A, except when P is in  $P2'$ “, the following exception needs to be added:

```
(, A) (P)
```

This enzyme will always cleave before A when not followed by P:

```
rpg -a
Name of the new enzyme?
rpg_example_userguide
Create a cleaving rule (c) or an exception (e)? (q) to quit:
c
Write your cleaving rule, (q) to quit:
(, A)
Create a cleaving rule (c) or an exception (e)? (q) to quit:
e
Write your exception rule, (q) to quit:
(, A) (P)
Create a cleaving rule (c) or an exception (e)? (q) to quit:
q
Add another enzyme? (y/n)
n

rpg -s CWBADE -e 43
>Input_0_rpg_example_userguide_3_3_307.36728_5.46
CWB
>Input_1_rpg_example_userguide_6_3_333.29818_3.4
ADE

rpg -s CWBAPE -e 43
>Input_0_rpg_example_userguide_0_6_604.67828_3.6
CWBAPE
```

It is possible to input complex exceptions. For the previous enzyme, we can add the following exception:

```
(G) (T) () (, A) () (F)
```

This enzyme will always cleave before A ( $P1'$ ) when not followed by P ( $P2'$ ) or preceded by G in  $P3$ , T in  $P2$ , by any amino acid in  $P1$  and  $P2'$ , and F in  $P3'$  **at the same time**:

```
rpg -a
Name of the new enzyme?
rpg_example_userguide
```

(continues on next page)



(continued from previous page)

```

Create a cleaving rule (c) or an exception (e)? (q) to quit:
c
Write your cleaving rule, (q) to quit:
(,A)
Create a cleaving rule (c) or an exception (e)? (q) to quit:
e
Write your exception rule, (q) to quit:
(,A)(P)
Create a cleaving rule (c) or an exception (e)? (q) to quit:
e
Write your exception rule, (q) to quit:
(G)(T)() (,A)() (F)
Create a cleaving rule (c) or an exception (e)? (q) to quit:
q
Add another enzyme? (y/n)
n

rpg -s CWBADE -e 43
>Input_0_rpg_example_userguide_3_3_307.36728_5.46
CWB
>Input_1_rpg_example_userguide_6_3_333.29818_3.4
ADE

rpg -s CWBAPE -e 43
>Input_0_rpg_example_userguide_0_6_604.67828_3.6
CWBAPE

rpg -s GTBAMF -e 43
>Input_0_rpg_example_userguide_0_6_525.62028_5.54
GTBAMF

rpg -s GTBAPE -e 43
>Input_0_rpg_example_userguide_0_6_473.48328_3.6
GTBAPE

rpg -s GTBAME -e 43
>Input_0_rpg_example_userguide_3_3_176.17228_5.54
GTB
>Input_1_rpg_example_userguide_6_3_349.40218_3.6
AME

```

It is important to understand that an exception should always be linked to a rule. If one inputs this rule:

```
(A,)
```

followed by this exception:

```
(B,) (C)
```

the exception will not be taken into account. This enzyme will just always cleave after A.

#### 4.1.11.3 Easily writing complex enzymes

To make enzyme creation easier to use, two tricks are available.

The first one simplifies the definition of enzymes cleaving **before** and **after** a given amino acid. Defining an enzyme cleaving, for example, before **and** after A, can be done with two rules:

```
(,A)
(A,)
```

or simply using:

```
(,A,)
```

The second trick is the use of the keyword *or*. This allows multiple possibilities for on position. For example:

```
(,A or B)
```

is equivalent to:

```
(,A)
(,B)
```

**Warning:** do not input `(,A or ,B)`, as a comma must always directly preceding or following a parenthesis.

Those two tricks help on complex enzymes. For example, *Pepsin pH 1.3* preferentially cleaves around F or L, sometimes before, sometimes after, depending on the context. More specifically, it will not cleave before F or L in *P1'* followed by P in *P2'*. It will not cleave before F or L in *P1'* preceded by R in *P1* or P in *P2* or H/K/R in *P3*. It will not cleave after F or L in *P1* followed by P in *P2'*. And it will not cleave after F or L in *P1* preceded by P in *P2* or H/K/R in *P3*.

It can be defined either by:

```
cleaving rules:
```

```
(F,)
(L,)
(,F)
(,L)
```

```
exception rules:
```

```
(,F) (P)
(,L) (P)
(R) (,F)
(R) (,L)
(P) () (,F)
(P) () (,L)
(H) () () (,F)
(K) () () (,F)
(R) () () (,F)
(H) () () (,L)
(K) () () (,L)
(R) () () (,L)
(F,) () (P)
(L,) () (P)
(P) (F,)
(P) (L,)
(H) () (F,)
(K) () (F,)
(R) () (F,)
(H) () (L,)
```

(continues on next page)

(continued from previous page)

```
(K) () (L,)
(R) () (L,)
```

or, in a condensed way:

```
cleaving rule:
(,F or L,)

exception rules:
(,F or L) (P)
(R) (,F or L)
(P) () (,F or L)
(H or K or R) () () (,F or L)
(F or L,) () (P)
(P) (F or L,)
(H or K or R) () (F or L,)
```

Those two definitions are completely equivalent for **RPG**.

#### 4.1.11.4 Example of enzymes

All available enzymes are in *Enzyme definitions*, including their **RPG**'s definition.

### 4.1.12 Deleting user-defined enzymes

All user-defined enzymes are stored in `~/rpg_user.py`. This file is automatically generated by **RPG** and written in **Python**.

Each enzyme definition starts with:

```
# User-defined enzyme <name of the enzyme>
```

and finishes with:

```
CPT_ENZ += 1
```

followed by 3 blank line.

To remove an enzyme, be sure to backup the file **before** any modifications. Then just remove the whole Python code of the enzyme, including the above-mentioned lines. Do not do any other modifications, as this code is used in **RPG** and any wrong modifications will make the software unable to run.

To remove all user-defined enzymes, just delete `~/rpg_user.py` file. It will be created again (empty) at the next launch of **RPG**.

Obviously, all deleted enzymes can not be recovered. If one wants to use them again they will need to be redefined in **RPG**, using `-a` option.

## 4.2 Enzyme definitions

All default available enzymes (*enzymes\_definition.py*) are listed bellow.

For each of them, there is the equivalent in *RPG* grammar.

In the following, nomenclature of [Schechter and Berger](#) is used. Amino acids before the cleavage site are designated as *P1*, *P2*, *P3*, etc in the N-terminal direction, and as *P1'*, *P2'*, *P3'*, etc in the C-terminal direction. For example, with cleavage site represented as 'I':

```
...P3-P2-P1-|-P1'-P2'-P3'...
```

In **RPG**, this nomenclature is represented as:

```
... (P3) (P2) (P1) (,) (P1') (P2') (P3') ...
```

## 4.2.1 Available enzymes

1: <i>Arg-C</i>	2: <i>Asp-N</i>	3: <i>BNPS-Skatole</i>
4: <i>Bromelain</i>	5: <i>Caspase 1</i>	6: <i>Caspase 2</i>
7: <i>Caspase 3</i>	8: <i>Caspase 4</i>	9: <i>Caspase 5</i>
10: <i>Caspase 6</i>	11: <i>Caspase 7</i>	12: <i>Caspase 8</i>
13: <i>Caspase 9</i>	14: <i>Caspase 10</i>	15: <i>Chymotrypsin high specificity</i>
16: <i>Chymotrypsin low specificity</i>	17: <i>Clostripain</i>	18: <i>CNBr</i>
19: <i>Enterokinase</i>	20: <i>Factor Xa</i>	21: <i>Ficin</i>
22: <i>Formic acid</i>	23: <i>Glu-C</i>	24: <i>Glutamyl endopeptidase</i>
25: <i>Granzyme B</i>	26: <i>Hydroxylamine</i>	27: <i>Iodosobenzoic acid</i>
28: <i>Lys-C</i>	29: <i>Lys-N</i>	30: <i>Neutrophil elastase</i>
31: <i>NTCB</i>	32: <i>Papain</i>	33: <i>Pepsin pH 1.3</i>
34: <i>Pepsin pH &gt;=2</i>	35: <i>Proline-endopeptidase</i>	36: <i>Proteinase K</i>
37: <i>Staphylococcal peptidase I</i>	38: <i>Thermolysin</i>	39: <i>Thrombin (PeptideCutter)</i>
40: <i>Thrombin SG</i>	41: <i>Tobacco etch virus protease</i>	42: <i>Trypsin</i>

### 4.2.1.1 Arg-C

Arg-C proteinase preferentially cleaves after R (*P1*)

**RPG definition:**

cleaving rule:

- (R,)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#ArgC](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#ArgC)

### 4.2.1.2 Asp-N

Asp-N Sequencing Grade preferentially cleaves before C or D (*P1'*)

**RPG definition:**

cleaving rule:

- (,C or D)

More information: <https://france.promega.com/resources/pubhub/using-endoproteinasas-asp-n-and-glu-c-to-improve-protein-character>

#### 4.2.1.3 BNPS-Skatole

BNPS-Skatole preferentially cleaves after W (*PI*)

**RPG definition:**

cleaving rule:

- (W, )

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#BNPS](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#BNPS)

#### 4.2.1.4 Bromelain

Bromelain preferentially cleaves after K, A or Y (*PI*)

**RPG definition:**

cleaving rule:

- (K or A or Y, )

More information: <https://www.sigmaaldrich.com/life-science/biochemicals/biochemical-products.html?TablePage=16410479>

#### 4.2.1.5 Caspase 1

Caspase 1 preferentially cleaves after D (*PI*) preceded by H, A or T in *P2* and preceded by F, W, Y or L in *P4*. It will not cleave if D is followed by P, E, D, Q, K or R in *PI*'.

**RPG definition:**

cleaving rule:

- (F or W or Y or L) ( ) (H or A or T) (D, )

exception rule:

- (F or W or Y or L) ( ) (H or A or T) (D, ) (P or E or D or Q or K or R)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Casp1](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Casp1)

#### 4.2.1.6 Caspase 2

Caspase 2 preferentially cleaves after D (*PI*) preceded by DVA or DEH. It will not cleave if D is followed by P, E, D, Q, K or R in *PI*'.

**RPG definition:**

cleaving rules:

- (D) (V) (A) (D, )
- (D) (E) (H) (D, )

exception rules:

- (D) (V) (A) (D, ) (P or E or D or Q or K or R)
- (D) (E) (H) (D, ) (P or E or D or Q or K or R)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Casp2](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Casp2)

#### 4.2.1.7 Caspase 3

Caspase 3 preferentially cleaves after D (*PI*) preceded by DMQ or DEV. It will not cleave if D is followed by P, E, D, Q, K or R in *PI*'.

**RPG definition:**

cleaving rules:

- (D) (M) (Q) (D, )
- (D) (E) (V) (D, )

exception rules:

- (D) (M) (Q) (D, ) (P or E or D or Q or K or R)
- (D) (E) (V) (D, ) (P or E or D or Q or K or R)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Casp3](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Casp3)

#### 4.2.1.8 Caspase 4

Caspase 4 preferentially cleaves after D (*PI*) preceded by LEV or (W/L)EH. It will not cleave if D is followed by P, E, D, Q, K or R in *PI*'.

**RPG definition:**

cleaving rules:

- (L) (E) (V) (D, )
- (W or L) (E) (H) (D, )

exception rules:

- (L) (E) (V) (D, ) (P or E or D or Q or K or R)
- (W or L) (E) (H) (D, ) (P or E or D or Q or K or R)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Casp4](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Casp4)

#### 4.2.1.9 Caspase 5

Caspase 5 preferentially cleaves after D (*PI*) preceded by (W/L)EH.

**RPG definition:**

cleaving rule:

- (W or L) (E) (H) (D, )

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Casp5](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Casp5)

#### 4.2.1.10 Caspase 6

Caspase 6 preferentially cleaves after D (*PI*) preceded by VEI or VEH. It will not cleave if D is followed by P, E, D, Q, K or R in *PI*'.

**RPG definition:**

cleaving rule:

- (V) (E) (I or H) (D,)

exception rule:

- (V) (E) (I or H) (D,) (P or E or D or Q or K or R)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Casp6](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Casp6)

#### 4.2.1.11 Caspase 7

Caspase 7 preferentially cleaves after D (*PI*) preceded by DEV. It will not cleave if D is followed by P, E, D, Q, K or R in *PI*'.

**RPG definition:**

cleaving rule:

- (D) (E) (V) (D,)

exception rule:

- (D) (E) (V) (D,) (P or E or D or Q or K or R)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Casp7](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Casp7)

#### 4.2.1.12 Caspase 8

Caspase 8 preferentially cleaves after D (*PI*) preceded by (I/L)ET. It will not cleave if D is followed by P, E, D, Q, K or R in *PI*'.

**RPG definition:**

cleaving rule:

- (I or L) (E) (T) (D,)

exception rule:

- (I or L) (E) (T) (D,) (P or E or D or Q or K or R)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Casp8](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Casp8)

#### 4.2.1.13 Caspase 9

Caspase 9 preferentially cleaves after D (*PI*) preceded by LEH.

**RPG definition:**

cleaving rule:

- (L) (E) (H) (D,)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Casp9](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Casp9)

#### 4.2.1.14 Caspase 10

Caspase 10 preferentially cleaves after D (*PI*) preceded by IEA.

**RPG definition:**

cleaving rule:

- (I) (E) (A) (D,)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Casp10](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Casp10)

#### 4.2.1.15 Chymotrypsin high specificity

This chymotrypsin preferentially cleaves after F, Y or W (*PI*) if not followed by P in *PI'*. It will not cleave after W followed by M in *PI'*.

**RPG definition:**

cleaving rule:

- (F or Y or W,)

exception rules:

- (F or Y or W,)(P)
- (W,)(M)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Chym](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Chym)

#### 4.2.1.16 Chymotrypsin low specificity

This chymotrypsin preferentially cleaves after F, L, Y, W, M or H (*PI*) if not followed by P in *PI'*. It will not cleave after W followed by M in *PI'*. It will not cleave after M followed by Y in *PI'*. It will not cleave after H followed by D/M/W in *PI'*.

**RPG definition:**

cleaving rule:

- (F or L or Y or W or M or H,)

exception rules:

- (F or L or Y or W or M or H,)(P)
- (W,)(M)
- (M,)(Y)
- (H,)(D or M or W)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Chym](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Chym)

#### 4.2.1.17 Clostripain

Clostripain (Clostridiopeptidase B) preferentially cleaves after R (*PI*).

**RPG definition:**

cleaving rule:

- (R,)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Clost](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Clost)



#### 4.2.1.18 CNBr

CNBr preferentially cleaves after M (*P1*).

**RPG definition:**

cleaving rule:

- (M, )

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#CNBr](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#CNBr)

#### 4.2.1.19 Enterokinase

Enterokinase preferentially cleaves after K (*P1*) preceded by D/E in *P2*, *P3*, *P4* and *P5*.

**RPG definition:**

cleaving rule:

- (D or E) (D or E) (D or E) (D or E) (K, )

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Enter](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Enter)

#### 4.2.1.20 Factor Xa

Factor Xa preferentially cleaves after R (*P1*) preceded by G in *P2*, D/E in *P3* and A/F/I/L/V/W/G/T in *P4*.

**RPG definition:**

cleaving rule:

- (A or F or I or L or V or W or G or T) (D or E) (G) (R, )

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Xa](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Xa)

#### 4.2.1.21 Ficin

Ficin preferentially cleaves after G, S, E or Y (*P1*) preceded by A, V, I, L, F, Y or W in *P2*.

**RPG definition:**

cleaving rule:

- (A or V or I or L or F or Y or W) (G or S or E or Y, )

More information: <https://www.sigmaaldrich.com/life-science/biochemicals/biochemical-products.html?TablePage=16410578>

#### 4.2.1.22 Formic acid

Formic acid preferentially cleaves after D (*P1*).

**RPG definition:**

cleaving rule:

- (D, )

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#HCOOH](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#HCOOH)

#### 4.2.1.23 Glu-C

Glu-C Sequencing Grade preferentially cleaves after D or E (*PI*).

**RPG definition:**

cleaving rule:

- (D or E, )

More information: <https://france.promega.com/resources/pubhub/using-endoproteinasas-asp-n-and-glu-c-to-improve-protein-character>

#### 4.2.1.24 Glutamyl endopeptidase

Glutamyl endopeptidase preferentially cleaves after E (*PI*).

**RPG definition:**

cleaving rule:

- (E, )

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Glu](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Glu)

#### 4.2.1.25 Granzyme B

Granzyme B preferentially cleaves after D (*PI*) preceded by IEP.

**RPG definition:**

cleaving rule:

- (I) (E) (P) (D, )

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#GranB](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#GranB)

#### 4.2.1.26 Hydroxylamine

Hydroxylamine (NH<sub>2</sub>OH) preferentially cleaves after N (*PI*) followed by G in *PI*'.

**RPG definition:**

cleaving rule:

- (N, ) (G)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Hydro](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Hydro)

#### 4.2.1.27 Iodosobenzoic acid

Iodosobenzoic acid preferentially cleaves after W (*PI*).

**RPG definition:**

cleaving rule:

- (W, )

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Iodo](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Iodo)

#### 4.2.1.28 Lys-C

LysC Lysyl endopeptidase (Achromobacter proteinase I) preferentially cleaves after K (*PI*).

**RPG definition:**

cleaving rule:

- (K, )

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#LysC](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#LysC)

#### 4.2.1.29 Lys-N

LysN Peptidyl-Lys metalloendopeptidase preferentially cleaves before K (*PI'*).

**RPG definition:**

cleaving rule:

- (, K)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#LysN](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#LysN)

#### 4.2.1.30 Neutrophil elastase

Neutrophil elastase preferentially cleaves after A or V (*PI*).

**RPG definition:**

cleaving rule:

- (A or V, )

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Elast](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Elast)

#### 4.2.1.31 NTCB

NTCB +Ni (2-nitro-5-thiocyanobenzoic acid) preferentially cleaves before C (*PI'*).

**RPG definition:**

cleaving rule:

- (, C)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#NTCB](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#NTCB)

#### 4.2.1.32 Papain

Papain preferentially cleaves after R or K (*PI*) preceded by A, V, I, L, F, Y or W in *P2*. It will not cleave if followed by V in *PI'*.

**RPG definition:**

cleaving rule:

- (A or V or I or L or F or Y or W) (R or K, )

exception rule:

- (A or V or I or L or F or Y or W) (R or K, ) (V)

More information: <https://www.sigmaaldrich.com/life-science/biochemicals/biochemical-products.html?TablePage=16410606>

### 4.2.1.33 Pepsin pH 1.3

This pepsin preferentially cleaves around F or L (*PI* or *PI'*). It will not cleave before F or L in *PI'* followed by P in *P2'*. It will not cleave before F or L in *PI'* preceded by R in *PI* or P in *P2* or H/K/R in *P3*. It will not cleave after F or L in *PI* followed by P in *P2'*. It will not cleave after F or L in *PI* preceded by P in *P2* or H/K/R in *P3*.

#### RPG definition:

cleaving rule:

- (,F or L,)

exception rules:

- (,F or L) (P)
- (R) (,F or L)
- (P) () (,F or L)
- (H or K or R) () () (,F or L)
- (F or L,) () (P)
- (P) (F or L,)
- (H or K or R) () (F or L,)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Peps](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Peps)

### 4.2.1.34 Pepsin pH >=2

This pepsin preferentially cleaves around F, L, W or Y (*PI* or *PI'*). It will not cleave before F, L, W or Y in *PI'* followed by P in *P2'*. It will not cleave before F, L, W or Y in *PI'* preceded by R in *PI* or P in *P2* or H/K/R in *P3*. It will not cleave after F, L, W or Y IN *PI* followed by P in *P2'*. It will not cleave after F, L, W or Y in *PI* preceded by P in *P2* or H/K/R in *P3*.

#### RPG definition:

cleaving rule:

- (,F or L or W or Y,)

exception rules:

- (,F or L or W or Y) (P)
- (R) (,F or L or W or Y)
- (P) () (,F or L or W or Y)
- (H or K or R) () () (,F or L or W or Y)
- (F or L or W or Y,) () (P)
- (P) (F or L or W or Y,)
- (H or K or R) () (F or L or W or Y,)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Peps](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Peps)

#### 4.2.1.35 Proline-endopeptidase

Proline-endopeptidase preferentially cleaves after P (*PI*) preceded by H, K or R in *P2* but will not cleaves if followed by P in *PI'*.

**RPG definition:**

cleaving rule:

- (H or K or R) (P, )

exception rule:

- (H or K or R) (P, ) (P)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Pro](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Pro)

#### 4.2.1.36 Proteinase K

Proteinase K preferentially cleaves after F, W, Y, T, E, A, V, L or I (*PI*). The predominant site of cleavage is the peptide bond adjacent to the carboxyl group of aliphatic and aromatic amino acids.

**RPG definition:**

cleaving rule:

- (F or W or Y or T or E or A or V or L or I, )

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#ProtK](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#ProtK)

#### 4.2.1.37 Staphylococcal peptidase I

Staphylococcal peptidase I preferentially cleaves after E (*PI*). It will not cleave after E in *PI* preceded by E in *P2*, but cleaves after E in *PI* followed by E in *PI'*.

**RPG definition:**

cleaving rule:

- (E, )

exception rule:

- (E) (E, )

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Staph](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Staph)

#### 4.2.1.38 Thermolysin

Thermolysin preferentially cleaves before A,F,I,L,M or V (*PI'*) when not followed by P in *P2'* nor preceded by D or E in *PI*.

**RPG definition:**

cleaving rule:

- (,A or F or I or L or M or V)

exception rules:

- (,A or F or I or L or M or V) (P)
- (D or E) (,A or F or I or L or M or V)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Therm](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Therm)

### 4.2.1.39 Thrombin (PeptideCutter)

This thrombin preferentially cleaves after R (*PI*). Optimum cleavage is when R is preceded and followed by G (*P2* and *PI'*). Cleavage also occurs when R is preceded by P in *P2* and A, F, I, L, V, W, G or T in *P3* and *P4*. It will not cleave after R followed by D/E in *PI'* or *P2'*.

It is not strictly coherent with the definition in PeptideCutter, as in this software there are differences between definition, summary and behavior of this enzyme.

#### RPG definition:

cleaving rules:

- (G) (R, ) (G)
- (A or F or I or L or V or W or G or T) (A or F or I or L or V or W or G or T) (P) (R, )

exception rules:

- (A or F or I or L or V or W or G or T) (A or F or I or L or V or W or G or T) (P) (R, ) (D or E)
- (A or F or I or L or V or W or G or T) (A or F or I or L or V or W or G or T) (P) (R, ) ( ) (D or E)

**Warning:** the following combined exception (A or F or I or L or V or W or G or T) (A or F or I or L or V or W or G or T) (P) (R, ) (D or E) (D or E) cannot be used instead, as it will cleave on [...] (R, ) (D or E).

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Throm](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Throm) <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3288055/>

### 4.2.1.40 Thrombin SG

This thrombin (Sequencing Grade) preferentially cleaves after R (*PI*) preceded by P in *P2*, V in *P3* and L in *P4* and followed by G in *PI'* and S in *P2'*.

This thrombin is defined in several kits (see below).

#### RPG definition:

cleaving rule:

- (L) (V) (P) (R, ) (G) (S)

More information: see thrombin cleavage kits of [Abcam](#), [BioVision](#), [Merck](#) or [Novagen](#).

### 4.2.1.41 Tobacco etch virus protease

Tobacco etch virus protease (TEV) preferentially cleaves after Q (*PI*) when followed by G or S in *PI'* and preceded by Y in *P3* and E in *P6*.

#### RPG definition:

cleaving rule:

- (E) ( ) ( ) (Y) ( ) (Q, ) (G or S)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#TEV](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#TEV)

#### 4.2.1.42 Trypsin

Trypsin preferentially cleaves after K or R (*PI*). It will not cleave after K followed by P in *PI*' except if W in *P2*. It will not cleave after R followed by P in *PI*' except if M in *P2*. It will not cleave CKD, DKD, CKH, CKY, CRK, RRH nor RRR.

##### RPG definition:

cleaving rules:

- (K or R, )
- (W) (K, ) (P)
- (M) (R, ) (P)

exception rules:

- (K or R, ) (P)
- (C) (K, ) (D)
- (D) (K, ) (D)
- (C) (K, ) (H)
- (C) (K, ) (Y)
- (C) (R, ) (K)
- (R) (R, ) (H)
- (R) (R, ) (R)

More information: [https://web.expasy.org/peptide\\_cutter/peptidecutter\\_enzymes.html#Tryps](https://web.expasy.org/peptide_cutter/peptidecutter_enzymes.html#Tryps)

## 4.3 rpg

### 4.3.1 rpg package

#### 4.3.1.1 Submodules

#### 4.3.1.2 rpg.RapidPeptidesGenerator module

Main file of RPG software, handle input/output and launch necessary functions

```
rpg.RapidPeptidesGenerator.ALL_ENZYMES = [Id: 1 Name: Arg-C Ratio Miscleavage: 0.00% Ru.
All available enzymes in RPG.
```

```
rpg.RapidPeptidesGenerator.create_enzymes_to_use(enzymes, miscleavage)
Create the list of chosen Enzyme to use. Each enzyme can be associated to a miscleavage value.
```

##### Parameters

- **enzymes** (*list(int)*) – enzymes ids chosen by user
- **miscleavage** (*list(float)*) – associated miscleavage values

**Returns** list of enzyme's id with associated miscleavage values

**Return type** list(int)

`rpg.RapidPeptidesGenerator.get_enzymes_to_use(mode, id_enz_selected, miscleavage)`

Get the list of chosen *Enzyme* to use. Each enzyme (and associated miscleavage value) are inputed by user. If there is a problem, user is interrogated again.

**Parameters**

- **mode** (*str*) – Digestion mode. If 'concurrent', no miscleavage values are used
- **enzymes** (*list(int)*) – enzyme's ids chosen by user
- **miscleavage** (*list(float)*) – associated miscleavage values

**Returns** list of enzyme's id with associated miscleavage values

**Return type** list(int)

`rpg.RapidPeptidesGenerator.list_enzyme()`

Print all available enzymes

`rpg.RapidPeptidesGenerator.main()`

Launcher of RapidPeptidesGenerator

`rpg.RapidPeptidesGenerator.restricted_enzyme_id(enz_id)`

Restrict input enzyme id to an int corresponding to an enzyme.

**Parameters** **mc\_val** (*int*) – value to test

**Returns** the inputed enzyme id

**Return type** int

**Raises**

- **custom ValueError** – if id does not correspond to any enzyme
- **custom TypeError** – if value is not an int

`rpg.RapidPeptidesGenerator.restricted_float(mc_val)`

Restricts input miscleavage value to a float between 0 and 100.

**Parameters** **mc\_val** (*float*) – value to test

**Returns** the inputed value if correct

**Return type** float

**Raises**

- **custom ValueError** – if value is not between 0 and 100
- **custom TypeError** – if value is not a float

### 4.3.1.3 rpg.context module

### 4.3.1.4 rpg.core module

Contains generic functions and global variables used by RPG

`rpg.core.AA_MASS_AVERAGE = {'#': 0.0, '*': 0.0, 'A': 71.0788, 'B': 0.0, 'C': 103.1388, 'D': 115.0914, 'E': 133.075, 'F': 147.1566, 'G': 146.1554, 'H': 147.1534, 'I': 150.1539, 'K': 188.1562, 'L': 132.1509, 'M': 148.1494, 'N': 132.1477, 'O': 146.1471, 'P': 150.1476, 'Q': 146.1471, 'R': 172.1562, 'S': 146.1534, 'T': 147.1534, 'V': 117.1509, 'W': 173.1534, 'Y': 163.1534, 'Z': 146.1534}`

Mass of all amino acids.



`rpg.core.AA_PKI_IPC` = {'C': 8.297, 'Cterm': 2.383, 'D': 3.887, 'E': 4.317, 'H': 6.018, 'K': 10.8} (from IPC\_peptide. See <http://isoelectric.org/theory.html> for details).

**Type** pKa of important amino acid to compute pI (from IPC\_peptide. See <http://isoelectric.org/theory.html> for details).

`rpg.core.AA_PKI_IPC_2` = {'C': 9.454, 'Cterm': 2.977, 'D': 3.969, 'E': 4.507, 'H': 6.439, 'K': 10.8} (from IPC\_peptide2. See <http://www.ipc2-isoelectric-point.org/> for details).

**Type** pKa of important amino acid to compute pI (from IPC\_peptide2. See <http://www.ipc2-isoelectric-point.org/> for details).

`rpg.core.AA_PKI_S` = {'C': 8.3, 'Cterm': 3.1, 'D': 4.1, 'E': 4.1, 'H': 6.0, 'K': 10.8, 'Nterm': 10.8} (from Stryer).  
pKa of important amino acid to compute pI (from Stryer).

`rpg.core.AMINOACIDS` = ['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'Y']  
All character accepted in a peptide.

`rpg.core.WATER_MASS` = 18.01528  
Mass of a water molecule.

`rpg.core.get_header` (*fmt='fasta'*)  
Construct a header for output file in *csv* or *tsv*.

**Parameters** *fmt* (*str*) – format of header

**Returns** formatted header

**Return type** *str* or *None*

Informations on the header are:

Original\_header No\_pep Enzyme Cleav\_pos Pep\_size Pep\_mass pI Seq

No header for *fasta* or other format.

`rpg.core.handle_errors` (*message=""*, *err=1*, *error\_type=""*)  
Custom handling of errors and warnings.

**Parameters**

- **message** (*str*) – error message to print
- **err** (*int*) – Type of message
- **error\_type** (*str*) – header of error to print

*Type of message* is:

- **0** for critical error (exit)
- **1** for warning (no exit, default)
- **2** for print in *stderr*

`rpg.core.next_read` (*file*, *offset\_start*, *offset\_end*)

**Return each sequence between offsets range of a file** as a tuple (header, seq) using a generator. Can be *fasta* or *fastq*, *gzipped* or not.

**Parameters**

- **file** (*str*) – *fasta/fastq* file to read
- **offset\_start** (*int*) – offset in the file from where to read
- **offset\_end** (*int*) – offset in the file until where to read

`rpg.core.output_results` (*output\_file, all\_seq\_digested, fmt, quiet, verbose*)

Output results of digestion in file and optionally in *stdout*.

### Parameters

- **output\_file** (*str*) – the file where to print results, if exist
- **all\_seq\_digested** (`list(list(ResultOneDigestion))`) – results of digestions
- **fmt** (*str*) – output format (*csv, tsv* or *fasta*)
- **quiet** (*bool*) – quiet mode, no *stdout* message
- **verbose** (*int*) – verbosity level

### 4.3.1.5 rpg.digest module

Contains class and function needed to perform a digestion

**class** `rpg.digest.ResultOneDigestion` (*enzyme\_name, peptides=None, nb\_cleavage=0, pos\_misceavage=None*)

Bases: `object`

Result of the digestion of one sequence by one enzyme.

### Parameters

- **enzyme\_name** (*str*) – name of the enzyme used
- **peptides** (`list(Peptide)`) – all resulting peptides after digestion
- **nb\_cleavage** (*int*) – number of cleavage that occurs
- **pos\_misceavage** (`list(int)`) – position of miscleavage that occurs

**add\_misceavage** (*new\_pos\_misceavage*)

Add a miscleavage to `self.pos_misceavage`.

**Parameters** *new\_pos\_misceavage* (*int*) – position of miscleavage

**add\_peptide** (*pep*)

Add a peptide to `self.peptides`.

**Parameters** *pep* (*Peptide*) – peptide to add

**get\_cleavage\_pos** ()

Get positions of cleavage as a string.

**Returns** positions of cleavage

**Return type** `str`

**get\_misceavage\_pos** ()

Get positions of miscleavage as a string.

**Returns** positions of miscleavage

**Return type** `str`

**get\_more\_info** ()

Return informations and statistics about this digestion, *i.e.* number of (mis)-cleavages and positions, mis-cleavage ratio, size of the smallest peptide and first and last peptide.

**Returns** informations and statistics ready to be printed

**Return type** `str`

**get\_nb\_misceavage ()**

Get the number of miscleavages that occurs on this digestion.

**Returns** number of miscleavage

**Return type** int

**get\_ratio\_misceavage ()**

Get ratio of miscleavage.

**Returns** ratio of miscleavage

**Return type** float

**get\_smallest\_peptide ()**

Get the (first) smallest peptide of `self.peptides`.

**Returns** the smallest peptide

**Return type** *Peptide*

**inc\_nb\_cleavage ()**

Increase `self.nb_cleavage` by 1.

**merge (other)**

Fuse two *ResultOneDigestion* by adding to `self` the peptides of `other` and changing their *Enzyme*. It also update `self.nb_cleavage` and `self.pos_misceavage`.

**Parameters** **other** (*ResultOneDigestion*) – object to fuse with *self*

**pop\_peptides ()**

Empty `self.peptides` and returns all peptides.

**Returns** all the peptides

**Return type** list(*Peptide*)

`rpg.digest.concurrent_digest (seq, enz, aa_pka)`

Concurrently digest a sequence with all Enzymes **at the same time**.

**Parameters**

- **seq** (*Sequence*) – sequence to digest
- **enz** (list(*Enzyme*)) – enzymes to digest with
- **aa\_pka** (*str*) – pKa values (IPC / Stryer)

**Returns** result of the digestion

**Return type** list(*ResultOneDigestion*)

`rpg.digest.digest_from_input (input_data, input_type, enz, mode, aa_pka, nb_proc=1)`

Digest all sequences of input data according to selected enzymes and mode. Can be done in parallel using `nb_proc` argument.

**Parameters**

- **input\_data** (*str*) – either a sequence or the path of a file of sequence (fasta/fastq, gzipped or not)
- **input\_type** (*str*) – either ‘sequence’ or ‘file’
- **enz** (list(*Enzyme*)) – enzymes to digest with
- **mode** (*str*) – digestion mode (concurrent / sequential)
- **aa\_pka** (*str*) – pKa values (IPC / Stryer)

- **nb\_proc** (*int* (default: 1)) – number of process to run in parallel

**Returns** result of digestions

**Return type** list(list(*ResultOneDigestion*))

`rpg.digest.digest_one_sequence` (*seq, enz, mode, aa\_pka*)

Launch a digest procedure on one sequence.

**Parameters**

- **sequence** (*Sequence*) – sequence to digest
- **enz** (list(*Enzyme*)) – enzymes to digest with
- **mode** (*str*) – digestion mode (concurrent / sequential)
- **aa\_pka** (*str*) – pKa values (IPC / Stryer)

**Returns** result of the digestion

**Return type** list(*ResultOneDigestion*)

`rpg.digest.digest_part` (*offset\_start, offset\_end, file, enz, mode, aa\_pka*)

**Main parallelized function that digest each sequence of a file** in an offset range.

**Parameters**

- **offset\_start** (*int*) – where to start taking sequences in the file
- **offset\_end** (*int*) – where to stop taking sequences in the file
- **file** (*string*) – the filename of the file where to take sequences from
- **enz** (list(*Enzyme*)) – enzymes to digest with
- **mode** (*str*) – digestion mode (concurrent / sequential)
- **aa\_pka** (*str*) – pKa values (IPC / Stryer)

`rpg.digest.one_digest` (*pep, enz, aa\_pka*)

Digest a peptide with an enzyme.

**Parameters**

- **pep** (*Peptide*) – peptide to digest
- **enz** (*Enzyme*) – enzyme to digest with
- **aa\_pka** (*str*) – pKa values (IPC / Stryer)

**Returns** result of the digestion

**Return type** *ResultOneDigestion*

`rpg.digest.sequential_digest` (*seq, enz, aa\_pka*)

Sequentially digest a sequence with all Enzymes, **one by one**.

**Parameters**

- **seq** (*Sequence*) – sequence to digest
- **enz** (list(*Enzyme*)) – enzymes to digest with
- **aa\_pka** (*str*) – pKa values (IPC / Stryer)

**Returns** result of the digestion

**Return type** `list(ResultOneDigestion)`

#### 4.3.1.6 rpg.enzyme module

Contains class and functions related to enzymes definition and use

**class** `rpg.enzyme.Enzyme` (*id\_*, *name*, *rules*, *ratio\_miscleavage=0*)  
 Bases: `object`

Definition of an cleaving enzyme containing specific rules.

##### Parameters

- **id** (*int*) – id of the enzyme
- **name** (*str*) – name of the enzyme
- **rules** (`list(Rule)`) – cleaving rules
- **ratio\_miscleavage** (*float*) – miscleavage ratio

**write\_enzyme\_in\_user\_file** (*enz\_file='/home/docs/rpg\_user.py'*)

Write enzyme to user's enzyme file as a Python function.

**Parameters** *enz\_file* (*str*) – location of user file (default: `~/rpg_user.py`)

`rpg.enzyme.check_enzyme_name` (*name\_new\_enz*, *all\_name\_enz*)

Validate the name of a new enzyme.

##### Parameters

- **name\_new\_enz** (*str*) – name of the new enzyme
- **all\_name\_enz** (`list(str)`) – names of already created enzymes

**Returns** True if name is correct

**Return type** `bool`

Enzyme name should not contains whitespace character (' ', \t, \n, \r, \f, \v), be empty or be already used.

`rpg.enzyme.user_creation_enzyme` (*all\_enzymes*)

Text-mod form to input a new enzyme.

**Warning:** Not tested

**Warning:** It could be a problem to immediately use the new enzyme (see in-code warning)

#### 4.3.1.7 rpg.enzymes\_definition module

Definition of default enzymes in RPG

#### 4.3.1.8 rpg.rule module

Contains class and functions related to definition of cleaving rules

**class** `rpg.rule.Rule` (*index, amino\_acid, cleavage, pos*)

Bases: `object`

Definition of a principal rule defining where a cleavage occurs.

**Parameters**

- **index** (*signed int*) – position where to look for a specific amino acid
- **amino\_acid** (*char*) – amino acid to look for
- **cleavage** (*bool*) – cleavage or not at this position on this amino acid
- **pos** (*int*) – cleavage before (0) of after (1) amino acid. -1 for unused value

**Variables** **rules** (`list(Rule)`) – additional sub-rules of this rule

**contains** (*other*)

Test if another rule is contains in sub-rules **without taking into account sub-rules nor ‘cleavage’** and return the sub-rule if founded.

**Parameters** **other** (*Rule*) – rule to compare with

**Returns** the sub-rule is founded, *None* otherwise

**Return type** *Rule*

**contains\_any\_level** (*other, ret=False*)

Test if another rule is contains within **without taking into account sub-rules nor ‘cleavage’** no matter what level.

**Parameters**

- **other** (*Rule*) – rule to compare with
- **ret** (*bool*) – previous *ret* value (default: *False*)

**Returns** *True* if rule is included, *False* otherwise

**Return type** `bool`

**equ** (*other*)

Test equality with another rule **without taking into account sub-rules nor ‘cleavage’**.

**Parameters** **other** (*Rule*) – rule to compare with

**Returns** *True* if rules are equal, *False* otherwise

**Return type** `bool`

**Note** use ‘==’ for complete equality including sub-rules.

**format\_a\_rule** (*prev\_name, prev\_com*)

Format the rule in Python.

**Parameters**

- **prev\_name** (*str*) – name of the upper rule
- **prev\_com** (*str*) – comment of the upper rule

**Returns** this rule in Python

**Return type** `str`

**format\_rule** (*prev\_name=”, prev\_com=’ # ’*)

Format the whole rule, including sub-rules, in Python, ready to be written.

**Parameters**

- **prev\_name** (*str*) – name of the upper rule
- **prev\_com** (*str*) – comment of the upper rule

**Returns** the whole rule in Python

**Return type** str

**get\_all\_headers** ()

Format header of the rule and sub-rules in Python.

**Returns** the complete header in Python

**Return type** str

**get\_header** ()

Format header of the rule in Python.

**Returns** header of this rule in Python

**Return type** str

`rpg.rule.add_missing_rule(main_rule, dict_of_rule, rule_to_add)`

Add a rule of an exception in a main rule

**Parameters**

- **main\_rule** (*Rule*) – the main rule to add in
- **dict\_of\_rule** (*Dict of pos/val*) – Raw rules of an exception
- **rule\_to\_add** (*Tuple (pos/val)*) – The rule to add

`rpg.rule.add_rule(rules_list, a_rule)`

Add (recursively) a rule to a list of rules.

**Parameters**

- **rules\_list** (*list(Rule)*) – the list of rules where to add
- **a\_rule** (*Rule*) – the rule to add

**Warning:** Modify *rules\_list* :python:

`rpg.rule.check_rule(exprule)`

Check if a rule is properly inputed.

**Parameters** **exprule** (*str*) – the raw expression of a rule

**Returns** *exprule* if it is correct, empty char otherwise

**Return type** str

`rpg.rule.create_rules(all_rules)`

Create proper rules for an enzyme from raw rules.

**Parameters** **all\_rules** (*list(Rule)*) – rules corresponding to the enzyme

**Returns** rules ready to populate an *Enzyme*

**Return type** *list(Rule)*

This function handle ‘ or ‘ keywords, multiple parenthesis, sort the simples rules, create sub-rules, etc. The output is ready to be used to create an *Enzyme*.

`rpg.rule.find_missing_rule` (*main\_rule*, *dict\_of\_rule*, *depth=0*)

Find all missing rules of an exception in a main rule

**Parameters**

- **main\_rule** (*Rule*) – the main rule to search in
- **dict\_of\_rule** (*Dict of pos/val*) – Raw rules of an exception
- **depth** (*int*) – Depth of missing rule (default: 0)

**Returns** Missing rules and their positions and depth (key)

**Return type** defaultdict(list)

`rpg.rule.find_reachable_pos` (*main\_rule*, *dict\_of\_rule*)

Find all positions reachable on a main rule according to several rules.

**param main\_rule** the main rule to search in

**type main\_rule** *Rule*

**param dict\_of\_rule** Raw rules of an exception

**type dict\_of\_rule** Dict of pos/val

**return** Reachable *Rule* and their depth (key)

**rtype** dict()

`rpg.rule.handle_rule` (*seq*, *pos*, *a\_rule*, *cleavage*)

Recursive handling of a *Rule* determining if a sequence must be cleavaged at a given position according to the rule.

**Parameters**

- **seq** (*str*) – sequence to test
- **pos** (*int*) – position on the sequence
- **a\_rule** (*Rule*) – the rule
- **cleavage** (*bool*) – boolean telling if it must be cleavaged or not

**Returns** *True* if sequence must be cleavaged

**Return type** bool or None

`rpg.rule.split_complex_rule` (*a\_rule*)

Split a complex rules containing ‘ or ‘ into simpler rules.

**Parameters** **a\_rule** (*str*) – the rule to split

**Returns** the simple rules

**Return type** list(str)

### 4.3.1.9 rpg.sequence module

Contains classes and function related to sequences

**class** `rpg.sequence.Peptide` (*header*, *sequence*, *enzyme\_name*, *aa\_pka*, *nb\_peptide=0*, *position=0*)

Bases: object

Definition of a peptide, containing the header of its original sequence, an amino acid sequence, the name of the enzyme used to produce it and more informations.



**Parameters**

- **header** (*str*) – header of the peptide
- **sequence** (*str*) – sequence in amino acids
- **enzyme\_name** (*str*) – name of the enzyme used
- **aa\_pkas** (*str*) – pKa values (IPC / Stryer)
- **nb\_peptide** (*int*) – number of this peptide (default: 0)
- **position** (*int*) – position of cleavage on the original sequence (default: 0)

**Variables**

- **size** (*int*) – size of the peptide
- **mass** (*float*) – mass of the peptide
- **p\_i** (*float*) – pI of the peptide

**get\_isoelectric\_point** ()

Compute isoelectric point (pI) of the peptide using binary search.

**Returns** computed pI

**Return type** float

**Note** This function used AA\_PKAs

**class** `rpg.sequence.Sequence` (*header, sequence*)

Bases: `object`

Definition of an amino acid sequence to digest.

**Parameters**

- **header** (*str*) – header of the sequence
- **sequence** (*str*) – sequence itself

`rpg.sequence.check_sequence` (*seq*)

Validate an input sequence. Each amino acid should be in `AMINOACIDS`.

**Parameters** **seq** (*str*) – the sequence to check

**Returns** Sequence in UPPERCASE

**Return type** `str`

**4.3.1.10 Module contents**

Contains everything related to RPG software

**4.4 CHANGELOG**

- **1.2.4** Remove ‘deprecated’ for IPC v1
- **1.2.3** Add IPC2 pKa values (thanks Lukasz Kozlowski, see <https://doi.org/10.1093/nar/gkab295>). Add RPG’s publication on documentation
- **1.2.2** Correct a major bug arising when a new enzyme is define with at least 3 amino acids while the first one being the cleaving site

- **1.2.1** Add functional tests
- **1.2.0** Input files can be gzipped
  - Input file can be processed in parallel (-c options for the number of processes to use)
  - Remove Python 3.5 compatibility
- **1.1.0** Modify input. Now, option -i only takes files. Use option -s to input sequence
- **1.0.9** Correct a bug of random dict in the creation of new enzyme
  - Modifying auto repr of a rule and argument name
  - Fixing typo
- **1.0.8** Adding doc for -p option
  - Fixing typo
- **1.0.7** Adding choice for pKa values (option -p)
  - Fixing alphabetic order for enzymes
- **1.0.6** No default output file, only stdout
  - Fixing -e and -m behavior
- **1.0.5** Fix version date inside RPG
- **1.0.4** Fix version number inside RPG
- **1.0.3** Support of Python 3.7
  - Does not support Python 3.4
- **1.0.2** Correct minor but funny typo
- **1.0.1** Minor change in enzymes definition in user guide
- **1.0.0** Correct a bug when to rules were applying at the same time
  - Update user-guide
  - Beta to stable
- **0.7.0** Add the last 7 enzymes
  - Correct block-code in user-guide
- **0.6.1** Add 7 enzymes
  - Correct some typo
- **0.6.0** Finish docs
  - Minor change in user-define enzymes
  - Alpha to Beta version
- **0.5.4** Bugfix: protect new enzyme name when a new enzyme is directly inputted
- **0.5.3** Writing Doc
- **0.5.2** Incorporating tests, rtd and Gitlab

**r**

rpg, 43  
rpg.core, 34  
rpg.digest, 36  
rpg.enzyme, 39  
rpg.enzymes\_definition, 39  
rpg.RapidPeptidesGenerator, 33  
rpg.rule, 39  
rpg.sequence, 42



**A**

AA\_MASS\_AVERAGE (in module *rpg.core*), 34  
 AA\_PKA\_IPC (in module *rpg.core*), 34  
 AA\_PKA\_IPC\_2 (in module *rpg.core*), 35  
 AA\_PKA\_S (in module *rpg.core*), 35  
 add\_misceavage() (rpg.digest.ResultOneDigestion method), 36  
 add\_missing\_rule() (in module *rpg.rule*), 41  
 add\_peptide() (rpg.digest.ResultOneDigestion method), 36  
 add\_rule() (in module *rpg.rule*), 41  
 ALL\_ENZYMES (in module *rpg.RapidPeptidesGenerator*), 33  
 AMINOACIDS (in module *rpg.core*), 35

**C**

check\_enzyme\_name() (in module *rpg.enzyme*), 39  
 check\_rule() (in module *rpg.rule*), 41  
 check\_sequence() (in module *rpg.sequence*), 43  
 concurrent\_digest() (in module *rpg.digest*), 37  
 contains() (rpg.rule.Rule method), 40  
 contains\_any\_level() (rpg.rule.Rule method), 40  
 create\_enzymes\_to\_use() (in module *rpg.RapidPeptidesGenerator*), 33  
 create\_rules() (in module *rpg.rule*), 41

**D**

digest\_from\_input() (in module *rpg.digest*), 37  
 digest\_one\_sequence() (in module *rpg.digest*), 38  
 digest\_part() (in module *rpg.digest*), 38

**E**

Enzyme (class in *rpg.enzyme*), 39  
 equ() (rpg.rule.Rule method), 40

**F**

find\_missing\_rule() (in module *rpg.rule*), 41

find\_reachable\_pos() (in module *rpg.rule*), 42  
 format\_a\_rule() (rpg.rule.Rule method), 40  
 format\_rule() (rpg.rule.Rule method), 40

**G**

get\_all\_headers() (rpg.rule.Rule method), 41  
 get\_cleavage\_pos() (rpg.digest.ResultOneDigestion method), 36  
 get\_enzymes\_to\_use() (in module *rpg.RapidPeptidesGenerator*), 34  
 get\_header() (in module *rpg.core*), 35  
 get\_header() (rpg.rule.Rule method), 41  
 get\_isoelectric\_point() (rpg.sequence.Peptide method), 43  
 get\_misceavage\_pos() (rpg.digest.ResultOneDigestion method), 36  
 get\_more\_info() (rpg.digest.ResultOneDigestion method), 36  
 get\_nb\_misceavage() (rpg.digest.ResultOneDigestion method), 36  
 get\_ratio\_misceavage() (rpg.digest.ResultOneDigestion method), 37  
 get\_smallest\_peptide() (rpg.digest.ResultOneDigestion method), 37

**H**

handle\_errors() (in module *rpg.core*), 35  
 handle\_rule() (in module *rpg.rule*), 42

**I**

inc\_nb\_cleavage() (rpg.digest.ResultOneDigestion method), 37

## L

`list_enzyme()` (in module `rpg.RapidPeptidesGenerator`), 34

## M

`main()` (in module `rpg.RapidPeptidesGenerator`), 34

`merge()` (`rpg.digest.ResultOneDigestion` method), 37

## N

`next_read()` (in module `rpg.core`), 35

## O

`one_digest()` (in module `rpg.digest`), 38

`output_results()` (in module `rpg.core`), 35

## P

`Peptide` (class in `rpg.sequence`), 42

`pop_peptides()` (`rpg.digest.ResultOneDigestion` method), 37

## R

`restricted_enzyme_id()` (in module `rpg.RapidPeptidesGenerator`), 34

`restricted_float()` (in module `rpg.RapidPeptidesGenerator`), 34

`ResultOneDigestion` (class in `rpg.digest`), 36

`rpg` (module), 43

`rpg.core` (module), 34

`rpg.digest` (module), 36

`rpg.enzyme` (module), 39

`rpg.enzymes_definition` (module), 39

`rpg.RapidPeptidesGenerator` (module), 33

`rpg.rule` (module), 39

`rpg.sequence` (module), 42

`Rule` (class in `rpg.rule`), 39

## S

`Sequence` (class in `rpg.sequence`), 43

`sequential_digest()` (in module `rpg.digest`), 38

`split_complex_rule()` (in module `rpg.rule`), 42

## U

`user_creation_enzyme()` (in module `rpg.enzyme`), 39

## W

`WATER_MASS` (in module `rpg.core`), 35

`write_enzyme_in_user_file()` (`rpg.enzyme.Enzyme` method), 39