propy3

Jul 27, 2022

User Guide

CHAPTER 1

Installation

```
pip install propy3
```

CHAPTER 2

Download proteins from Uniprot

You can get a protein sequence from the Uniprot website by providing a Uniprot ID:

```
from propy.GetProteinFromUniprot import GetProteinSequence as gps

uniprotid = "P48039"
proseq = gps(uniprotid)

print(proseq)
```

gives

```
MQGNGSALPNASQPVLRGDGARPSWLASALACVLIFTIVVDILGNLLVILSVYRNKKLRNAGNIFVVSLAVA\
DLVVAIYPYPLVLMSIFNNGWNLGYLHCQVSGFLMGLSVIGSIFNITGIAINRYCYICHSLKYDKLYSSKNS\
LCYVLLIWLLTLAAVLPNLRAGTLQYDPRIYSCTFAQSVSSAYTIAVVVFHFLVPMIIVIFCYLRIWILVLQ\
VRQRVKPDRKPKLKPQDFRNFVTMFVVFVLFAICWAPLNFIGLAVASDPASMVPRIPEWLFVASYYMAYFNS\
CLNAIIYGLLNQNFRKEYRRIIVSLCTARVFFVDSSNDVADRVKWKPSPLMTNNNVVKVDSV
```

You can get the window 2 + 1 sub-sequences whose central point is the given amino acid ToAA.

```
from propy import GetSubSeq

subseq = GetSubSeq.GetSubSequence(proseq, ToAA="S", window=5)
print(subseq)
```

gives

```
[
    "MQGNGSALPNA",
    "ALPNASQPVLR",
    "DGARPSWLASA",
    "PSWLASALACV",
    "LLVILSVYRNK",
    "NIFVVSLAVAD",
    "PLVLMSIFNNG",
```

(continues on next page)

(continued from previous page)

```
    "LHCQVSGFLMG",
    "FLMGLSVIGSI",
    "LSVIGSIFNIT",
    "CYICHSLKYDK",
    "YDKLYSSKNSL",
    "DKLYSSKNSLC",
    "YSSKNSLCYVL",
    "DPRIYSCTFAQ",
    "CTFAQSVSSAY",
    "FAQSVSSAYTI",
    "AQSVSSAYTIA",
    "GLAVASDPASM",
    "ASDPASMVPRI",
    "WLFVASYYMAY",
    "MAYFNSCLNAI",
    "RRIIVSLCTAR",
    "VFFVDSSNDVA",
    "FFVDSSNDVAD",
    "VKWKPSPLMTN",
]
```

You can also get several protein sequences by providing a file containing Uniprot IDs of these proteins.

```
from propy.GetProteinFromUniprot import GetProteinSequenceFromTxt as gpst

tag = gpst("propy/data", "target.txt", "target1.txt")
```

prints

```
--------------------------------------------------------------------------------
The 1 protein sequence has been downloaded!
MADSCRNLTYVRGSVGPATSTLMFVAGVVGNGLALGILSARRPARPSAFAVLVTGLAATDLLGTSFLSPAVFVAYARNSSLLGLARGGPALCDAFAFAMTF
--------------------------------------------------------------------------------

TODO: HTTP Error 300!
```

The downloaded protein sequences have been saved in "propy/data/target1.txt".

You could check whether the input sequence is a valid protein sequence or not:

```
from propy import ProCheck

temp = ProCheck.ProteinCheck(proseq)
print(tmp)
```

which prints `350`. This output is the number of the protein sequence if it is valid; otherwise 0.

CHAPTER 3

Obtaining the property from the AAindex database

You could get the properties of amino acids from the AAindex database by providing a property name (e.g., KRIW790103). The output is given in the form of dictionary.

If the user provides the directory containing the AAindex database (the AAindex database could be downloaded from ftp://ftp.genome.jp/pub/db/community/aaindex/. It consists of three files: aaindex1, aaindex2 and aaindex3), the program will read the given database to get the property.

CHAPTER 4

Calculating protein descriptors

CHAPTER 5

propy.AAComposition

The module is used for computing the composition of amino acids, dipetide and 3-mers (tri-peptide) for a given protein sequence.

References

propy.AAComposition.**CalculateAAComposition**(*ProteinSequence: str*) → Dict[str, float]

Calculate the composition of Amino acids for a given protein sequence.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains the composition of 20 amino acids.

Return type Dict[str, float]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateAAComposition(protein)
```

propy.AAComposition.**CalculateAADipeptideComposition**(*ProteinSequence: str*) → Dict[str, float]

Calculate the composition of AADs, dipeptide and 3-mers for a given protein sequence.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains all composition values of AADs, dipeptide and 3-mers (8420).

Return type Dict[str, float]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateAADipeptideComposition(protein)
```

propy.AAComposition.**CalculateDipeptideComposition**(*ProteinSequence: str*) → Dict[str, float]

Calculate the composition of dipeptidefor a given protein sequence.

Parameters **ProteinSequence** (*a pure protein sequence*) –

Returns **result** – contains the composition of 400 dipeptides

Return type Dict[str, float]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateDipeptideComposition(protein)
```

propy.AAComposition.**GetSpectrumDict**(*proteinsequence: str*) → Dict[str, int]

Calcualte the spectrum descriptors of 3-mers for a given protein.

Parameters **proteinsequence** (*a pure protein sequence*) –

Returns **result** – contains the composition values of 8000 3-mers

Return type Dict[str, int]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetSpectrumDict(protein)
```

propy.AAComposition.**Getkmers**() → List[str]

Get the amino acid list of 3-mers.

Returns **result** – contains 8000 tri-peptides

Return type List[str]

Examples

```
>>> result = Getkmers()
```

CHAPTER 6

propy.AAIndex

This module is used for obtaining the properties of amino acids or their pairs from the aaindex database.

propy.AAIndex.**GetAAIndex1**(*name: str*, *path: Optional[str] = '.'*) → Dict[str, float]
Get the amino acid property values from aaindex1.

Parameters **name** (*str*) – name of amino acid property (e.g., KRIW790103)

Returns **result** – contains the properties of 20 amino acids

Return type Dict[str, float]

Examples

```
>>> result = GetAAIndex1("KRIW790103")
```

propy.AAIndex.**GetAAIndex23**(*name: str*, *path: Optional[str] = '.'*) → Dict[str, float]
Get the amino acid property values from aaindex2 and aaindex3.

Parameters **name** (*str*) – name of amino acid property (e.g. TANS760101, GRAR740104)

Returns **result** – contains the properties of 400 amino acid pairs

Return type Dict[str, float]

Examples

```
>>> result = GetAAIndex23("TANS760101")
```

class propy.AAIndex.**MatrixRecord**
Bases: *propy.AAIndex.Record*

Matrix record for mutation matrices or pair-wise contact potentials.

extend(*row*)
Extend self.index by the elements of the list.

get(*aai*, *aaj*, *d=None*)

median()

class propy.AAIndex.**Record**
Bases: object

Amino acid index (AAindex) Record.

aakeys = 'ARNDCQEGHILKMFPSTWYV'

extend(*row: List[Optional[float]]*) → None
Extend self.index by the elements of the list.

get(*aai*, *aaj=None*, *d=None*)

median()

propy.AAIndex.**get**(*key: str*)
Get record for key.

propy.AAIndex.**grep**(*pattern*)
Search for pattern in title and description of all records (case insensitive) and print results on standard output.

propy.AAIndex.**init**(*path: Optional[str] = None*, *index: str = '123'*)
Read in the aaindex files. You need to run this (once) before you can access any records. If the files are not within the current directory, you need to specify the correct directory path. By default all three aaindex files are read in.

propy.AAIndex.**init_from_file**(*filename*, *type=<class 'propy.AAIndex.Record'>*)

propy.AAIndex.**search**(*pattern*, *searchtitle=True*, *casesensitive=False*)
Search for pattern in description and title (optional) of all records and return matched records as list. By default search case insensitive.

CHAPTER 7

propy.Autocorrelation

This module is used for computing the Autocorrelation descriptors based different properties of AADs. You can also input your properties of AADs, then it can help you to compute Autocorrelation descriptors based on the property of AADs. Currently, you can get 720 descriptors for a given protein sequence based on our provided physicochemical properties of AADs.

References

propy.Autocorrelation.**CalculateAutoTotal**(*ProteinSequence: str*) → Dict[Any, Any]

Compute all autocorrelation descriptors based on 8 properties of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

- *result contains 30*8*3=720 normalized Moreau Broto, Moran, and Geary*
- *autocorrelation descriptors based on the given properties(i.e.,*
- *_AAPropert).*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateGearyAutoTotal(protein)
```

propy.Autocorrelation.**CalculateEachGearyAuto**(*ProteinSequence, AAP: Dict[Any, Any], AAPName*) → Dict[Any, Any]

Compute GearyAuto descriptors for different properties based on AADs.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence.

- **AAP** (*Dict[Any, Any]*) – contains the properties of 20 amino acids (e.g., _AvFlexibility).
- **AAPName** (*str*) – used for indicating the property (e.g., '_AvFlexibility').

Returns **result** – contains 30 Geary autocorrelation descriptors based on the given property.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> AAP, AAPName = _Hydrophobicity, "_Hydrophobicity"
>>> result = CalculateEachGearyAuto(protein, AAP, AAPName)
```

propy.Autocorrelation.**CalculateEachMoranAuto**(*ProteinSequence: str, AAP: Dict[Any, Any], AAPName: str*) → Dict[Any, Any]

Compute MoranAuto descriptors for different properties based on AADs.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence.
- **AAP** (*Dict[Any, Any]*) – contains the properties of 20 amino acids (e.g., _AvFlexibility).
- **AAPName** (*str*) – used for indicating the property (e.g., '_AvFlexibility').

Returns

- *result contains 30 Moran autocorrelation descriptors based on the given*
- *property.*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> AAP, AAPName = _Hydrophobicity, "_Hydrophobicity"
>>> result = CalculateEachMoranAuto(protein, AAP, AAPName)
```

propy.Autocorrelation.**CalculateEachNormalizedMoreauBrotoAuto**(*ProteinSequence: str, AAP: Dict[Any, Any], AAPName: str*) → Dict[str, float]

Compute MoreauBrotoAuto descriptors for different properties based on AADs.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **AAP** (*Dict[Any, Any]*) – contains the properties of 20 amino acids (e.g., _AvFlexibility).
- **AAPName** (*str*) – used for indicating the property (e.g., '_AvFlexibility').

Returns **result** – contains 30 Normalized Moreau-Broto autocorrelation descriptors based on the given property.

Return type Dict[str, float]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> AAP, AAPName = _Hydrophobicity, "_Hydrophobicity"
>>> result = CalculateEachNormalizedMoreauBrotoAuto(protein, AAP, AAPName)
```

propy.Autocorrelation.**CalculateGearyAuto**(*ProteinSequence*, *AAProperty*, *AAPropertyName*) → Dict[Any, Any]

A method used for computing GearyAuto for all properties.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **AAProperty** (*list or tuple form*) – contains the properties of 20 amino acids (e.g., _AAProperty).
- **AAPName** (*list or tuple form*) – used for indicating the property (e.g., '_AAPropertyName').

Returns

- *result contains 30*p Geary autocorrelation descriptors based on the given*
- *properties.*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> AAP, AAPName = _Hydrophobicity, "_Hydrophobicity"
>>> result = CalculateGearyAuto(protein, AAP, AAPName)
```

propy.Autocorrelation.**CalculateGearyAutoAvFlexibility**(*ProteinSequence: str*)

Calculte the Geary Autocorrelation descriptors based on AvFlexibility.

Parameters **ProteinSequence** (*str*) – a pure protein sequence.

Returns **result** – contains 30 Geary Autocorrelation descriptors based on AvFlexibility.

Return type Dict

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateGearyAutoAvFlexibility(protein)
```

propy.Autocorrelation.**CalculateGearyAutoFreeEnergy**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the Geary Autocorrelation descriptors based on FreeEnergy.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

Return type result contains 30 Geary Autocorrelation descriptors based on FreeEnergy.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateGearyAutoFreeEnergy(protein)
```

propy.Autocorrelation.**CalculateGearyAutoHydrophobicity**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the Geary Autocorrelation descriptors based on hydrophobicity.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

- *result contains 30 Geary Autocorrelation descriptors based on*
- *hydrophobicity.*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateGearyAutoHydrophobicity(protein)
```

propy.Autocorrelation.**CalculateGearyAutoMutability**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the Geary Autocorrelation descriptors based on Mutability.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

Return type result contains 30 Geary Autocorrelation descriptors based on Mutability.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateGearyAutoMutability(protein)
```

propy.Autocorrelation.**CalculateGearyAutoPolarizability**(*ProteinSequence: str*)

Calculte the Geary Autocorrelation descriptors based on Polarizability.

Parameters **ProteinSequence** (*str*) – a pure protein sequence.

Returns **result** – contains 30 Geary Autocorrelation descriptors based on Polarizability.

Return type Dict

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateGearyAutoPolarizability(protein)
```

propy.Autocorrelation.**CalculateGearyAutoResidueASA**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the Geary Autocorrelation descriptors based on ResidueASA.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

Return type result contains 30 Geary Autocorrelation descriptors based on ResidueASA.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateGearyAutoResidueASA(protein)
```

propy.Autocorrelation.**CalculateGearyAutoResidueVol**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the Geary Autocorrelation descriptors based on ResidueVol.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

Return type result contains 30 Geary Autocorrelation descriptors based on ResidueVol.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateGearyAutoResidueVol(protein)
```

propy.Autocorrelation.**CalculateGearyAutoSteric**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the Geary Autocorrelation descriptors based on Steric.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

Return type result contains 30 Geary Autocorrelation descriptors based on Steric.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateGearyAutoSteric(protein)
```

propy.Autocorrelation.**CalculateGearyAutoTotal**(*ProteinSequence: str*) → Dict[Any, Any]

Compute Geary autocorrelation descriptors based on 8 properties of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

- *result contains 30*8=240 Geary autocorrelation descriptors based on the*
- *given properties(i.e., _AApropert).*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateGearyAutoTotal(protein)
```

propy.Autocorrelation.**CalculateMoranAuto**(*ProteinSequence*, *AAProperty*, *AAPropertyName*) → Dict[Any, Any]

A method used for computing MoranAuto for all properties.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **AAProperty** (*list or tuple form*) – contains the properties of 20 amino acids (e.g., _AAProperty).
- **AAPName** (*list or tuple form*) – used for indicating the property (e.g., '_AAPropertyName').

Returns

- *result contains 30*p Moran autocorrelation descriptors based on the given*
- *properties.*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> AAP, AAPName = _Hydrophobicity, "_Hydrophobicity"
>>> result = CalculateMoranAuto(protein, AAP, AAPName)
```

propy.Autocorrelation.**CalculateMoranAutoAvFlexibility**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the MoranAuto Autocorrelation descriptors based on AvFlexibility.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

- *result contains 30 Moran Autocorrelation descriptors based on*
- *AvFlexibility.*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateMoranAutoAvFlexibility(protein)
```

propy.Autocorrelation.**CalculateMoranAutoFreeEnergy**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the MoranAuto Autocorrelation descriptors based on FreeEnergy.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

Return type result contains 30 Moran Autocorrelation descriptors based on FreeEnergy.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateMoranAutoFreeEnergy(protein)
```

propy.Autocorrelation.**CalculateMoranAutoHydrophobicity**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the MoranAuto Autocorrelation descriptors based on hydrophobicity.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

- *result contains 30 Moran Autocorrelation descriptors based on*
- *hydrophobicity.*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateMoranAutoHydrophobicity(protein)
```

propy.Autocorrelation.**CalculateMoranAutoMutability**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the MoranAuto Autocorrelation descriptors based on Mutability.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

Return type result contains 30 Moran Autocorrelation descriptors based on Mutability.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateMoranAutoMutability(protein)
```

propy.Autocorrelation.**CalculateMoranAutoPolarizability**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the MoranAuto Autocorrelation descriptors based on Polarizability.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

- *result contains 30 Moran Autocorrelation descriptors based on*
- *Polarizability.*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateMoranAutoPolarizability(protein)
```

propy.Autocorrelation.**CalculateMoranAutoResidueASA**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the MoranAuto Autocorrelation descriptors based on ResidueASA.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

Return type result contains 30 Moran Autocorrelation descriptors based on ResidueASA.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateMoranAutoResidueASA(protein)
```

propy.Autocorrelation.**CalculateMoranAutoResidueVol**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the MoranAuto Autocorrelation descriptors based on ResidueVol.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

Return type result contains 30 Moran Autocorrelation descriptors based on ResidueVol.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateMoranAutoResidueVol(protein)
```

propy.Autocorrelation.**CalculateMoranAutoSteric**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the MoranAuto Autocorrelation descriptors based on AutoSteric.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

Return type result contains 30 Moran Autocorrelation descriptors based on AutoSteric.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateMoranAutoSteric(protein)
```

propy.Autocorrelation.**CalculateMoranAutoTotal**(*ProteinSequence: str*) → Dict[Any, Any]

Compute Moran autocorrelation descriptors based on 8 properties of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

- *result contains 30*8=240 Moran autocorrelation descriptors based on the*
- *given properties(i.e., _AApropert).*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateMoranAutoTotal(protein)
```

propy.Autocorrelation.**CalculateNormalizedMoreauBrotoAuto**(*ProteinSequence*, *AAProperty*, *AAPropertyName*) → Dict[Any, Any]

A method used for computing MoreauBrotoAuto for all properties.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **AAProperty** (*a list or tuple form*) – contains the properties of 20 amino acids (e.g., _AAProperty).
- **AAPName** (*a list or tuple form*) – used for indicating the property (e.g., '_AAPropertyName').

Returns

- *result contains 30*p Normalized Moreau-Broto autocorrelation descriptors*
- *based on the given properties.*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> AAP, AAPName = _Hydrophobicity, "_Hydrophobicity"
>>> result = CalculateNormalizedMoreauBrotoAuto(protein, AAP, AAPName)
```

propy.Autocorrelation.**CalculateNormalizedMoreauBrotoAutoAvFlexibility**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the NormalizedMoreauBorto Autocorrelation descriptors based on AvFlexibility.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

- *result contains 30 Normalized Moreau-Broto Autocorrelation descriptors*
- *based on AvFlexibility.*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateNormalizedMoreauBrotoAutoAvFlexibility(protein)
```

propy.Autocorrelation.**CalculateNormalizedMoreauBrotoAutoFreeEnergy**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the NormalizedMoreauBorto Autocorrelation descriptors based on FreeEnergy.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

- *result contains 30 Normalized Moreau-Broto Autocorrelation descriptors*
- *based on FreeEnergy.*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateNormalizedMoreauBrotoAutoFreeEnergy(protein)
```

propy.Autocorrelation.**CalculateNormalizedMoreauBrotoAutoHydrophobicity**(*ProteinSequence*) → Dict[Any, Any]

Calculte the NormalizedMoreauBorto Autocorrelation descriptors based on hydrophobicity.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

- *result contains 30 Normalized Moreau-Broto Autocorrelation descriptors*
- *based on Hydrophobicity.*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateNormalizedMoreauBrotoAutoHydrophobicity(protein)
```

propy.Autocorrelation.**CalculateNormalizedMoreauBrotoAutoMutability**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the NormalizedMoreauBorto Autocorrelation descriptors based on Mutability.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

- *result contains 30 Normalized Moreau-Broto Autocorrelation descriptors*
- *based on Mutability.*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateNormalizedMoreauBrotoAutoMutability(protein)
```

propy.Autocorrelation.**CalculateNormalizedMoreauBrotoAutoPolarizability**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the NormalizedMoreauBorto Autocorrelation descriptors based on Polarizability.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

- *result contains 30 Normalized Moreau-Broto Autocorrelation descriptors*
- *based on Polarizability.*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateNormalizedMoreauBrotoAutoPolarizability(protein)
```

propy.Autocorrelation.**CalculateNormalizedMoreauBrotoAutoResidueASA**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the NormalizedMoreauBorto Autocorrelation descriptors based on ResidueASA.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

- *result contains 30 Normalized Moreau-Broto Autocorrelation descriptors*
- *based on ResidueASA.*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateNormalizedMoreauBrotoAutoResidueASA(protein)
```

propy.Autocorrelation.**CalculateNormalizedMoreauBrotoAutoResidueVol**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the NormalizedMoreauBorto Autocorrelation descriptors based on ResidueVol.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

- *result contains 30 Normalized Moreau-Broto Autocorrelation descriptors*
- *based on ResidueVol.*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateNormalizedMoreauBrotoAutoResidueVol(protein)
```

propy.Autocorrelation.**CalculateNormalizedMoreauBrotoAutoSteric**(*ProteinSequence: str*) → Dict[Any, Any]

Calculte the NormalizedMoreauBorto Autocorrelation descriptors based on Steric.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

- *result contains 30 Normalized Moreau-Broto Autocorrelation descriptors*
- *based on Steric.*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateNormalizedMoreauBrotoAutoSteric(protein)
```

propy.Autocorrelation.**CalculateNormalizedMoreauBrotoAutoTotal**(*ProteinSequence: str*) → Dict[Any, Any]

Compute normalized Moreau Broto autocorrelation descriptors based on 8 proterties of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

- *result contains 30*8=240 normalized Moreau Broto autocorrelation*
- *descriptors based on the given properties(i.e., _AAPropert).*

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateNormalizedMoreauBrotoAutoTotal(protein)
```

propy.Autocorrelation.**NormalizeEachAAP**(*AAP: Dict[Any, Any]*) → Dict[Any, Any]

Centralizes and standardizes all amino acid indices before the calculation

Parameters **AAP** (*Dict[Any, Any]*) – contains the properties of 20 amino acids

Returns **result** – contains the normalized properties of 20 amino acids

Return type Dict

CHAPTER 8

propy.CTD

Compute the composition, transition and distribution descriptors based on the different properties of AADs.

The AADs with the same properties is marked as the same number. You can get 147 descriptors for a given protein sequence.

References

propy.CTD.**CalculateC**(*ProteinSequence: str*) → Dict[Any, Any]
Calculate all composition descriptors based seven different properties of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains all composition descriptors.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateC(protein)
```

propy.CTD.**CalculateCTD**(*ProteinSequence: str*) → Dict[Any, Any]
Calculate all CTD descriptors based seven different properties of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains all CTD descriptors.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateCTD(protein)
```

propy.CTD.**CalculateComposition**(*ProteinSequence: str, AAProperty: Dict[Any, Any], AAPName: str*) → Dict[Any, Any]

Compute composition descriptors.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **AAProperty** (*Dict[Any, Any]*) – contains classifciation of amino acids such as _Polarizability.
- **AAPName** (*str*) – used for indicating a AAP name.

Returns **result** – contains composition descriptors based on the given property.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> AAProperty, AAPName = _Hydrophobicity, "_Hydrophobicity"
>>> result = CalculateComposition(protein, AAProperty, AAPName)
```

propy.CTD.**CalculateCompositionCharge**(*ProteinSequence: str*) → Dict[Any, Any]

Calculate composition descriptors based on Charge of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Composition descriptors based on Charge.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateCompositionCharge(protein)
```

propy.CTD.**CalculateCompositionHydrophobicity**(*ProteinSequence: str*)

Calculate composition descriptors based on Hydrophobicity of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Composition descriptors based on Hydrophobicity.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateCompositionHydrophobicity(protein)
```

propy.CTD.**CalculateCompositionNormalizedVDWV**(*ProteinSequence: str*)
Calculate composition descriptors based on NormalizedVDWV of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Composition descriptors based on NormalizedVDWV.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateCompositionNormalizedVDWV(protein)
```

propy.CTD.**CalculateCompositionPolarity**(*ProteinSequence: str*)
Calculate composition descriptors based on Polarity of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Composition descriptors based on Polarity.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateCompositionPolarity(protein)
```

propy.CTD.**CalculateCompositionPolarizability**(*ProteinSequence: str*) → Dict[Any, Any]
Calculate composition descriptors based on Polarizability of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Composition descriptors based on Polarizability.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateCompositionPolarizability(protein)
```

propy.CTD.**CalculateCompositionSecondaryStr**(*ProteinSequence: str*) → Dict[Any, Any]
Calculate composition descriptors based on SecondaryStr of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Composition descriptors based on SecondaryStr.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateCompositionSecondaryStr(protein)
```

propy.CTD.**CalculateCompositionSolventAccessibility**(*ProteinSequence: str*) → Dict[Any, Any]

Clculate composition descriptors based on SolventAccessibility of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Composition descriptors based on SolventAccessibility.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateCompositionSolventAccessibility(protein)
```

propy.CTD.**CalculateD**(*ProteinSequence: str*) → Dict[Any, Any]

Calculate all distribution descriptors based seven different properties of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains all distribution descriptors.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateD(protein)
```

propy.CTD.**CalculateDistribution**(*ProteinSequence: str, AAProperty: Dict[Any, Any], AAPName: str*) → Dict[Any, Any]

Compute distribution descriptors.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence.
- **AAProperty** (*Dict[Any, Any]*) – contains classifciation of amino acids such as _Polarizability
- **AAPName** (*str*) –

Returns **result** – contains Distribution descriptors based on the given property.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> AAProperty, AAPName = _Hydrophobicity, "_Hydrophobicity"
>>> result = CalculateDistribution(protein, AAProperty, AAPName)
```

propy.CTD.**CalculateDistributionCharge**(*ProteinSequence: str*) → Dict[Any, Any]
Calculate Distribution descriptors based on Charge of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Distribution descriptors based on Charge.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateDistributionCharge(protein)
```

propy.CTD.**CalculateDistributionHydrophobicity**(*ProteinSequence: str*) → Dict[Any, Any]
Calculate Distribution descriptors based on Hydrophobicity of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Distribution descriptors based on Hydrophobicity.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateDistributionHydrophobicity(protein)
```

propy.CTD.**CalculateDistributionNormalizedVDWV**(*ProteinSequence: str*) → Dict[Any, Any]
Calculate Distribution descriptors based on NormalizedVDWV of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Distribution descriptors based on NormalizedVDWV.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateDistributionNormalizedVDWV(protein)
```

propy.CTD.**CalculateDistributionPolarity**(*ProteinSequence: str*) → Dict[Any, Any]
Calculate Distribution descriptors based on Polarity of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Distribution descriptors based on Polarity.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateDistributionPolarity(protein)
```

propy.CTD.**CalculateDistributionPolarizability**(*ProteinSequence: str*) → Dict[Any, Any]
Calculate Distribution descriptors based on Polarizability of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Distribution descriptors based on Polarizability.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateDistributionPolarizability(protein)
```

propy.CTD.**CalculateDistributionSecondaryStr**(*ProteinSequence: str*) → Dict[Any, Any]
Calculate Distribution descriptors based on SecondaryStr of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Distribution descriptors based on SecondaryStr.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateDistributionSecondaryStr(protein)
```

propy.CTD.**CalculateDistributionSolventAccessibility**(*ProteinSequence: str*) → Dict[Any, Any]
Calculate Distribution descriptors based on SolventAccessibility of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Distribution descriptors based on SolventAccessibility.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateDistributionSolventAccessibility(protein)
```

propy.CTD.**CalculateT**(*ProteinSequence: str*) → Dict[Any, Any]
Calculate all transition descriptors based seven different properties of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains all transition descriptors.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateT(protein)
```

propy.CTD.**CalculateTransition**(*ProteinSequence: str, AAProperty: Dict[Any, Any], AAPName: str*) → Dict[Any, Any]

Compute transition descriptors.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **AAProperty** (*Dict[Any, Any]*) – contains classifciation of amino acids such as _Polarizability.
- **AAPName** (*str*) – used for indicating a AAP name.

Returns **result** – contains transition descriptors based on the given property.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> AAProperty, AAPName = _Hydrophobicity, "_Hydrophobicity"
>>> result = CalculateTransition(protein, AAProperty, AAPName)
```

propy.CTD.**CalculateTransitionCharge**(*ProteinSequence: str*) → Dict[Any, Any]

Calculate Transition descriptors based on Charge of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Transition descriptors based on Charge.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateTransitionCharge(protein)
```

propy.CTD.**CalculateTransitionHydrophobicity**(*ProteinSequence: str*) → Dict[Any, Any]

Calculate Transition descriptors based on Hydrophobicity of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Transition descriptors based on Hydrophobicity.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateTransitionHydrophobicity(protein)
```

propy.CTD.**CalculateTransitionNormalizedVDWV**(*ProteinSequence: str*) → Dict[Any, Any]
Calculate Transition descriptors based on NormalizedVDWV of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Transition descriptors based on NormalizedVDWV.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateTransitionNormalizedVDWV(protein)
```

propy.CTD.**CalculateTransitionPolarity**(*ProteinSequence: str*) → Dict[Any, Any]
Calculate Transition descriptors based on Polarity of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Transition descriptors based on Polarity.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateTransitionPolarity(protein)
```

propy.CTD.**CalculateTransitionPolarizability**(*ProteinSequence: str*) → Dict[Any, Any]
Calculate Transition descriptors based on Polarizability of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Transition descriptors based on Polarizability.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateTransitionPolarizability(protein)
```

propy.CTD.**CalculateTransitionSecondaryStr**(*ProteinSequence: str*) → Dict[Any, Any]
Calculate Transition descriptors based on SecondaryStr of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Transition descriptors based on SecondaryStr.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateTransitionSecondaryStr(protein)
```

propy.CTD.**CalculateTransitionSolventAccessibility**(*ProteinSequence: str*) → Dict[Any, Any]

Calculate Transition descriptors based on SolventAccessibility of AADs.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains Transition descriptors based on SolventAccessibility.

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateTransitionSolventAccessibility(protein)
```

propy.CTD.**StringtoNum**(*ProteinSequence: str, AAProperty: Dict[Any, Any]*) → str

Tranform the protein sequence into the string form such as 32123223132121123.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **AAProperty** (*Dict[Any, Any]*) – contains classifciation of amino acids such as _Polarizability.

Returns **result** – e.g. 123321222132111123222

Return type str

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> AAProperty, AAPName = _Hydrophobicity, "_Hydrophobicity"
>>> result = StringtoNum(protein, AAProperty)
```

CHAPTER 9

propy.GetProteinFromUniprot

Download the protein sequence from the uniprot website.

You can only need input a protein ID or prepare a file (ID.txt) related to ID. You can obtain a .txt (ProteinSequence.txt) file saving protein sequence you need.

propy.GetProteinFromUniprot.**GetProteinSequence**(*ProteinID: str*) → str

Get the protein sequence from the uniprot website by ID.

Parameters **ProteinID** (*str*) – indicating ID such as "P48039" or "Q9NQ39".

Returns **protein_sequence**

Return type str

Examples

```
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
```

propy.GetProteinFromUniprot.**GetProteinSequenceFromTxt**(*path: str*, *openfile: str*, *savefile: str*)

Get the protein sequence from the uniprot website by the file containing ID.

Parameters

- **path** (*str*) – a directory path containing the ID file such as "/home/orient/protein/"
- **openfile** (*str*) – the ID file such as "proteinID.txt"
- **savefile** (*str*) – the file saving the obtained protein sequences such as "protein.txt"

CHAPTER 10

propy.GetSubSeq

The prediction of functional sites (e.g. methylation) of proteins usually needs to split the total protein into a set of segments around specific amino acid. Given a specific window size p, we can obtain all segments of length equal to (2*p+1) very easily. Note that the output of the method is a list form.

propy.GetSubSeq.**GetSubSequence**(*ProteinSequence: str*, *ToAA: str = 'S'*, *window: int = 3*) → List[str]

Get all 2*window+1 sub-sequences whose cener is ToAA in a protein.

Parameters

- **ProteinSequence** (*str*) – a pure problem sequence
- **ToAA** (*str*) – the central (query point) amino acid in the sub-sequence
- **window** (*int*) – the span

Returns **result** – contains all satisfied sub-sequences

Return type List[str]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetSubSequence(protein)
```

CHAPTER 11

propy.ProCheck

Check whether the input protein sequence is a valid amino acid sequence.

propy.ProCheck.**ProteinCheck**(*ProteinSequence: str*) → int

Check whether the protein sequence is a valid amino acid sequence or not.

Parameters **ProteinSequence** (*a pure protein sequence*) –

Returns **flag** – if the check is no problem, result will return the length of protein. if the check has problems, result will return 0.

Return type bool

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = ProteinCheck(protein)
```

CHAPTER 12

propy.PseudoAAC

Instead of using the conventional 20-D amino acid composition to represent the sample of a protein, Prof. Kuo-Chen Chou proposed the pseudo amino acid (PseAA) composition in order for inluding the sequence-order information. Based on the concept of Chou's pseudo amino acid composition, the server PseAA was designed in a flexible way, allowing users to generate various kinds of pseudo amino acid composition for a given protein sequence by selecting different parameters and their combinations. This module aims at computing two types of PseAA descriptors: Type I and Type II.

References

The hydrophobicity values are from JACS, 1962, 84: 4240-4246. (C. Tanford).

The hydrophilicity values are from PNAS, 1981, 78:3824-3828 (T.P.Hopp & K.R.Woods).

The side-chain mass for each of the 20 amino acids.

CRC Handbook of Chemistry and Physics, 66th ed., CRC Press, Boca Raton, Florida (1985).

R.M.C. Dawson, D.C. Elliott, W.H. Elliott, K.M. Jones, Data for Biochemical Research 3rd ed.,

Clarendon Press Oxford (1986).

propy.PseudoAAC.**GetAAComposition**(*ProteinSequence: str*) → Dict[Any, Any]

Calculate the composition of Amino acids for a given protein sequence.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns

Return type result is a dict form containing the composition of 20 amino acids.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetAAComposition(protein)
```

propy.PseudoAAC.**GetAPseudoAAC**(*ProteinSequence*, *lamda: int = 30*, *weight: float = 0.5*)

Computing all of type II pseudo-amino acid compostion descriptors based on the given properties. Note that the number of PAAC strongly depends on the lamda value. if lamda = 20, we can obtain 20+20=40 PAAC descriptors. The size of these values depends on the choice of lamda and weight simultaneously.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **lamda** (*int*) – reflects the rank of correlation and is a non-Negative integer, such as 15. Note that (1)lamda should NOT be larger than the length of input protein sequence; (2) lamda must be non-Negative integer, such as 0, 1, 2, ...; (3) when lamda =0, the output of PseAA server is the 20-D amino acid composition.
- **weight** (*float*) – is designed for the users to put weight on the additional PseAA components with respect to the conventional AA components. The user can select any value within the region from 0.05 to 0.7 for the weight factor.

Returns **result** – contains calculated 20+lamda PAAC descriptors

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetAPseudoAAC(protein)
```

propy.PseudoAAC.**GetAPseudoAAC1**(*ProteinSequence*, *lamda=30*, *weight=0.5*)

Computing the first 20 of type II pseudo-amino acid compostion descriptors based on

[_Hydrophobicity, _hydrophilicity].

propy.PseudoAAC.**GetAPseudoAAC2**(*ProteinSequence*, *lamda=30*, *weight=0.5*)

Computing the last lamda of type II pseudo-amino acid compostion descriptors

based on (_Hydrophobicity, _hydrophilicity).

propy.PseudoAAC.**GetCorrelationFunction**(*Ri='S'*, *Rj='D'*, *AAP=None*)

Computing the correlation between two given amino acids using the given properties.

Parameters

- **Ri** (*str*) – amino acids
- **Rj** (*str*) – amino acids
- **AAP** (*List[Any]*) – contains the properties, each of which is a dict form.

Returns

Return type result is the correlation value between two amino acids.

Examples

```
>>> GetCorrelationFunction(Ri="S", Rj="D", AAP=_Hydrophobicity)
```

propy.PseudoAAC.**GetPseudoAAC**(*ProteinSequence: str*, *lamda: int = 30*, *weight: float = 0.05*, *AAP=None*)

Computing all of type I pseudo-amino acid compostion descriptors based on the given properties. Note that

the number of PAAC strongly depends on the lamda value. if lamda = 20, we can obtain 20+20=40 PAAC descriptors. The size of these values depends on the choice of lamda and weight simultaneously. You must specify some properties into AAP.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **lamda** (*int*) – reflects the rank of correlation and is a non-Negative integer, such as 15. Note that (1)lamda should NOT be larger than the length of input protein sequence; (2) lamda must be non-Negative integer, such as 0, 1, 2, . . . ; (3) when lamda =0, the output of PseAA server is the 20-D amino acid composition.
- **weight** (*float*) – is designed for the users to put weight on the additional PseAA components with respect to the conventional AA components. The user can select any value within the region from 0.05 to 0.7 for the weight factor.
- **AAP** (*List[Any]*) – contains the properties, each of which is a dict form.

Returns

Return type result is a dict form containing calculated 20+lamda PAAC descriptors.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetPseudoAAC(protein)
```

propy.PseudoAAC.**GetPseudoAAC1**(*ProteinSequence*, *lamda=30*, *weight=0.05*, *AAP=None*)

Computing the first 20 of type I pseudo-amino acid compostion descriptors based on the given properties.

propy.PseudoAAC.**GetPseudoAAC2**(*ProteinSequence*, *lamda: int = 30*, *weight: float = 0.05*, *AAP=None*)

Compute the last lamda of type I pseudo-amino acid compostion descriptors based on the given properties.

propy.PseudoAAC.**GetSequenceOrderCorrelationFactor**(*ProteinSequence*, *k: int = 1*, *AAP=None*)

Computing the Sequence order correlation factor with gap equal to k based on the given properities.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **k** (*int*) – the gap.
- **AAP** (*List[Any]*) – contains the properties, each of which is a dict form.

Returns

Return type result is the correlation factor value with the gap equal to k.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetSequenceOrderCorrelationFactor(protein)
```

propy.PseudoAAC.**GetSequenceOrderCorrelationFactorForAPAAC**(*ProteinSequence*, *k=1*)

Computing the Sequence order correlation factor with gap equal to k based on

[_Hydrophobicity, _hydrophilicity] for APAAC (type II PseAAC) .

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **is the gap.** (*k*) –

Returns

Return type result is the correlation factor value with the gap equal to k.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetSequenceOrderCorrelationFactorForAPAAC(protein)
```

propy.PseudoAAC.**NormalizeEachAAP**(*AAP*)

All of the amino acid indices are centralized and standardized before the calculation.

Parameters **is a dict form containing the properties of 20 amino acids.** (*AAP*) –

Returns

- *result is the a dict form containing the normalized properties of 20 amino*
- *acids.*

Examples

```
>>> result = NormalizeEachAAP(AAP=_Hydrophobicity)
```

CHAPTER 13

propy.PyPro

Computing different types of protein descriptors.

class propy.PyPro.**GetProDes**(*ProteinSequence: str = ''*)

Bases: object

Collect all descriptor calcualtion modules.

AALetter = ['A', 'R', 'N', 'D', 'C', 'E', 'Q', 'G', 'H', 'I', 'L', 'K', 'M', 'F', 'P',

GetAAComp() → Dict[str, float]

Amino acid compositon descriptors (20).

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetAAComp()
```

GetAAindex1(*name: str*, *path: Optional[str] = '.'*) → Dict[str, float]

Get the amino acid property values from aaindex1.

Parameters **name** (*str*) – is the name of amino acid property (e.g., KRIW790103)

Returns

Return type result is a dict form containing the properties of 20 amino acids

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetAAindex1(name="KRIW790103")
```

GetAAindex23 (*name: str*, *path: Optional[str] = '.'*) → Dict[str, float]
Get the amino acid property values from aaindex2 and aaindex3.

Parameters **is the name of amino acid property (e.g. TANS760101, GRAR740104)** (*name*) –

Returns

Return type result is a dict form containing the properties of 400 amino acid pairs

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetAAindex23(name="KRIW790103")
```

GetALL (*paac_lamda: int = 10*, *paac_weight: float = 0.05*, *apaac_lamda: int = 10*, *apaac_weight: float = 0.5*, *socn_maxlag: int = 45*, *qso_maxlag: int = 30*, *qso_weight: float = 0.1*) → Dict[Any, Any]
Calcualte all descriptors except tri-peptide descriptors.

Parameters

- **paac_lamda** (*int, optional (default: 10)*) – used by GetPAAC() reflects the rank of correlation and is a non-Negative integer, such as 15. Note that (1)lamda should NOT be larger than the length of input protein sequence; (2) lamda must be non-Negative integer, such as 0, 1, 2, ...; (3) when lamda =0, the output of PseAA server is the 20-D amino acid composition.
- **paac_weight** (*float, optional (default: 0.05)*) – used by GetPAAC() is designed for the users to put weight on the additional PseAA components with respect to the conventional AA components. The user can select any value within the region from 0.05 to 0.7 for the weight factor.
- **apaac_lamda** (*int, optional (default: 10)*) – Same as "paac_lambda" but for APAAC()
- **apaac_weight** (*float, optional (default: 0.5)*) – Same as "paac_weight" but for APAAC()
- **socn_maxlag** (*int, optional (default: 45)*) – Used by GetSOCN() is the maximum lag and the length of the protein should be larger than maxlag.
- **qso_maxlag** (*int, optional (default: 30)*) – Used by GetQSO() is the maximum lag and the length of the protein should be larger than maxlag.
- **qso_weight** (*float, optional (default: 0.1)*) – Used by GetQSO()

GetAPAAC (*lamda: int = 10*, *weight: float = 0.5*) → Dict[Any, Any]
Amphiphilic (Type II) Pseudo amino acid composition descriptors.

default is 30

Parameters

- **lamda** (*int, optional (default: 10)*) – reflects the rank of correlation and is a non-Negative integer, such as 15. Note that (1)lamda should NOT be larger than the length of input protein sequence; (2) lamda must be non-Negative integer, such as 0, 1, 2, ...; (3) when lamda =0, the output of PseAA server is the 20-D amino acid composition.

- **weight** (*float, optional (default: 0.05)*) – is designed for the users to put weight on the additional PseAA components with respect to the conventional AA components. The user can select any value within the region from 0.05 to 0.7 for the weight factor.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetAPAAC(lamda=10, weight=0.5)
```

GetCTD() → Dict[Any, Any]
Composition Transition Distribution descriptors (147).

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetCTD()
```

GetDPComp() → Dict[str, float]
Dipeptide composition descriptors (400).

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetDPComp()
```

GetGearyAuto() → Dict[Any, Any]
Geary autocorrelation descriptors (240).

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetGearyAuto()
```

GetGearyAutop(*AAP: Optional[Dict[Any, Any]] = None*, *AAPName: str = 'p'*) → Dict[Any, Any]
Geary autocorrelation descriptors for the given property (30).

Parameters **AAP** (*Dict[Any, Any]*) – contains physicochemical properities of 20 amino acids

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetGearyAutop(AAP={}, AAPName='p')
```

GetMoranAuto() → Dict[Any, Any]
Moran autocorrelation descriptors (240).

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetMoranAuto()
```

GetMoranAutop(*AAP: Optional[Dict[Any, Any]] = None*, *AAPName: str = 'p'*) → Dict[Any, Any]
Moran autocorrelation descriptors for the given property (30).

Parameters **AAP** (*Dict[Any, Any]*) – contains physicochemical properities of 20 amino acids

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetMoranAutop(AAP={}, AAPName='p')
```

GetMoreauBrotoAuto() → Dict[Any, Any]
Normalized Moreau-Broto autocorrelation descriptors (240).

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetMoreauBrotoAuto()
```

GetMoreauBrotoAutop(*AAP: Optional[Dict[Any, Any]] = None*, *AAPName: str = 'p'*) → Dict[str, float]
Normalized Moreau-Broto autocorrelation descriptors for the given property (30).

Parameters **AAP** (*Dict[Any, Any]*) – contains physicochemical properities of 20 amino acids

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetMoreauBrotoAutop(AAP={}, AAPName='p')
```

GetPAAC(*lamda: int = 10*, *weight: float = 0.05*) → Dict[Any, Any]
Type I Pseudo amino acid composition descriptors (default is 30).

Parameters

- **lamda** (*int, optional (default: 10)*) – reflects the rank of correlation and is a non-Negative integer, such as 15. Note that (1)lamda should NOT be larger than the length of input protein sequence; (2) lamda must be non-Negative integer, such as 0, 1, 2, . . . ; (3) when lamda =0, the output of PseAA server is the 20-D amino acid composition.

- **weight** (*float, optional (default: 0.05)*) – is designed for the users to put weight on the additional PseAA components with respect to the conventional AA components. The user can select any value within the region from 0.05 to 0.7 for the weight factor.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetPAAC(lamda=10, weight=0.05)
```

GetPAACp(*lamda: int = 10*, *weight: float = 0.05*, *AAP: Optional[List[Any]] = None*) → Dict[Any, Any]

Type I Pseudo amino acid composition descriptors for the given properties

Default is 30.

Parameters

- **lamda** (*int, optional (default: 10)*) – reflects the rank of correlation and is a non-Negative integer, such as 15. Note that (1)lamda should NOT be larger than the length of input protein sequence; (2) lamda must be non-Negative integer, such as 0, 1, 2, . . . ; (3) when lamda =0, the output of PseAA server is the 20-D amino acid composition.
- **weight** (*float, optional (default: 0.05)*) – is designed for the users to put weight on the additional PseAA components with respect to the conventional AA components. The user can select any value within the region from 0.05 to 0.7 for the weight factor.
- **AAP** (*List*) – contains the properties, each of which is a dict form.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetPAACp(lamda=10, weight=0.05, AAP=[])
```

GetQSO(*maxlag: int = 30*, *weight: float = 0.1*) → Dict[Any, Any]

Quasi sequence order descriptors default is 50.

Parameters

- **= GetQSO(maxlag=30, weight=0.1)** (*result*) –
- **is the maximum lag and the length of the protein should be larger** (*maxlag*) –
- **maxlag. default is 45.** (*than*) –

GetQSOp(*maxlag: int = 30*, *weight: float = 0.1*, *distancematrix: Optional[Dict[Any, Any]] = None*) → Dict[Any, Any]

Quasi sequence order descriptors default is 50.

Parameters

- **= GetQSO(maxlag=30, weight=0.1)** (*result*) –
- **is the maximum lag and the length of the protein should be larger** (*maxlag*) –

- **maxlag. default is 45.** (*than*) –
- **is a dict form containing 400 distance values** (*distancematrix*) –

GetSOCN (*maxlag: int = 45*) → Dict[Any, Any]
Sequence order coupling numbers default is 45.

Parameters **maxlag** (*int*) – is the maximum lag and the length of the protein should be larger than maxlag

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetSOCN(maxlag=45)
```

GetSOCNp (*maxlag: int = 45*, *distancematrix: Optional[Dict[Any, Any]] = None*) → Dict[Any, Any]
Sequence order coupling numbers default is 45.

Parameters

- **is the maximum lag and the length of the protein should be larger** (*maxlag*) –
- **maxlag. default is 45.** (*than*) –
- **is a dict form containing 400 distance values** (*distancematrix*) –

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetSOCN(maxlag=45)
```

GetSubSeq (*ToAA: str = 'S'*, *window: int = 3*) → List[str]
Obtain the sub sequences wit length 2*window+1, whose central point is ToAA.

ToAA is the central (query point) amino acid in the sub-sequence.

window is the span.

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetSubSeq(ToAA='S', window=3)
```

GetTPComp () → Dict[str, int]
Tri-peptide composition descriptors (8000).

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetProDes(protein).GetTPComp()
```

Version = 1.0

CHAPTER 14

propy.QuasiSequenceOrder

Compute the quasi sequence order descriptors based on the given protein sequence. We can obtain two types of descriptors: Sequence-order-coupling number and quasi-sequence-order descriptors. Two distance matrixes between 20 amino acids are employed.

References

propy.QuasiSequenceOrder.**GetAAComposition**(*ProteinSequence: str*) → Dict[str, float]

Calculate the composition of Amino acids for a given protein sequence.

Parameters **ProteinSequence** (*str*) – a pure protein sequence

Returns **result** – contains the composition of 20 amino acids.

Return type Dict[str, float]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> from propy.AAComposition import CalculateAAComposition
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = CalculateAAComposition(protein)
```

propy.QuasiSequenceOrder.**GetQuasiSequenceOrder**(*ProteinSequence: str*, *maxlag: int = 30*, *weight: float = 0.1*) → Dict[Any, Any]

Compute quasi-sequence-order descriptors for a given protein.

See[1] for details.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence

[1] Kuo-Chen Chou. Prediction of Protein Subcellar Locations by Incorporating Quasi-Sequence-Order Effect. Biochemical and Biophysical Research Communications 2000, 278, 477-483.

- **maxlag** (*int, optional (default: 30)*) – the maximum lag and the length of the protein should be larger than maxlag
- **weight** (*float, optional (default: 0.1)*) – a weight factor. Please see reference 1 for its choice.

Returns **result** – contains all quasi-sequence-order descriptors

Return type Dict

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetQuasiSequenceOrder(protein)
```

propy.QuasiSequenceOrder.**GetQuasiSequenceOrder1**(*ProteinSequence: str, maxlag: int = 30, weight: float = 0.1, distancematrix=None*)

Compute the first 20 quasi-sequence-order descriptors for a given protein sequence.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **maxlag** (*int, optional (default: 30)*) – the maximum lag and the length of the protein should be larger

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetQuasiSequenceOrder1(protein)
```

see *GetQuasiSequenceOrder()* for the choice of parameters.

propy.QuasiSequenceOrder.**GetQuasiSequenceOrder1Grant**(*ProteinSequence: str, maxlag: int = 30, weight: float = 0.1, distancematrix={'AA': 0, 'AC': 195, 'AD': 126, 'AE': 107, 'AF': 113, 'AG': 60, 'AH': 86, 'AI': 94, 'AK': 106, 'AL': 96, 'AM': 84, 'AN': 111, 'AP': 27, 'AQ': 91, 'AR': 112, 'AS': 99, 'AT': 58, 'AV': 64, 'AW': 148, 'AY': 112, 'CA': 195, 'CC': 0, 'CD': 154, 'CE': 170, 'CF': 205, 'CG': 159, 'CH': 174, 'CI': 198, 'CK': 202, 'CL': 198, 'CM': 196, 'CN': 139, 'CP': 169, 'CQ': 154, 'CR': 180, 'CS': 112, 'CT': 149, 'CV': 192, 'CW': 215, 'CY': 194, 'DA': 126, 'DC': 154, 'DD': 0, 'DE': 45, 'DF': 177, 'DG': 94, 'DH': 81, 'DI': 168, 'DK': 101, 'DL': 172, 'DM': 160, 'DN': 23, 'DP': 108, 'DQ': 61, 'DR': 96, 'DS': 65, 'DT': 85, 'DV': 152, 'DW': 181, 'DY': 160, 'EA': 107, 'EC': 170, 'ED': 45, 'EE': 0, 'EF': 140, 'EG': 98, 'EH': 40, 'EI': 134, 'EK': 56, 'EL': 138, 'EM': 126, 'EN': 42, 'EP': 93, 'EQ': 29, 'ER': 54, 'ES': 80, 'ET': 65, 'EV': 121, 'EW': 152, 'EY': 122, 'FA': 113, 'FC': 205, 'FD': 177, 'FE': 140, 'FF': 0, 'FG': 153, 'FH': 100, 'FI': 21, 'FK': 102, 'FL': 22, 'FM': 28, 'FN': 158, 'FP': 114, 'FQ': 116, 'FR': 97, 'FS': 155, 'FT': 103, 'FV': 50, 'FW': 40, 'FY': 22, 'GA': 60, 'GC': 159, 'GD': 94, 'GE': 98, 'GF': 153, 'GG': 0, 'GH': 98, 'GI': 135, 'GK': 127, 'GL': 138, 'GM': 127, 'GN': 80, 'GP': 42, 'GQ': 87, 'GR': 125, 'GS': 56, 'GT': 59, 'GV': 109, 'GW': 184, 'GY': 147, 'HA': 86, 'HC': 174, 'HD': 81, 'HE': 40, 'HF': 100, 'HG': 98, 'HH': 0, 'HI': 94, 'HK': 32, 'HL': 99, 'HM': 87, 'HN': 68, 'HP': 77, 'HQ': 24, 'HR': 29, 'HS': 89, 'HT': 47, 'HV': 84, 'HW': 115, 'HY': 83, 'IA': 94, 'IC': 198, 'ID': 168, 'IE': 134, 'IF': 21, 'IG': 135, 'IH': 94, 'II': 0, 'IK': 102, 'IL': 5, 'IM': 10, 'IN': 149, 'IP': 95, 'IQ': 109, 'IR': 97, 'IS': 142, 'IT': 89, 'IV': 29, 'IW': 61, 'IY': 33, 'KA': 106, 'KC': 202,*

Compute the first 20 quasi-sequence-order descriptors for a given protein sequence.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **maxlag** (*int, optional (default: 30)*) – the maximum lag and the length of the protein should be larger

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetQuasiSequenceOrder1Grant(protein)
```

see *GetQuasiSequenceOrder()* for the choice of parameters.

propy.QuasiSequenceOrder.**GetQuasiSequenceOrder1SW**(*ProteinSequence: str, maxlag=30, weight=0.1, distancematrix={'AA': 0.0, 'AC': 0.112, 'AD': 0.819, 'AE': 0.827, 'AF': 0.54, 'AG': 0.208, 'AH': 0.696, 'AI': 0.407, 'AK': 0.891, 'AL': 0.406, 'AM': 0.379, 'AN': 0.318, 'AP': 0.191, 'AQ': 0.372, 'AR': 1.0, 'AS': 0.094, 'AT': 0.22, 'AV': 0.273, 'AW': 0.739, 'AY': 0.552, 'CA': 0.114, 'CC': 0.0, 'CD': 0.847, 'CE': 0.838, 'CF': 0.437, 'CG': 0.32, 'CH': 0.66, 'CI': 0.304, 'CK': 0.887, 'CL': 0.301, 'CM': 0.277, 'CN': 0.324, 'CP': 0.157, 'CQ': 0.341, 'CR': 1.0, 'CS': 0.176, 'CT': 0.233, 'CV': 0.167, 'CW': 0.639, 'CY': 0.457, 'DA': 0.729, 'DC': 0.742, 'DD': 0.0, 'DE': 0.124, 'DF': 0.924, 'DG': 0.697, 'DH': 0.435, 'DI': 0.847, 'DK': 0.249, 'DL': 0.841, 'DM': 0.819, 'DN': 0.56, 'DP': 0.657, 'DQ': 0.584, 'DR': 0.295, 'DS': 0.667, 'DT': 0.649, 'DV': 0.797, 'DW': 1.0, 'DY': 0.836, 'EA': 0.79, 'EC': 0.788, 'ED': 0.133, 'EE': 0.0, 'EF': 0.932, 'EG': 0.779, 'EH': 0.406, 'EI': 0.86, 'EK': 0.143, 'EL': 0.854, 'EM': 0.83, 'EN': 0.599, 'EP': 0.688, 'EQ': 0.598, 'ER': 0.234, 'ES': 0.726, 'ET': 0.682, 'EV': 0.824, 'EW': 1.0, 'EY': 0.837, 'FA': 0.508, 'FC': 0.405, 'FD': 0.977, 'FE': 0.918, 'FF': 0.0, 'FG': 0.69, 'FH': 0.663, 'FI': 0.128, 'FK': 0.903, 'FL': 0.131, 'FM': 0.169, 'FN': 0.541, 'FP': 0.42, 'FQ': 0.459, 'FR': 1.0, 'FS': 0.548, 'FT': 0.499, 'FV': 0.252, 'FW': 0.207, 'FY': 0.179, 'GA': 0.206, 'GC': 0.312, 'GD': 0.776, 'GE': 0.807, 'GF': 0.727, 'GG': 0.0, 'GH': 0.769, 'GI': 0.592, 'GK': 0.894, 'GL': 0.591, 'GM': 0.557, 'GN': 0.381, 'GP': 0.323, 'GQ': 0.467, 'GR': 1.0, 'GS': 0.158, 'GT': 0.272, 'GV': 0.464, 'GW': 0.923, 'GY': 0.728, 'HA': 0.896, 'HC': 0.836, 'HD': 0.629, 'HE': 0.547, 'HF': 0.907, 'HG': 1.0, 'HH': 0.0, 'HI': 0.848, 'HK': 0.566, 'HL': 0.842, 'HM': 0.825, 'HN': 0.754, 'HP': 0.777, 'HQ': 0.716, 'HR': 0.697, 'HS': 0.865, 'HT': 0.834, 'HV': 0.831, 'HW': 0.981, 'HY': 0.821, 'IA': 0.403, 'IC': 0.296, 'ID': 0.942, 'IE': 0.891, 'IF': 0.134, 'IG': 0.592, 'IH': 0.652, 'II': 0.0, 'IK': 0.892, 'IL': 0.013, 'IM': 0.057, 'IN': 0.457, 'IP': 0.311, 'IQ': 0.383,*

Compute the first 20 quasi-sequence-order descriptors for a given protein sequence.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **maxlag** (*int, optional (default: 30)*) – the maximum lag and the length of the protein should be larger

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetQuasiSequenceOrder1SW(protein)
```

see *GetQuasiSequenceOrder()* for the choice of parameters.

propy.QuasiSequenceOrder.**GetQuasiSequenceOrder2**(*ProteinSequence: str, maxlag=30, weight=0.1, distancematrix=None*)

Compute the last maxlag quasi-sequence-order descriptors for a given protein sequence.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **maxlag** (*int, optional (default: 30)*) – the maximum lag and the length of the protein should be larger

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetQuasiSequenceOrder2(protein)
```

see *GetQuasiSequenceOrder()* for the choice of parameters.

propy.QuasiSequenceOrder.**GetQuasiSequenceOrder2Grant**(*ProteinSequence: str, maxlag: int = 30, weight: float = 0.1, distancematrix={'AA': 0, 'AC': 195, 'AD': 126, 'AE': 107, 'AF': 113, 'AG': 60, 'AH': 86, 'AI': 94, 'AK': 106, 'AL': 96, 'AM': 84, 'AN': 111, 'AP': 27, 'AQ': 91, 'AR': 112, 'AS': 99, 'AT': 58, 'AV': 64, 'AW': 148, 'AY': 112, 'CA': 195, 'CC': 0, 'CD': 154, 'CE': 170, 'CF': 205, 'CG': 159, 'CH': 174, 'CI': 198, 'CK': 202, 'CL': 198, 'CM': 196, 'CN': 139, 'CP': 169, 'CQ': 154, 'CR': 180, 'CS': 112, 'CT': 149, 'CV': 192, 'CW': 215, 'CY': 194, 'DA': 126, 'DC': 154, 'DD': 0, 'DE': 45, 'DF': 177, 'DG': 94, 'DH': 81, 'DI': 168, 'DK': 101, 'DL': 172, 'DM': 160, 'DN': 23, 'DP': 108, 'DQ': 61, 'DR': 96, 'DS': 65, 'DT': 85, 'DV': 152, 'DW': 181, 'DY': 160, 'EA': 107, 'EC': 170, 'ED': 45, 'EE': 0, 'EF': 140, 'EG': 98, 'EH': 40, 'EI': 134, 'EK': 56, 'EL': 138, 'EM': 126, 'EN': 42, 'EP': 93, 'EQ': 29, 'ER': 54, 'ES': 80, 'ET': 65, 'EV': 121, 'EW': 152, 'EY': 122, 'FA': 113, 'FC': 205, 'FD': 177, 'FE': 140, 'FF': 0, 'FG': 153, 'FH': 100, 'FI': 21, 'FK': 102, 'FL': 22, 'FM': 28, 'FN': 158, 'FP': 114, 'FQ': 116, 'FR': 97, 'FS': 155, 'FT': 103, 'FV': 50, 'FW': 40, 'FY': 22, 'GA': 60, 'GC': 159, 'GD': 94, 'GE': 98, 'GF': 153, 'GG': 0, 'GH': 98, 'GI': 135, 'GK': 127, 'GL': 138, 'GM': 127, 'GN': 80, 'GP': 42, 'GQ': 87, 'GR': 125, 'GS': 56, 'GT': 59, 'GV': 109, 'GW': 184, 'GY': 147, 'HA': 86, 'HC': 174, 'HD': 81, 'HE': 40, 'HF': 100, 'HG': 98, 'HH': 0, 'HI': 94, 'HK': 32, 'HL': 99, 'HM': 87, 'HN': 68, 'HP': 77, 'HQ': 24, 'HR': 29, 'HS': 89, 'HT': 47, 'HV': 84, 'HW': 115, 'HY': 83, 'IA': 94, 'IC': 198, 'ID': 168, 'IE': 134, 'IF': 21, 'IG': 135, 'IH': 94, 'II': 0, 'IK': 102, 'IL': 5, 'IM': 10, 'IN': 149, 'IP': 95, 'IQ': 109, 'IR': 97, 'IS': 142, 'IT': 89, 'IV': 29, 'IW': 61, 'IY': 33, 'KA': 106, 'KC': 202,*

Compute the last maxlag quasi-sequence-order descriptors for a given protein sequence.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **maxlag** (*int, optional (default: 30)*) – the maximum lag and the length of the protein should be larger

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetQuasiSequenceOrder2Grant(protein)
```

see *GetQuasiSequenceOrder()* for the choice of parameters.

propy.QuasiSequenceOrder.**GetQuasiSequenceOrder2SW**(*ProteinSequence: str, maxlag=30, weight=0.1, distancematrix={'AA': 0.0, 'AC': 0.112, 'AD': 0.819, 'AE': 0.827, 'AF': 0.54, 'AG': 0.208, 'AH': 0.696, 'AI': 0.407, 'AK': 0.891, 'AL': 0.406, 'AM': 0.379, 'AN': 0.318, 'AP': 0.191, 'AQ': 0.372, 'AR': 1.0, 'AS': 0.094, 'AT': 0.22, 'AV': 0.273, 'AW': 0.739, 'AY': 0.552, 'CA': 0.114, 'CC': 0.0, 'CD': 0.847, 'CE': 0.838, 'CF': 0.437, 'CG': 0.32, 'CH': 0.66, 'CI': 0.304, 'CK': 0.887, 'CL': 0.301, 'CM': 0.277, 'CN': 0.324, 'CP': 0.157, 'CQ': 0.341, 'CR': 1.0, 'CS': 0.176, 'CT': 0.233, 'CV': 0.167, 'CW': 0.639, 'CY': 0.457, 'DA': 0.729, 'DC': 0.742, 'DD': 0.0, 'DE': 0.124, 'DF': 0.924, 'DG': 0.697, 'DH': 0.435, 'DI': 0.847, 'DK': 0.249, 'DL': 0.841, 'DM': 0.819, 'DN': 0.56, 'DP': 0.657, 'DQ': 0.584, 'DR': 0.295, 'DS': 0.667, 'DT': 0.649, 'DV': 0.797, 'DW': 1.0, 'DY': 0.836, 'EA': 0.79, 'EC': 0.788, 'ED': 0.133, 'EE': 0.0, 'EF': 0.932, 'EG': 0.779, 'EH': 0.406, 'EI': 0.86, 'EK': 0.143, 'EL': 0.854, 'EM': 0.83, 'EN': 0.599, 'EP': 0.688, 'EQ': 0.598, 'ER': 0.234, 'ES': 0.726, 'ET': 0.682, 'EV': 0.824, 'EW': 1.0, 'EY': 0.837, 'FA': 0.508, 'FC': 0.405, 'FD': 0.977, 'FE': 0.918, 'FF': 0.0, 'FG': 0.69, 'FH': 0.663, 'FI': 0.128, 'FK': 0.903, 'FL': 0.131, 'FM': 0.169, 'FN': 0.541, 'FP': 0.42, 'FQ': 0.459, 'FR': 1.0, 'FS': 0.548, 'FT': 0.499, 'FV': 0.252, 'FW': 0.207, 'FY': 0.179, 'GA': 0.206, 'GC': 0.312, 'GD': 0.776, 'GE': 0.807, 'GF': 0.727, 'GG': 0.0, 'GH': 0.769, 'GI': 0.592, 'GK': 0.894, 'GL': 0.591, 'GM': 0.557, 'GN': 0.381, 'GP': 0.323, 'GQ': 0.467, 'GR': 1.0, 'GS': 0.158, 'GT': 0.272, 'GV': 0.464, 'GW': 0.923, 'GY': 0.728, 'HA': 0.896, 'HC': 0.836, 'HD': 0.629, 'HE': 0.547, 'HF': 0.907, 'HG': 1.0, 'HH': 0.0, 'HI': 0.848, 'HK': 0.566, 'HL': 0.842, 'HM': 0.825, 'HN': 0.754, 'HP': 0.777, 'HQ': 0.716, 'HR': 0.697, 'HS': 0.865, 'HT': 0.834, 'HV': 0.831, 'HW': 0.981, 'HY': 0.821, 'IA': 0.403, 'IC': 0.296, 'ID': 0.942, 'IE': 0.891, 'IF': 0.134, 'IG': 0.592, 'IH': 0.652, 'II': 0.0, 'IK': 0.892, 'IL': 0.013, 'IM': 0.057, 'IN': 0.457, 'IP': 0.311, 'IQ': 0.383,*

Compute the last maxlag quasi-sequence-order descriptors for a given protein sequence.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **maxlag** (*int, optional (default: 30)*) – the maximum lag and the length of the protein should be larger

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetQuasiSequenceOrder2SW(protein)
```

see *GetQuasiSequenceOrder()* for the choice of parameters.

propy.QuasiSequenceOrder.**GetQuasiSequenceOrderp**(*ProteinSequence: str*, *maxlag: int = 30*, *weight: float = 0.1*, *distancematrix: Dict[Any, Any] = None*) → Dict[Any, Any]

Compute quasi-sequence-order descriptors for a given protein.

See[1] for details.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **maxlag** (*int, optional (default: 30)*) – the maximum lag and the length of the protein should be larger than maxlag
- **weight** (*float, optional (default: 0.1)*) – a weight factor. Please see reference 1 for its choice.
- **distancematrix** (*Dict[Any, Any]*) – contains 400 distance values

Returns **result** – contains all quasi-sequence-order descriptors

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetQuasiSequenceOrderp(protein)
```

propy.QuasiSequenceOrder.**GetSequenceOrderCouplingNumber**(*ProteinSequence: str, d: int = 1, distancematrix: Dict[str, float] = {'AA': 0.0, 'AC': 0.112, 'AD': 0.819, 'AE': 0.827, 'AF': 0.54, 'AG': 0.208, 'AH': 0.696, 'AI': 0.407, 'AK': 0.891, 'AL': 0.406, 'AM': 0.379, 'AN': 0.318, 'AP': 0.191, 'AQ': 0.372, 'AR': 1.0, 'AS': 0.094, 'AT': 0.22, 'AV': 0.273, 'AW': 0.739, 'AY': 0.552, 'CA': 0.114, 'CC': 0.0, 'CD': 0.847, 'CE': 0.838, 'CF': 0.437, 'CG': 0.32, 'CH': 0.66, 'CI': 0.304, 'CK': 0.887, 'CL': 0.301, 'CM': 0.277, 'CN': 0.324, 'CP': 0.157, 'CQ': 0.341, 'CR': 1.0, 'CS': 0.176, 'CT': 0.233, 'CV': 0.167, 'CW': 0.639, 'CY': 0.457, 'DA': 0.729, 'DC': 0.742, 'DD': 0.0, 'DE': 0.124, 'DF': 0.924, 'DG': 0.697, 'DH': 0.435, 'DI': 0.847, 'DK': 0.249, 'DL': 0.841, 'DM': 0.819, 'DN': 0.56, 'DP': 0.657, 'DQ': 0.584, 'DR': 0.295, 'DS': 0.667, 'DT': 0.649, 'DV': 0.797, 'DW': 1.0, 'DY': 0.836, 'EA': 0.79, 'EC': 0.788, 'ED': 0.133, 'EE': 0.0, 'EF': 0.932, 'EG': 0.779, 'EH': 0.406, 'EI': 0.86, 'EK': 0.143, 'EL': 0.854, 'EM': 0.83, 'EN': 0.599, 'EP': 0.688, 'EQ': 0.598, 'ER': 0.234, 'ES': 0.726, 'ET': 0.682, 'EV': 0.824, 'EW': 1.0, 'EY': 0.837, 'FA': 0.508, 'FC': 0.405, 'FD': 0.977, 'FE': 0.918, 'FF': 0.0, 'FG': 0.69, 'FH': 0.663, 'FI': 0.128, 'FK': 0.903, 'FL': 0.131, 'FM': 0.169, 'FN': 0.541, 'FP': 0.42, 'FQ': 0.459, 'FR': 1.0, 'FS': 0.548, 'FT': 0.499, 'FV': 0.252, 'FW': 0.207, 'FY': 0.179, 'GA': 0.206, 'GC': 0.312, 'GD': 0.776, 'GE': 0.807, 'GF': 0.727, 'GG': 0.0, 'GH': 0.769, 'GI': 0.592, 'GK': 0.894, 'GL': 0.591, 'GM': 0.557, 'GN': 0.381, 'GP': 0.323, 'GQ': 0.467, 'GR':*

Compute the dth-rank sequence order coupling number for a protein.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **d** (*int*) – the gap between two amino acids.
- **distancematrix** (*Dict[str, float]*) –

Returns tau

Return type float

Example

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetSequenceOrderCouplingNumber(protein)
```

propy.QuasiSequenceOrder.**GetSequenceOrderCouplingNumberGrant**(*ProteinSequence: str*, *maxlag: int = 30*, *distance-matrix={'AA': 0, 'AC': 195, 'AD': 126, 'AE': 107, 'AF': 113, 'AG': 60, 'AH': 86, 'AI': 94, 'AK': 106, 'AL': 96, 'AM': 84, 'AN': 111, 'AP': 27, 'AQ': 91, 'AR': 112, 'AS': 99, 'AT': 58, 'AV': 64, 'AW': 148, 'AY': 112, 'CA': 195, 'CC': 0, 'CD': 154, 'CE': 170, 'CF': 205, 'CG': 159, 'CH': 174, 'CI': 198, 'CK': 202, 'CL': 198, 'CM': 196, 'CN': 139, 'CP': 169, 'CQ': 154, 'CR': 180, 'CS': 112, 'CT': 149, 'CV': 192, 'CW': 215, 'CY': 194, 'DA': 126, 'DC': 154, 'DD': 0, 'DE': 45, 'DF': 177, 'DG': 94, 'DH': 81, 'DI': 168, 'DK': 101, 'DL': 172, 'DM': 160, 'DN': 23, 'DP': 108, 'DQ': 61, 'DR': 96, 'DS': 65, 'DT': 85, 'DV': 152, 'DW': 181, 'DY': 160, 'EA': 107, 'EC': 170, 'ED': 45, 'EE': 0, 'EF': 140, 'EG': 98, 'EH': 40, 'EI': 134, 'EK': 56, 'EL': 138, 'EM': 126, 'EN': 42, 'EP': 93, 'EQ': 29, 'ER': 54, 'ES': 80, 'ET': 65, 'EV': 121, 'EW': 152, 'EY': 122, 'FA': 113, 'FC': 205, 'FD': 177, 'FE': 140, 'FF': 0, 'FG': 153, 'FH': 100, 'FI': 21, 'FK': 102, 'FL': 22, 'FM': 28, 'FN': 158, 'FP':*

Compute the sequence order coupling numbers from 1 to maxlag for a given protein sequence based on the Grantham chemical distance matrix.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **maxlag** (*int, optional (default: 30)*) – the maximum lag and the length of the protein should be larger than maxlag
- **distancematrix** (*Dict[Any, Any]*) – contains Schneider-Wrede physicochemical distance matrix

Returns **Tau** – contains all sequence order coupling numbers based on the Grantham chemical distance matrix

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetSequenceOrderCouplingNumberGrant(protein)
```

propy.QuasiSequenceOrder.**GetSequenceOrderCouplingNumberSW**(*ProteinSequence: str, maxlag: int = 30, distancematrix={'AA': 0.0, 'AC': 0.112, 'AD': 0.819, 'AE': 0.827, 'AF': 0.54, 'AG': 0.208, 'AH': 0.696, 'AI': 0.407, 'AK': 0.891, 'AL': 0.406, 'AM': 0.379, 'AN': 0.318, 'AP': 0.191, 'AQ': 0.372, 'AR': 1.0, 'AS': 0.094, 'AT': 0.22, 'AV': 0.273, 'AW': 0.739, 'AY': 0.552, 'CA': 0.114, 'CC': 0.0, 'CD': 0.847, 'CE': 0.838, 'CF': 0.437, 'CG': 0.32, 'CH': 0.66, 'CI': 0.304, 'CK': 0.887, 'CL': 0.301, 'CM': 0.277, 'CN': 0.324, 'CP': 0.157, 'CQ': 0.341, 'CR': 1.0, 'CS': 0.176, 'CT': 0.233, 'CV': 0.167, 'CW': 0.639, 'CY': 0.457, 'DA': 0.729, 'DC': 0.742, 'DD': 0.0, 'DE': 0.124, 'DF': 0.924, 'DG': 0.697, 'DH': 0.435, 'DI': 0.847, 'DK': 0.249, 'DL': 0.841, 'DM': 0.819, 'DN': 0.56, 'DP': 0.657, 'DQ': 0.584, 'DR': 0.295, 'DS': 0.667, 'DT': 0.649, 'DV': 0.797, 'DW': 1.0, 'DY': 0.836, 'EA': 0.79, 'EC': 0.788, 'ED': 0.133, 'EE': 0.0, 'EF': 0.932, 'EG': 0.779, 'EH': 0.406, 'EI': 0.86, 'EK': 0.143, 'EL': 0.854, 'EM': 0.83, 'EN': 0.599, 'EP': 0.688, 'EQ': 0.598, 'ER': 0.234, 'ES': 0.726, 'ET': 0.682, 'EV': 0.824, 'EW': 1.0, 'EY': 0.837, 'FA': 0.508, 'FC': 0.405, 'FD': 0.977, 'FE': 0.918, 'FF': 0.0, 'FG': 0.69, 'FH': 0.663, 'FI': 0.128, 'FK': 0.903, 'FL': 0.131, 'FM': 0.169, 'FN': 0.541, 'FP':*

Compute the sequence order coupling numbers from 1 to maxlag for a given protein sequence based on the Schneider-Wrede physicochemical distance matrix.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **maxlag** (*int, optional (default: 30)*) – the maximum lag and the length of the protein should be larger than maxlag
- **distancematrix** (*Dict[Any, Any]*) – contains Schneider-Wrede physicochemical distance matrix

Returns **Tau** – contains all sequence order coupling numbers based on the Schneider-Wrede physicochemical distance matrix

Return type Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetSequenceOrderCouplingNumberSW(protein)
```

propy.QuasiSequenceOrder.**GetSequenceOrderCouplingNumberTotal**(*ProteinSequence: str, maxlag: int = 30*) → Dict[Any, Any]

Compute the sequence order coupling numbers from 1 to maxlag for a given protein sequence.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **maxlag** (*int, optional (default: 30)*) – the maximum lag and the length of the protein should be larger

Returns **result** – contains all sequence order coupling numbers

Return type Dict

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetSequenceOrderCouplingNumberTotal(protein)
```

propy.QuasiSequenceOrder.**GetSequenceOrderCouplingNumberp**(*ProteinSequence: str, maxlag: int = 30, distancematrix: Dict[Any, Any] = None*)

Compute the sequence order coupling numbers from 1 to maxlag for a given protein sequence based on the user-defined property.

Parameters

- **ProteinSequence** (*str*) – a pure protein sequence
- **maxlag** (*int, optional (default: 30)*) – the maximum lag and the length of the protein should be larger than maxlag.

- **distancematrix** (*Dict[Any, Any]*) – contains 400 distance values

Returns **Tau** – contains all sequence order coupling numbers based on the given property

Return type Dict[str]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetSequenceOrderCouplingNumberp(protein)
```

CHAPTER 15

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

E

G

I

M

N

P

R

S

V