User Guide for propy 1.0

1.1 What is this?

This document is intended to provide an overview of how one can use the propy functionality from Python. It's not comprehensive and it's not a manual.
If you find mistakes, or have suggestions for improvements, please either fix them yourselves in the source document (the .py file) or send them to the mailing list: orietal-cds@hotmail.com

1.2 Install the propy package

propy has been successfully tested on Linux and Windows systems. The author could download the propy package from http://code.google.com/p/protpy/downloads/list (.zip and .tar.gz). The install process of propy is very easy:

On Windows:

(1): download the propy package (.zip)

(2): extract or uncompress the .zip file

(3): cd propy-1.0

(4): python setup.py install

On Linux:

(1): download the propy package (.tar.gz)

(2): tar -zxf propy-1.0.tar.gz

(3): cd propy-1.0

(4): python setup.py install or sudo python setup.py install

1.3 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
MQGNGSALPNASQPVLRGDGARPSWLASALACVLIFTIVVDILGNLLVILSVYRNKKLRNAGNIFVVSLAVADLVVAIYP
YPLVLMSIFNNGWNLGYLHCQVSGFLMGLSVIGSIFNITGIAINRYCYICHSLKYDKLYSSKNSLCYVLLIWLLTLAAVL
PNLRAGTLQYDPRIYSCTFAQSVSSAYTIAVVVFHFLVPMIIVIFCYLRIWILVLQVRQRVKPDRKPKLKPQDFRNFVTM
FVVFVLFAICWAPLNFIGLAVASDPASMVPRIPEWLFVASYYMAYFNSCLNAIIYGLLNQNFRKEYRRIIVSLCTARVFF
VDSSNDVADRVKWKPSPLMTNNNVVKVDSV
```

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
['MQGNGSALPNA', 'ALPNASQPVLR', 'DGARPSWLASA', 'PSWLASALACV', 'LLVILSVYRNK', 'NIF
VVSLAVAD', 'PLVLMSIFNNG', 'LHCQVSGFLMG', 'FLMGLSVIGSI', 'LSVIGSIFNIT', 'CYICHSLK
YDK', 'YDKLYSSKNSL', 'DKLYSSKNSLC', 'YSSKNSLCYVL', 'DPRIYSCTFAQ', 'CTFAQSVSSAY',
 'FAQSVSSAYTI', 'AQSVSSAYTIA', 'GLAVASDPASM', 'ASDPASMVPRI', 'WLFVASYYMAY', 'MAY
FNSCLNAI', 'RRIIVSLCTAR', 'VFFVDSSNDVA', 'FFVDSSNDVAD', 'VKWKPSPLMTN']
>>> print len(subseq)
26
```

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

```
>>> tag=gpst("F:\\","target.txt","target1.txt")
-------------------------------------------------------
The 1 protein sequence has been downloaded!
MADSCRNLTYVRGSVGPATSTLMFVAGVVGNGLALGILSARRPARPSAFAVLVTGLAATDLLGTSFLSPAVFVAYARNSS
LLGLARGGPALCDAFAFAMTFFGLASMLILFAMAVERCLALSHPYLYAQLDGPRCARLALPAIYAFCVLFCALPLLGLGQ
HQQYCPGSWCFLRMRWAQPGGAAFSLAYAGLVALLVAAIFLCNGSVTLSLCRMYRQQKRHQGSLGPRPRTGEDEVDHLIL
LALMTVVMAVCSLPLTIRCFTQAVAPDSSSEMGDLLAFRFYAFNPILDPWVFILFRKAVFQRLKLWVCCLCLGPAHGDSQ
TPLSQLASGRRDPRAPSAPVGKEGSCVPLSAWGEGQVEPLPPTQQSSGSAVGTSSKAEASVACSLC
-------------------------------------------------------
-------------------------------------------------------
The 2 protein sequence has been downloaded!
MPNNSTALSLANVTYITMEIFIGLCAIVGNVLVICVVKLNPSLQTTTFYFIVSLALADIAVGVLVMPLAIVVSLGITIHF
YSCLFMTCLLLIFTHASIMSLLAIAVDRYLRVKLTVRYKRVTTHRRIWLALGLCWLVSFLVGLTPMFGWNMKLTSEYHRN
VTFLSCQFVSVMRMDYMVYFSFLTWIFIPLVVMCAIYLDIFYIIRNKLSLNLSNSKETGAFYGREFKTAKSLFLVLFLFA
LSWLPLSIINCIIYFNGEVPQLVLYMGILLSHANSMMNPIVYAYKIKKFKETYLLILKACVVCHPSDSLDTSIEKNSE
-------------------------------------------------------
-------------------------------------------------------
The 3 protein sequence has been downloaded!
MQGNGSALPNASQPVLRGDGARPSWLASALACVLIFTIVVDILGNLLVILSVYRNKKLRNAGNIFVVSLAVADLVVAIYP
YPLVLMSIFNNGWNLGYLHCQVSGFLMGLSVIGSIFNITGIAINRYCYICHSLKYDKLYSSKNSLCYVLLIWLLTLAAVL
PNLRAGTLQYDPRIYSCTFAQSVSSAYTIAVVVFHFLVPMIIVIFCYLRIWILVLQVRQRVKPDRKPKLKPQDFRNFVTM
FVVFVLFAICWAPLNFIGLAVASDPASMVPRIPEWLFVASYYMAYFNSCLNAIIYGLLNQNFRKEYRRIIVSLCTARVFF
VDSSNDVADRVKWKPSPLMTNNNVVKVDSV
-------------------------------------------------------
>>> print tag
0
```

The downloaded protein sequences have been saved in "F:/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 temp
350
```

The output is the number of the protein sequence if it is valid; otherwise 0.

1.4 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.

```
>>> import propy
>>> from propy import AAIndex
>>> proindex=AAIndex.GetAAIndex1('KRIW790103',path=propy.__path__[0])
>>> print proindex
{'A': 27.5, 'C': 44.6, 'E': 62.0, 'D': 40.0, 'G': 0.0, 'F': 115.5, 'I': 93.5, 'H
': 79.0, 'K': 100.0, 'M': 94.1, 'L': 93.5, 'N': 58.7, 'Q': 80.7, 'P': 41.9, 'S':
 29.3, 'R': 105.0, 'T': 51.3, 'W': 145.5, 'V': 71.5, 'Y': 117.3}
```

It should be noted that the propy package has contained the AAindex database. The GetAAIndex1 methods in AAIndex will get the property from the aaindex1 database.

If the user does not provide the directory containing the AAindex database, the program will downlaod the three databases (i.e., aaindex1, aaindex2 and aaindex3) to obtain the property. It should be noted that the downloaded AAindex will be saved in the current directory. You can also specify the directory according to your needs.

```
>>> proindex=AAIndex.GetAAIndex23('GRAR740104',path='F:')
>>> print len(proindex)
400
```

The downloaded databases are saved in F disk. The GetAAIndex23 methods in AAIndex will get the property from the aaindex2 and aaindex3 databases.

1.5 Calculating protein descriptors

There are two ways to calculate protein descriptors in the propy package. One is to directly use the corresponding methods, the other one is firstly to construct a GetProDes class and then run their methods to obtain the protein descriptors. It should be noted that the output is a dictionary form, whose keys and values represent the descriptor name and the descriptor value, respectively. The user could clearly understand the meaning of each descriptor.

Use functions:

```
>>> from propy import AAComposition as AAC
>>> print AAC.CalculateAAComposition
<function CalculateAAComposition at 0x0000000003146908>
>>> print AAC.CalculateAAComposition(proseq)
{'A': 7.714, 'C': 2.857, 'E': 0.571, 'D': 3.429, 'G': 4.286, 'F': 5.714, 'I': 8.
0, 'H': 0.857, 'K': 3.714, 'M': 2.286, 'L': 12.286, 'N': 6.571, 'Q': 2.571, 'P':
 4.857, 'S': 7.714, 'R': 5.143, 'T': 2.571, 'W': 2.0, 'V': 11.714, 'Y': 5.143}
>>> from propy import CTD
>>> ctd=CTD.CalculateC(proseq)
>>> print ctd
{'_NormalizedVDWVC2': 0.417, '_PolarizabilityC2': 0.494, '_PolarizabilityC3': 0.
249, '_ChargeC1': 0.089, '_PolarizabilityC1': 0.257, '_SecondaryStrC2': 0.38, '_
SecondaryStrC3': 0.269, '_NormalizedVDWVC3': 0.249, '_SecondaryStrC1': 0.351, '_
SolventAccessibilityC1': 0.546, '_SolventAccessibilityC2': 0.22, '_SolventAccess
ibilityC3': 0.234, '_NormalizedVDWVC1': 0.306, '_HydrophobicityC3': 0.449, '_Hyd
rophobicityC1': 0.22, '_ChargeC3': 0.04, '_PolarityC2': 0.146, '_PolarityC1': 0.
5, '_HydrophobicityC2': 0.331, '_PolarityC3': 0.097, '_ChargeC2': 0.871}
>>> ctd=CTD.CalculateT(proseq)
>>> print ctd
{'_SecondaryStrT13': 0.212, '_SecondaryStrT12': 0.261, '_SolventAccessibilityT23
': 0.1, '_SecondaryStrT23': 0.149, '_HydrophobicityT13': 0.16, '_HydrophobicityT
12': 0.149, '_HydrophobicityT23': 0.292, '_NormalizedVDWVT23': 0.232, '_ChargeT1
2': 0.143, '_ChargeT13': 0.011, '_NormalizedVDWVT13': 0.132, '_PolarityT12': 0.1
12, '_PolarizabilityT23': 0.289, '_ChargeT23': 0.069, '_PolarityT23': 0.049, '_N
ormalizedVDWVT12': 0.261, '_SolventAccessibilityT13': 0.246, '_SolventAccessibil
ityT12': 0.209, '_PolarizabilityT13': 0.097, '_PolarizabilityT12': 0.258, '_Pola
rityT13': 0.097}
>>> ctd=CTD.CalculateCTD(proseq)
>>> for i in ctd:
        print i,  ctd[i]


_NormalizedVDWVD1075 75.143
_PolarityD1075 71.714
_SecondaryStrD3025 19.429
_PolarityD3100 98.857
 ChargeD1100 98.857
```

Use GetProDes class:

```
>>> from propy.PyPro import GetProDes
>>> print GetProDes.AALetter
['A', 'R', 'N', 'D', 'C', 'E', 'Q', 'G', 'H', 'I', 'L', 'K', 'M', 'F', 'P', 'S',
 'T', 'W', 'Y', 'V']
```

Example 1: Calculating amino acid composition descriptors

```
>>> from propy.PyPro import GetProDes
>>> Des=GetProDes(proseq)
>>> Des.GetAAComp
<bound method GetProDes.GetAAComp of <propy.PyPro.GetProDes instance at 0x000000
00032B2408>>
>>> print Des.GetAAComp()
{'A': 7.714, 'C': 2.857, 'E': 0.571, 'D': 3.429, 'G': 4.286, 'F': 5.714, 'I': 8.
0, 'H': 0.857, 'K': 3.714, 'M': 2.286, 'L': 12.286, 'N': 6.571, 'Q': 2.571, 'P':
 4.857, 'S': 7.714, 'R': 5.143, 'T': 2.571, 'W': 2.0, 'V': 11.714, 'Y': 5.143}
```

Example 2: Calculating Geary autocorrelation descriptors

```
>>> geary=Des.GetGearyAuto()
>>> for i in geary:
        print i,  geary[i]


GearyAuto_Mutability28 0.941
GearyAuto_Mutability29 1.001
GearyAuto_Mutability26 0.912
GearyAuto_Mutability27 0.965
GearyAuto_Mutability24 0.919
GearyAuto_Mutability25 1.002
GearyAuto_Mutability22 0.995
```

Example 3: Calculating pseudo amino acid composition descriptors

```
>>> PAAC=Des.GetPAAC(lamda=5,weight=0.05)
>>> for i in PAAC:
        print i,  PAAC[i]


PAAC24 5.613
PAAC25 5.887
PAAC8 3.043
PAAC9 0.609
PAAC2 3.652
PAAC3 4.666
PAAC1 5.478
PAAC6 0.405
PAAC7 1.826
PAAC4 2.435
PAAC5 2.029
PAAC21 6.071
PAAC20 8.318
PAAC23 5.897
PAAC22 5.521
PAAC18 1.42
PAAC19 3.652
PAAC14 4.058
PAAC15 3.449
PAAC16 5.478
PAAC17 1.826
PAAC10 5.681
PAAC11 8.724
PAAC12 2.637
PAAC13 1.623
```

When we change the values of lamda and weight, we could get different PAAC values. Note that the number of PAAC depends on the choice of lamda. If lamda = 10, we can obtain 20+lamda=30 PAAC descriptors.

```
>>> PAAC=Des.GetPAAC(lamda=10,weight=0.05)
>>> for i in PAAC:
        print i,  PAAC[i]


PAAC29 4.548
PAAC24 4.283
PAAC28 4.787
PAAC25 4.491
PAAC23 4.5
PAAC8 2.322
PAAC9 0.464
PAAC2 2.786
PAAC3 3.56
PAAC1 4.179
PAAC6 0.309
PAAC7 1.393
PAAC4 1.858
PAAC5 1.548
PAAC21 4.632
PAAC20 6.346
PAAC30 4.863
PAAC22 4.212
PAAC18 1.084
PAAC19 2.786
PAAC27 4.681
PAAC26 4.825
PAAC14 3.096
PAAC15 2.631
PAAC16 4.179
PAAC17 1.393
PAAC10 4.334
PAAC11 6.656
PAAC12 2.012
PAAC13 1.239
```

Example 4: Calculating all protein descriptors

The GetProDes class includes a built-in method which can calculate all protein descriptors.

```
>>> alldes=Des.GetALL()
>>> index=5
>>> for i,j in enumerate(alldes.items()):
        if i<index:
                print j[0], j[1]


MoreauBrotoAuto_AvFlexibility19 0.055
MoreauBrotoAuto_AvFlexibility18 0.053
MoreauBrotoAuto_AvFlexibility15 0.054
MoreauBrotoAuto_AvFlexibility14 0.057
MoreauBrotoAuto_AvFlexibility17 0.05
```

Example 5: Calculating protein descriptors based on the user-defined property

The user could provide some property in the form of dictionary in python. Thus, propy could calculate the descriptors based on the user-defined property.

```
>>> from propy import PseudoAAC
>>> Hydrophobicity=PseudoAAC._Hydrophobicity
>>> print Hydrophobicity
{'A': 0.62, 'C': 0.29, 'E': -0.74, 'D': -0.9, 'G': 0.48, 'F': 1.19, 'I': 1.38, '
H': -0.4, 'K': -1.5, 'M': 0.64, 'L': 1.06, 'N': -0.78, 'Q': -0.85, 'P': 0.12, 'S
': -0.18, 'R': -2.53, 'T': -0.05, 'W': 0.81, 'V': 1.08, 'Y': 0.26}
>>> pK1=PseudoAAC._pK1
>>> print pK1
{'A': 2.35, 'C': 1.71, 'E': 2.19, 'D': 1.88, 'G': 2.34, 'F': 2.58, 'I': 2.32, 'H
': 1.78, 'K': 2.2, 'M': 2.28, 'L': 2.36, 'N': 2.18, 'Q': 2.17, 'P': 1.99, 'S': 2
.21, 'R': 2.18, 'T': 2.15, 'W': 2.38, 'V': 2.29, 'Y': 2.2}
>>> residuemass=PseudoAAC._residuemass
>>> print residuemass
{'A': 15.0, 'C': 47.0, 'E': 73.0, 'D': 59.0, 'G': 1.0, 'F': 91.0, 'I': 57.0, 'H'
: 82.0, 'K': 73.0, 'M': 75.0, 'L': 57.0, 'N': 58.0, 'Q': 72.0, 'P': 42.0, 'S': 3
1.0, 'R': 101.0, 'T': 45.0, 'W': 130.0, 'V': 43.0, 'Y': 107.0}
>>> from propy.PyPro import GetProDes
>>> Des=GetProDes(proseq)
>>> PAACp=Des.GetPAACp(lamda=5,weight=0.01,AAP=[Hydrophobicity,pK1])
>>> print PAACp
{'PAAC24': 1.476, 'PAAC25': 1.547, 'PAAC8': 3.964, 'PAAC9': 0.793, 'PAAC2': 4.75
6, 'PAAC3': 6.077, 'PAAC1': 7.134, 'PAAC6': 0.528, 'PAAC7': 2.378, 'PAAC4': 3.17
1, 'PAAC5': 2.642, 'PAAC21': 1.523, 'PAAC20': 10.833, 'PAAC23': 1.463, 'PAAC22':
 1.515, 'PAAC18': 1.85, 'PAAC19': 4.756, 'PAAC14': 5.284, 'PAAC15': 4.492, 'PAAC
16': 7.134, 'PAAC17': 2.378, 'PAAC10': 7.398, 'PAAC11': 11.362, 'PAAC12': 3.435,
 'PAAC13': 2.114}
```

Example 6: Calculating protein descriptors based on the property from AAindex

A powerful ability of propy is that it can easily calculate thousands of protein features through automatically obtaining the needed property from AAindex.

```
>>> from propy.PyPro import GetProDes
>>> import propy
>>> from propy import AAIndex
>>> proindex=AAIndex.GetAAIndex1('KRIW790103',path=propy.__path__[0])
>>> print proindex
{'A': 27.5, 'C': 44.6, 'E': 62.0, 'D': 40.0, 'G': 0.0, 'F': 115.5, 'I': 93.5, 'H
': 79.0, 'K': 100.0, 'M': 94.1, 'L': 93.5, 'N': 58.7, 'Q': 80.7, 'P': 41.9, 'S':
 29.3, 'R': 105.0, 'T': 51.3, 'W': 145.5, 'V': 71.5, 'Y': 117.3}
>>> Des=GetProDes(proseq)
>>> Des.GetGearyAutop
<bound method GetProDes.GetGearyAutop of <propy.PyPro.GetProDes instance at 0x00
000000034C83C8>>
>>> geary=Des.GetGearyAutop(proindex)
>>> for i in geary:
        print i, geary[i]


GearyAutop29 1.076
GearyAutop28 0.965
GearyAutop27 1.015
GearyAutop26 0.87
GearyAutop25 1.02
```

```
>>> paac=Des.GetPAACp(lamda=10,weight=0.05,AAP=[proindex])
>>> for i in paac:
        print i, paac[i]


PAAC29 4.632
PAAC24 4.261
PAAC28 4.63
PAAC25 4.616
PAAC23 5.094
PAAC8 2.274
PAAC9 0.455
PAAC2 2.729
PAAC3 3.487
PAAC1 4.093
PAAC6 0.303
```

Table List of propy computed features for protein sequences

Feature group	Features	Number of descriptors
Amino acid composition	Amino acid composition	20
	Dipeptide composition	400
	Tripeptide composition	8000
Autocorrelation	Normalized Moreau-Broto autocorrelation	240[a]
	Moran autocorrelation	240 [a]
	Geary autocorrelation	240 [a]
CTD	Composition	21
	Transition	21
	Distribution	105
Quasi-sequence order	Sequence order coupling number	60
	Quasi-sequence order descriptors	100
Pseudo amino acid composition	Pseudo amino acid composition	50 [b]
	Amphiphilic pseudo amino acid composition	50[c]

[a] The number depends on the choice of the number of properties of amino acid and the choice of the maximum values of the lag. The default is use eight types of properties and lag = 30.

[b] The number depends on the choice of the number of the set of amino acid properties and the choice of the lamda value. The default is use three types of properties proposed by Chou et al and lamda = 30.

[c] The number depends on the choice of the lamda vlaue. The default is that lamda = 30.