

# COPADS - Collection of Python Algorithms and Data Structures

## API Documentation

June 17, 2013

## Contents

<b>Contents</b>	<b>1</b>
<b>1 Package copads</b>	<b>2</b>
1.1 Modules . . . . .	2
1.2 Variables . . . . .	4
<b>2 Module copads.array</b>	<b>5</b>
2.1 Variables . . . . .	5
2.2 Class ParallelArray . . . . .	5
2.2.1 Methods . . . . .	5
2.2.2 Properties . . . . .	6
<b>3 Module copads.bag</b>	<b>7</b>
3.1 Variables . . . . .	7
3.2 Class Bag . . . . .	7
3.2.1 Methods . . . . .	7
3.2.2 Properties . . . . .	9
<b>4 Module copads.colormap</b>	<b>10</b>
4.1 Functions . . . . .	10
4.2 Variables . . . . .	10
<b>5 Module copads.constants</b>	<b>11</b>
5.1 Variables . . . . .	11
<b>6 Module copads.copadsexceptions</b>	<b>13</b>
6.1 Variables . . . . .	14
6.2 Class CopadsError . . . . .	14
6.2.1 Methods . . . . .	14
6.2.2 Properties . . . . .	14
6.3 Class MatrixError . . . . .	15
6.3.1 Methods . . . . .	15
6.3.2 Properties . . . . .	15
6.4 Class MatrixArithmeticError . . . . .	16
6.4.1 Methods . . . . .	16
6.4.2 Properties . . . . .	17
6.5 Class MatrixMultiplicationError . . . . .	17
6.5.1 Methods . . . . .	17

6.5.2	Properties . . . . .	18
6.6	Class MatrixAdditionError . . . . .	18
6.6.1	Methods . . . . .	18
6.6.2	Properties . . . . .	19
6.7	Class MatrixSquareError . . . . .	19
6.7.1	Methods . . . . .	20
6.7.2	Properties . . . . .	20
6.8	Class MatrixTraceError . . . . .	21
6.8.1	Methods . . . . .	21
6.8.2	Properties . . . . .	21
6.9	Class MatrixMinorError . . . . .	22
6.9.1	Methods . . . . .	22
6.9.2	Properties . . . . .	23
6.10	Class MatrixDeterminantError . . . . .	23
6.10.1	Methods . . . . .	23
6.10.2	Properties . . . . .	24
6.11	Class GraphError . . . . .	24
6.11.1	Methods . . . . .	24
6.11.2	Properties . . . . .	25
6.12	Class EdgeNotFoundError . . . . .	25
6.12.1	Methods . . . . .	25
6.12.2	Properties . . . . .	26
6.13	Class VertexNotFoundError . . . . .	26
6.13.1	Methods . . . . .	26
6.13.2	Properties . . . . .	27
6.14	Class UnknownGraphMatrixError . . . . .	27
6.14.1	Methods . . . . .	28
6.14.2	Properties . . . . .	28
6.15	Class NotAdjacencyGraphMatrixError . . . . .	29
6.15.1	Methods . . . . .	29
6.15.2	Properties . . . . .	30
6.16	Class GraphEdgeSizeMismatchError . . . . .	30
6.16.1	Methods . . . . .	30
6.16.2	Properties . . . . .	31
6.17	Class GraphParameterError . . . . .	31
6.17.1	Methods . . . . .	31
6.17.2	Properties . . . . .	32
6.18	Class StatisticsError . . . . .	32
6.18.1	Methods . . . . .	33
6.18.2	Properties . . . . .	33
6.19	Class DistributionError . . . . .	33
6.19.1	Methods . . . . .	34
6.19.2	Properties . . . . .	34
6.20	Class NormalDistributionTypeError . . . . .	34
6.20.1	Methods . . . . .	35
6.20.2	Properties . . . . .	35
6.21	Class DistributionParameterError . . . . .	36
6.21.1	Methods . . . . .	36
6.21.2	Properties . . . . .	36
6.22	Class DistributionFunctionError . . . . .	37
6.22.1	Methods . . . . .	37

6.22.2	Properties . . . . .	38
6.23	Class DistanceError . . . . .	38
6.23.1	Methods . . . . .	38
6.23.2	Properties . . . . .	39
6.24	Class DistanceInputSizeError . . . . .	39
6.24.1	Methods . . . . .	39
6.24.2	Properties . . . . .	40
6.25	Class TreeError . . . . .	40
6.25.1	Methods . . . . .	40
6.25.2	Properties . . . . .	41
6.26	Class TreeNodeTypeError . . . . .	41
6.26.1	Methods . . . . .	41
6.26.2	Properties . . . . .	42
6.27	Class FunctionParameterTypeError . . . . .	42
6.27.1	Methods . . . . .	42
6.27.2	Properties . . . . .	43
6.28	Class FunctionParameterValueError . . . . .	43
6.28.1	Methods . . . . .	43
6.28.2	Properties . . . . .	44
6.29	Class ArrayError . . . . .	44
6.29.1	Methods . . . . .	45
6.29.2	Properties . . . . .	45
6.30	Class MaxIterationsException . . . . .	46
6.30.1	Methods . . . . .	46
6.30.2	Properties . . . . .	46
<b>7</b>	<b>Module copads.dataframe</b> . . . . .	<b>47</b>
7.1	Variables . . . . .	47
7.2	Class dataframe . . . . .	47
7.2.1	Methods . . . . .	47
7.2.2	Properties . . . . .	49
7.2.3	Class Variables . . . . .	49
<b>8</b>	<b>Module copads.dose_entities</b> . . . . .	<b>50</b>
8.1	Variables . . . . .	50
8.2	Class Organism . . . . .	50
8.2.1	Methods . . . . .	51
8.2.2	Properties . . . . .	52
8.2.3	Class Variables . . . . .	53
8.3	Class Population . . . . .	53
8.3.1	Methods . . . . .	53
8.3.2	Properties . . . . .	55
8.4	Class World . . . . .	56
8.4.1	Methods . . . . .	56
8.4.2	Properties . . . . .	58
8.4.3	Class Variables . . . . .	58
<b>9</b>	<b>Module copads.dose_executor</b> . . . . .	<b>59</b>
9.1	Functions . . . . .	59
9.2	Variables . . . . .	60
<b>10</b>	<b>Module copads.dose_parameters</b> . . . . .	<b>61</b>

10.1 Variables . . . . .	61
<b>11 Module copads.dose_world</b>	<b>64</b>
11.1 Variables . . . . .	64
11.2 Class World . . . . .	64
11.2.1 Methods . . . . .	65
11.2.2 Properties . . . . .	68
11.2.3 Class Variables . . . . .	68
<b>12 Module copads.genetic</b>	<b>69</b>
12.1 Functions . . . . .	69
12.2 Variables . . . . .	71
12.3 Class Chromosome . . . . .	71
12.3.1 Methods . . . . .	72
12.3.2 Properties . . . . .	75
12.4 Class Organism . . . . .	75
12.4.1 Methods . . . . .	77
12.4.2 Properties . . . . .	79
12.4.3 Class Variables . . . . .	80
12.5 Class Population . . . . .	80
12.5.1 Methods . . . . .	81
12.5.2 Properties . . . . .	84
<b>13 Module copads.graph</b>	<b>85</b>
13.1 Variables . . . . .	85
13.2 Class Graph . . . . .	85
13.2.1 Methods . . . . .	85
13.2.2 Class Variables . . . . .	88
<b>14 Module copads.hash</b>	<b>89</b>
14.1 Functions . . . . .	89
14.2 Variables . . . . .	91
<b>15 Module copads.hypothesis</b>	<b>92</b>
15.1 Functions . . . . .	93
15.2 Variables . . . . .	114
<b>16 Module copads.lc_bf</b>	<b>116</b>
16.1 Functions . . . . .	116
16.2 Variables . . . . .	117
<b>17 Module copads.matrix</b>	<b>118</b>
17.1 Functions . . . . .	118
17.2 Variables . . . . .	120
17.3 Class Vector . . . . .	120
17.3.1 Methods . . . . .	121
17.3.2 Properties . . . . .	122
17.3.3 Class Variables . . . . .	122
17.4 Class Matrix . . . . .	123
17.4.1 Methods . . . . .	124
17.4.2 Class Variables . . . . .	127
17.5 Class SparseMatrix . . . . .	127
17.5.1 Methods . . . . .	128

17.5.2 Properties . . . . .	129
17.5.3 Class Variables . . . . .	129
<b>18 Module copads.n_bf</b>	<b>130</b>
18.1 Functions . . . . .	130
18.2 Variables . . . . .	130
<b>19 Module copads.neural</b>	<b>131</b>
19.1 Functions . . . . .	131
19.2 Variables . . . . .	131
19.3 Class Neuron . . . . .	131
19.3.1 Methods . . . . .	132
19.3.2 Class Variables . . . . .	133
19.4 Class Brain . . . . .	134
19.4.1 Methods . . . . .	135
19.4.2 Class Variables . . . . .	137
<b>20 Module copads.nrpv</b>	<b>138</b>
20.1 Functions . . . . .	139
20.2 Variables . . . . .	149
<b>21 Module copads.objectdistance</b>	<b>150</b>
21.1 Functions . . . . .	150
21.2 Variables . . . . .	167
<b>22 Module copads.operations</b>	<b>169</b>
22.1 Functions . . . . .	169
22.2 Variables . . . . .	172
22.3 Class Modulus2 . . . . .	172
22.3.1 Methods . . . . .	172
22.4 Class Boolean . . . . .	172
22.4.1 Methods . . . . .	173
<b>23 Module copads.prioritydictionary</b>	<b>174</b>
23.1 Variables . . . . .	174
23.2 Class PriorityDictionary . . . . .	174
23.2.1 Methods . . . . .	174
23.2.2 Properties . . . . .	175
23.2.3 Class Variables . . . . .	175
<b>24 Module copads.ragaraja</b>	<b>176</b>
24.1 Functions . . . . .	176
24.2 Variables . . . . .	188
<b>25 Module copads.register_machine</b>	<b>189</b>
25.1 Functions . . . . .	189
25.2 Variables . . . . .	190
<b>26 Module copads.ring</b>	<b>191</b>
26.1 Variables . . . . .	191
26.2 Class RingList . . . . .	191
26.2.1 Methods . . . . .	191

<b>27 Module copads.samplestatistics</b>	<b>193</b>
27.1 Variables . . . . .	193
27.2 Class SingleSample . . . . .	193
27.2.1 Methods . . . . .	193
27.2.2 Class Variables . . . . .	195
27.3 Class SampleDistribution . . . . .	195
27.3.1 Methods . . . . .	195
27.4 Class TwoSample . . . . .	195
27.4.1 Methods . . . . .	196
27.4.2 Class Variables . . . . .	196
<b>28 Module copads.set</b>	<b>197</b>
28.1 Variables . . . . .	197
28.2 Class Set . . . . .	197
28.2.1 Methods . . . . .	197
<b>29 Module copads.statisticsdistribution</b>	<b>199</b>
29.1 Functions . . . . .	201
29.2 Variables . . . . .	203
29.3 Class Distribution . . . . .	204
29.3.1 Methods . . . . .	204
29.4 Class BetaDistribution . . . . .	205
29.4.1 Methods . . . . .	206
29.5 Class BinomialDistribution . . . . .	207
29.5.1 Methods . . . . .	208
29.6 Class CauchyDistribution . . . . .	209
29.6.1 Methods . . . . .	209
29.7 Class CosineDistribution . . . . .	211
29.7.1 Methods . . . . .	211
29.8 Class ExponentialDistribution . . . . .	213
29.8.1 Methods . . . . .	213
29.9 Class FDistribution . . . . .	215
29.9.1 Methods . . . . .	216
29.10Class GammaDistribution . . . . .	217
29.10.1 Methods . . . . .	217
29.11Class ChiSquareDistribution . . . . .	218
29.11.1 Methods . . . . .	219
29.12Class GeometricDistribution . . . . .	219
29.12.1 Methods . . . . .	220
29.13Class HypergeometricDistribution . . . . .	221
29.13.1 Methods . . . . .	221
29.14Class LogarithmicDistribution . . . . .	223
29.14.1 Methods . . . . .	223
29.15Class NormalDistribution . . . . .	224
29.15.1 Methods . . . . .	224
29.16Class PoissonDistribution . . . . .	226
29.16.1 Methods . . . . .	227
29.17Class SemicircularDistribution . . . . .	228
29.17.1 Methods . . . . .	228
29.18Class TDistribution . . . . .	230
29.18.1 Methods . . . . .	230
29.19Class TriangularDistribution . . . . .	232

29.19.1 Methods . . . . .	233
29.20Class UniformDistribution . . . . .	234
29.20.1 Methods . . . . .	235
29.21Class WeiBullDistribution . . . . .	236
29.21.1 Methods . . . . .	237
29.22Class BernoulliDistribution . . . . .	238
29.22.1 Methods . . . . .	238
29.23Class BradfordDistribution . . . . .	239
29.23.1 Methods . . . . .	240
29.24Class BurrDistribution . . . . .	241
29.24.1 Methods . . . . .	242
29.25Class ChiDistribution . . . . .	243
29.25.1 Methods . . . . .	243
29.26Class DoubleGammaDistribution . . . . .	243
29.26.1 Methods . . . . .	244
29.27Class DoubleWeibullDistribution . . . . .	245
29.27.1 Methods . . . . .	245
29.28Class ExtremeLBDistribution . . . . .	245
29.28.1 Methods . . . . .	246
29.29Class FiskDistribution . . . . .	246
29.29.1 Methods . . . . .	246
29.30Class FoldedNormalDistribution . . . . .	246
29.30.1 Methods . . . . .	247
29.31Class GenLogisticDistribution . . . . .	247
29.31.1 Methods . . . . .	247
29.32Class GumbelDistribution . . . . .	247
29.32.1 Methods . . . . .	248
29.33Class HalfNormalDistribution . . . . .	249
29.33.1 Methods . . . . .	250
29.34Class HyperbolicSecantDistribution . . . . .	251
29.34.1 Methods . . . . .	251
29.35Class LaplaceDistribution . . . . .	253
29.35.1 Methods . . . . .	253
29.36Class LogisticDistribution . . . . .	253
29.36.1 Methods . . . . .	253
29.37Class LogNormalDistribution . . . . .	254
29.37.1 Methods . . . . .	254
29.38Class MaxwellDistribution . . . . .	254
29.38.1 Methods . . . . .	255
29.39Class NakagamiDistribution . . . . .	256
29.39.1 Methods . . . . .	256
29.40Class NegativeBinomialDistribution . . . . .	256
29.40.1 Methods . . . . .	257
29.41Class ParetoDistribution . . . . .	258
29.41.1 Methods . . . . .	258
29.42Class PascalDistribution . . . . .	260
29.42.1 Methods . . . . .	260
29.43Class PowerFunctionDistribution . . . . .	261
29.43.1 Methods . . . . .	261
29.44Class RademacherDistribution . . . . .	263
29.44.1 Methods . . . . .	263

29.45	Class RayleighDistribution . . . . .	264
29.45.1	Methods . . . . .	264
29.46	Class ReciprocalDistribution . . . . .	265
29.46.1	Methods . . . . .	266
<b>30</b>	<b>Module copads.tree</b>	<b>267</b>
30.1	Variables . . . . .	267
30.2	Class RBTreeIter . . . . .	267
30.2.1	Methods . . . . .	267
30.2.2	Properties . . . . .	268
30.3	Class OrderedBinaryTree . . . . .	268
30.3.1	Methods . . . . .	268
30.4	Class RBTree . . . . .	268
30.4.1	Methods . . . . .	269
30.4.2	Properties . . . . .	270
30.5	Class RBList . . . . .	270
30.5.1	Methods . . . . .	271
30.5.2	Properties . . . . .	272
30.6	Class RBDict . . . . .	272
30.6.1	Methods . . . . .	272
30.6.2	Properties . . . . .	274
<b>31</b>	<b>Module copads.treenodes</b>	<b>275</b>
31.1	Variables . . . . .	275
31.2	Class RBNode . . . . .	275
31.2.1	Methods . . . . .	275
31.2.2	Properties . . . . .	276
31.3	Class BinaryNode . . . . .	276
31.3.1	Methods . . . . .	276
31.4	Class ThreeNode . . . . .	276
31.4.1	Methods . . . . .	276
31.5	Class FourNode . . . . .	276
31.5.1	Methods . . . . .	276



# 1 Package copads

COPADS (Collection of Python Algorithms and Data Structures).

The main aim of COPADS is to develop a compilation of Python data structures and its algorithms, making it almost a purely developmental project. Personally, I look at this as a re-usable collection of tools that I can use in other projects. Therefore, this project is essentially "needs-driven", except a core subset of data structures and algorithms.

This project originated from 3 threads of thought. Firstly, while browsing through Mehta and Sahni's Handbook of Data Structures and Applications, I thought there might be utility to have a number of the listed data structures implemented in Python. Given my interest in biological data management, having a good set of data structures is always handy. The 2nd thread of thought came from Numerical Recipes. Again, I thought these algorithms will be handy to have and had started to translate some of them into Python during some overly energetic days. Finally, Python Cookbook had undergone 2 editions by 2008 and ActiveState had provided an online platform for Python Recipes which I found to be useful and can see how some of these recipes can be merged. Thus, COPADS is borned.

Project website: <http://copads.sf.net>

License: Unless specified otherwise, all parts of this package, except those adapted, are covered under Python Software Foundation License version 2.

**Version:** 0.4.1

**Author:** Maurice H.T. Ling <mauriceling@acm.org>

**Copyright:** (c) 2007-2013, Maurice H.T. Ling.

## 1.1 Modules

- **array:** Array Data Structures and Algorithms.  
(Section 2, p. 5)
- **bag:** Bag Data Structures and Algorithms.  
(Section 3, p. 7)
- **colormap:** These functions, when given a magnitude mag between cmin and cmax, return a colour tuple (red, green, blue).  
(Section 4, p. 10)
- **constants:** List of Constant Numbers  
(Section 5, p. 11)
- **copadsexceptions:** Exceptions Defined for COPADS.  
(Section 6, p. 13)
- **dataframe:** A generic dataframe to hold data and analysis Date created: 24th September 2012 Licence: Python Software Foundation License version 2  
(Section 7, p. 47)
- **dose\_entities:** Boiler-plate codes for DOSE (digital organism simulation environment) entities Date created: 13th September 2012 Licence: Python Software Foundation License version 2  
(Section 8, p. 50)
- **dose\_executor:** Default DOSE simulation runner in manuscript [1] Date created: 13th September 2012 Licence: Python Software Foundation License version 2  
(Section 9, p. 59)
- **dose\_parameters:** Parameter file for DOSE simulation.

- (Section 10, p. 61)*
- **dose.world**: World structure for DOSE (digital organism simulation environment) Date created: 13th September 2012 Licence: Python Software Foundation License version 2  
*(Section 11, p. 64)*
- **genetic**: Framework for Genetic Algorithm Applications.  
*(Section 12, p. 69)*
- **graph**: Graph Data Structures and Algorithms.  
*(Section 13, p. 85)*
- **hash**: Hash Generators Date created: 16th April 2013 Licence: Python Software Foundation License version 2  
*(Section 14, p. 89)*
- **hypothesis**: Statistical Hypothesis Testing Routines.  
*(Section 15, p. 92)*
- **lc\_bf**: Loose Circular Brainfuck (LCBF) Interpreter Date created: 15th August 2012 Licence: Python Software Foundation License version 2  
*(Section 16, p. 116)*
- **matrix**: Matrix Data Structures and Algorithms.  
*(Section 17, p. 118)*
- **n\_bf**: NucleotideBF (nBF) Interpreter Date created: 15th August 2012 Licence: Python Software Foundation License version 2  
*(Section 18, p. 130)*
- **neural**: Framework for Neural Network Applications Date created: 10th May 2013 License: Python Software Foundation License version 2  
*(Section 19, p. 131)*
- **nrpy**: Numerical Recipes in Python.  
*(Section 20, p. 138)*
- **objectdistance**: Functions for Calculating Similarity Coefficients between 2 Objects.  
*(Section 21, p. 150)*
- **operations**: Mathematical Operation Routines.  
*(Section 22, p. 169)*
- **prioritydictionary**: Priority Dictionary and Algorithms.  
*(Section 23, p. 174)*
- **ragaraja**: Ragaraja Interpreter Date created: 16th August 2012 Licence: Python Software Foundation License version 2  
*(Section 24, p. 176)*
- **register\_machine**: One dimensional tape/register machine Date created: 15th August 2012 Licence: Python Software Foundation License version 2  
*(Section 25, p. 189)*
- **ring**: Ring Data Structures and Algorithms.  
*(Section 26, p. 191)*
- **samplestatistics**: Data Structures and Algorithms for Data Collected from One or More Samples.  
*(Section 27, p. 193)*
- **set**: Sets Data Structure and Algorithms.  
*(Section 28, p. 197)*
- **statisticsdistribution**: Classes for Various Statistical Distributions.  
*(Section 29, p. 199)*
- **tree**: Tree Data Structures and Algorithms.  
*(Section 30, p. 267)*
- **treenodes**: Node Classes for Tree Data Structures.  
*(Section 31, p. 275)*

## 1.2 Variables

Name	Description
__package__	<b>Value:</b> None

## 2 Module *copads.array*

Array Data Structures and Algorithms.

Copyright (c) Maurice H.T. Ling <mauriceling@acm.org>

Date created: 19th March 2008

### 2.1 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'copads'

### 2.2 Class *ParallelArray*

object  **copads.array.ParallelArray**

Parallel Array is an array whereby each data list in the array is of the same size. Ref: [http://en.wikipedia.org/wiki/Parallel\\_Array](http://en.wikipedia.org/wiki/Parallel_Array)

#### 2.2.1 Methods

<b><code>--init--(self, fields=None)</code></b>
Constructor. Able to initiate the array with a list of fields names
<b>Parameters</b>
<b>fields:</b> list of field names to initiate. Default = None
Overrides: <code>object.__init__</code>

<b><code>initFields(self, fields)</code></b>
Initiates a list of fields into the array.
<b>Parameters</b>
<b>fields:</b> list of field names to initiate.

<b><code>addField(self, field)</code></b>
Method to add a new field into the array. Raises <code>ArrayError</code> if attempt to add an existing field.
<b>Parameters</b>
<b>field:</b> name of field
( <i>type=string</i> )

<b>removeField</b> ( <i>self</i> , <i>field</i> )
---

Removes a field, together with its data, from the array.
--

<b>Parameters</b>
-------------------

<b>field:</b> name of field
-----------------------------

( <i>type=string</i> )
------------------------

<b>changeField</b> ( <i>self</i> , <i>oldname</i> , <i>newname</i> )
--

Change a field name.
----------------------

<b>Parameters</b>
-------------------

<b>oldname:</b> existing name of field
--

( <i>type=string</i> )
------------------------

<b>newname:</b> new name to change to. A new field by this name will be created if 'oldname' is not found
---

( <i>type=string</i> )
------------------------

<b>addData</b> ( <i>self</i> , <i>fields</i> , <i>data</i> )
--

Add data into the array
-------------------------

<b>Parameters</b>
-------------------

<b>fields:</b> ordered list of fields
---------------------------------------

<b>data:</b> ordered list of data to add
--

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 2.2.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 3 Module copads.bag

Bag Data Structures and Algorithms.

#### 3.1 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'copads'

#### 3.2 Class Bag



Bag is also known as multi-set, or a set that can have duplicate elements

Adapted from: <http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/259174> Original author: Raymond Hettinger

##### 3.2.1 Methods

<b><code>--init--(self, iterable={})</code></b> Constructor. <b>Parameters</b> <b>iterable:</b> dictionary of elements to add Overrides: <code>object.__init__</code>
<b><code>update(self, iterable)</code></b> Update the bag <b>Parameters</b> <b>iterable:</b> dictionary if elements to add
<b><code>--contains--(self, elem)</code></b>
<b><code>--getitem--(self, elem)</code></b>
<b><code>--setitem--(self, elem, n)</code></b>

```
--delitem--(self, elem)
```

```
--len--(self)
```

```
--eq--(self, other)
```

```
--ne--(self, other)
```

```
--hash--(self)
```

```
hash(x)
```

```
Overrides: object.__hash__ extit(inherited documentation)
```

```
--repr--(self)
```

```
repr(x)
```

```
Overrides: object.__repr__ extit(inherited documentation)
```

```
copy(self)
```

```
--copy--(self)
```

```
--deepcopy--(self, memo)
```

```
--getstate--(self)
```

```
--setstate--(self, data)
```

```
clear(self)
```

```
Remove all data elements in the bag
```

```
--iter--(self)
```

```
iterunique(self)
```

```
itercounts(self)
```

<b>mostcommon</b> ( <i>self</i> , <i>n=None</i> )
---

Lists the most common elements of the bag and its counts
--

<b>Return Value</b>
---------------------

list of (element, count of the element)
---

***Inherited from object***

```
__delattr__(), __format__(), __getattr__(), __new__(), __reduce__(), __reduce_ex__(),
__setattr__(), __sizeof__(), __str__(), __subclasshook__()
```

**3.2.2 Properties**

Name	Description
<i>Inherited from object</i>	
__class__	



## 4 Module *copads.colormap*

These functions, when given a magnitude *mag* between *cmin* and *cmax*, return a colour tuple (red, green, blue). Light blue is cold (low magnitude) and yellow is hot (high magnitude).

Adapted from <http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/52273> Original author: Alexander Pletzer

### 4.1 Functions

<b>floatRgb</b> ( <i>mag</i> , <i>cmin</i> , <i>cmax</i> )
--

Return a tuple of floats between 0 and 1 for the red, green and blue amplitudes.
--

<b>strRgb</b> ( <i>mag</i> , <i>cmin</i> , <i>cmax</i> )
--

Return a tuple of strings to be used in Tk plots.
---

<b>rgb</b> ( <i>mag</i> , <i>cmin</i> , <i>cmax</i> )
---

Return a tuple of integers to be used in AWT/Java plots.
--

<b>htmlRgb</b> ( <i>mag</i> , <i>cmin</i> , <i>cmax</i> )
---

Return a tuple of strings to be used in HTML documents.
---

### 4.2 Variables

Name	Description
--package--	<b>Value:</b> 'copads'

## 5 Module **copads.constants**

### List of Constant Numbers

- CGOLD: 2 - GOLD
- FRODA: Froda Constant
- GAMMA: Euler's gamma constant
- GOLD: golden mean, also known as golden ratio  $((1 + \sqrt{5})/2)$
- GOLOMB = Golomb Constant
- INVLN10: inverse of natural logarithm of ten  $(1/\ln(10))$
- INVLN2: inverse of natural logarithm of 2  $(1/\ln(2))$
- INVPI180: inverse of PI180, also means 180 divide by  $\pi$
- INVSQRT2PI: inverse of SQRT2PI  $(1/(\sqrt{2\pi}))$
- INV2PI: inverse of  $2\pi$
- LEHMER: Lehmer constant
- LN10: natural logarithm of 10
- LN2: natural logarithm of 2
- LNPI: natural logarithm of  $\pi$
- PI: ratio of circumference of a circle to its diameter ( $\pi$ )
- PI2:  $2\pi$
- PI180:  $\pi$  divided by 180
- PIDIV2:  $\pi$  divided by 2
- RABBIT: Rabbit Constant
- SQRT2: square root of 2
- SQRT2PI: square root of  $2\pi$
- ZETA9:  $\sum(1/n^9, n=1..infinity)$
- CFRACT:  $\pi^2/(6 (\ln 2) (\ln 10))$
- PARIS: Paris Constant
- LENGYEL: Lengyel Constant
- BACKHOUSE: Backhouse Constant
- PORTER: Porter Constant

Copyright (c) Maurice H.T. Ling <mauriceling@acm.org>

Date created: 19th March 2008

### 5.1 Variables

Name	Description
CGOLD	Value: 0.38196601125
FRODA	Value: 6.58088599102
GAMMA	Value: 0.577215664902
GOLD	Value: 1.61803398875
GOLOMB	Value: 0.624329988544
INVLN10	Value: 0.434294481903
INVLN2	Value: 1.44269504089
INVPI180	Value: 57.2957795131
INVSQRT2PI	Value: 0.398942280401
INV2PI	Value: 0.159154943092
LEHMER	Value: 0.592632718292
LN10	Value: 2.30258509299
LN2	Value: 0.69314718056
LNPI	Value: 1.14472988585
PI	Value: 3.14159265359
PI2	Value: 6.28318530718
PI180	Value: 0.0174532925199
RABBIT	Value: 0.709803442861
PIDIV2	Value: 1.57079632679
SQRT2	Value: 1.41421356237
SQRT2PI	Value: 2.50662827463
ZETA9	Value: 1.00200839283
CFRACT	Value: 1.0306408341
PARIS	Value: 1.09864196439
LENGYEL	Value: 1.09868580553
BACKHOUSE	Value: 1.45607494858
PORTER	Value: 1.46707807943
__package__	Value: None

## 6 Module `copads.copadsexceptions`

Exceptions Defined for COPADS.

- `CopadsError`
  - `MatrixError`
    - \* `MatrixArithmeticError`
      - `MatrixMultiplicationError`
    - \* `MatrixAdditionError`
    - \* `MatrixSquareError`
    - \* `MatrixTraceError`
    - \* `MatrixMinorError`
    - \* `MatrixDeterminantError`
  - `GraphError`
    - \* `EdgeNotFoundError`
    - \* `VertexNotFoundError`
    - \* `UnknownGraphMatrixError`
      - `NotAdjacencyGraphMatrixError`
    - \* `GraphEdgeSizeMismatchError`
  - `StatisticsError`
    - \* `DistributionError`
      - `NormalDistributionTypeError`
    - \* `DistributionParameterError`
    - \* `DistributionFunctionError`
  - `DistanceError`
    - \* `DistanceInputSizeError`
  - `TreeError`
    - \* `TreeNodeTypeError`
  - `FunctionParameterTypeError`
  - `FunctionParameterValueError`
  - `ArrayError`
  - `MaxIterationException`

Credits

- `MatrixError` subclasses (<http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/189971>)

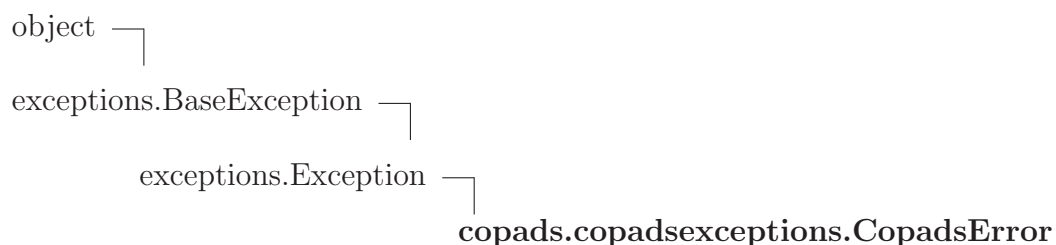
Copyright (c) Maurice H.T. Ling <[mauriceling@acm.org](mailto:mauriceling@acm.org)>

Date created: 1st May 2005

## 6.1 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> None

## 6.2 Class CopadsError



**Known Subclasses:** `copads.copadsexceptions.ArrayError`, `copads.copadsexceptions.DistanceError`, `copads.copadsexceptions.StatisticsError`, `copads.copadsexceptions.GraphError`, `copads.copadsexceptions.F`, `copads.copadsexceptions.FunctionParameterValueError`, `copads.copadsexceptions.MatrixError`, `copads.copadsexceptions.MaxIterationsException`, `copads.copadsexceptions.TreeError`

Base class for all Copads-defined exceptions.

### 6.2.1 Methods

*Inherited from `exceptions.Exception`*

`--init--()`, `--new--()`

*Inherited from `exceptions.BaseException`*

`--delattr--()`, `--getattr--()`, `--getitem--()`, `--getslice--()`, `--reduce--()`, `--repr--()`, `--setattr--()`, `--setstate--()`, `--str--()`, `--unicode--()`

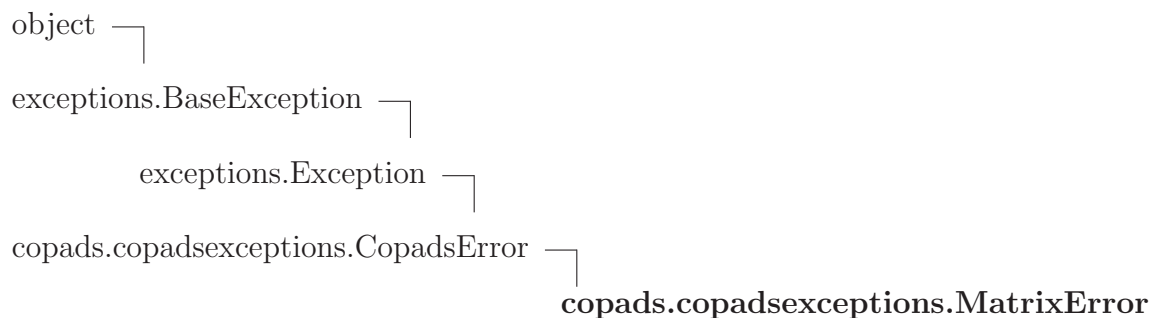
*Inherited from `object`*

`--format--()`, `--hash--()`, `--reduce_ex--()`, `--sizeof--()`, `--subclasshook--()`

### 6.2.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>--class--</code>	

### 6.3 Class `MatrixError`



**Known Subclasses:** `copads.copadsexceptions.MatrixArithmeticError`, `copads.copadsexceptions.MatrixS...`

Abstract parent for all matrix exceptions

#### 6.3.1 Methods

*Inherited from `exceptions.Exception`*

`__init__()`, `__new__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

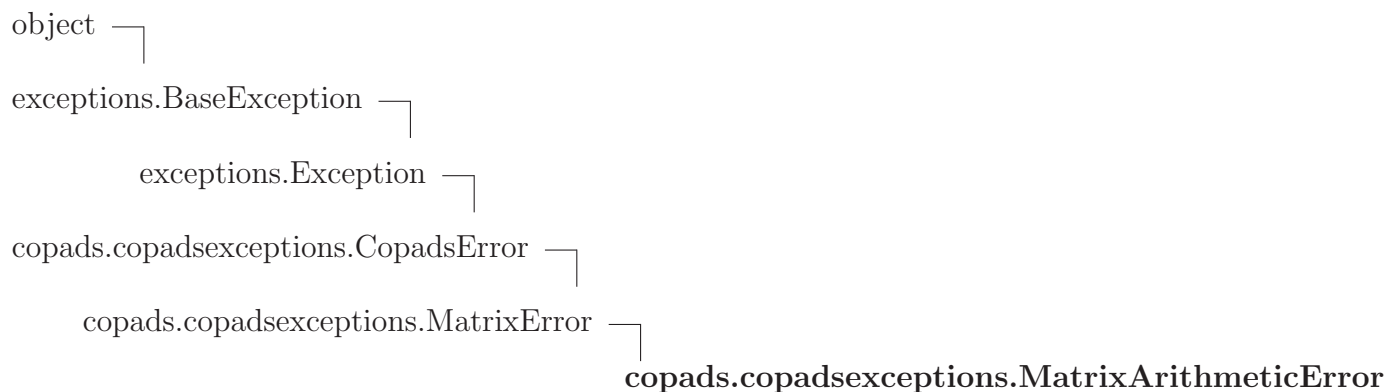
*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 6.3.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
args, message	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 6.4 Class `MatrixArithmeticError`



**Known Subclasses:** `copads.copadsexceptions.MatrixAdditionError`, `copads.copadsexceptions.MatrixMul`

Incorrect dimensions for arithmetic

This exception is thrown when you try to add or multiply matrices of incompatible sizes.

### 6.4.1 Methods

```
__init__(self, a, b, operation)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)
```

```
__str__(self)

str(x)
Overrides: object.__str__ extit(inherited documentation)
```

*Inherited from `exceptions.Exception`*

```
__new__()
```

*Inherited from `exceptions.BaseException`*

```
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __unicode__()
```

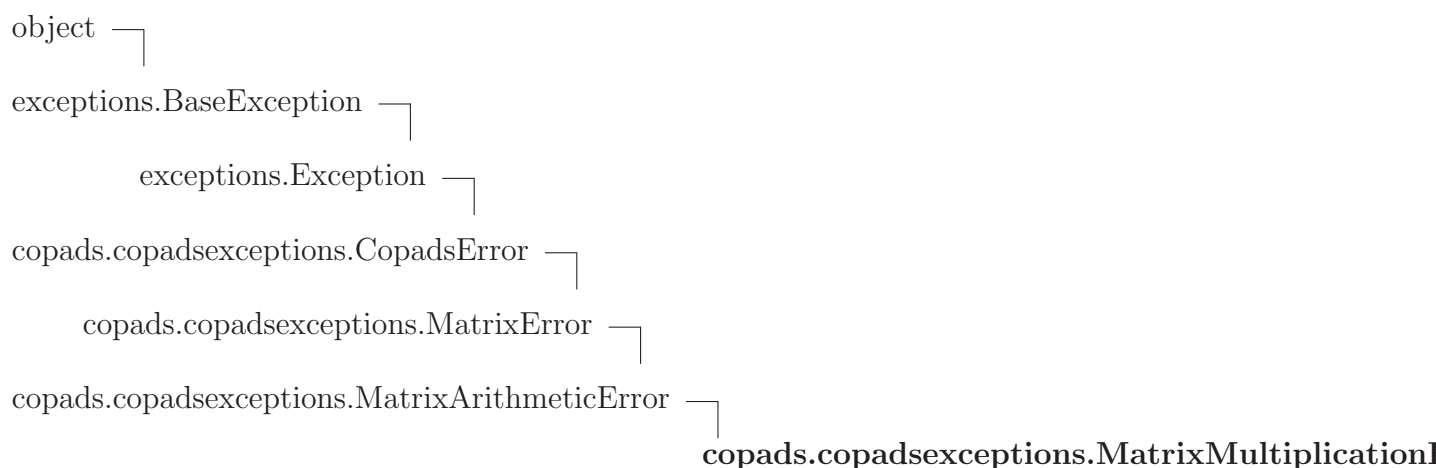
*Inherited from `object`*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

### 6.4.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

## 6.5 Class `MatrixMultiplicationError`



Thrown when you try to multiply matrices of incompatible dimensions.

This exception is also thrown when you try to right-multiply a row vector or left-multiply a column vector.

### 6.5.1 Methods

```

__init__(self, a, b)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

```

*Inherited from `copads.copadsexceptions.MatrixArithmeticError` (Section 6.4)*

```
__str__()
```

*Inherited from `exceptions.Exception`*

```
__new__()
```



***Inherited from `exceptions.BaseException`***

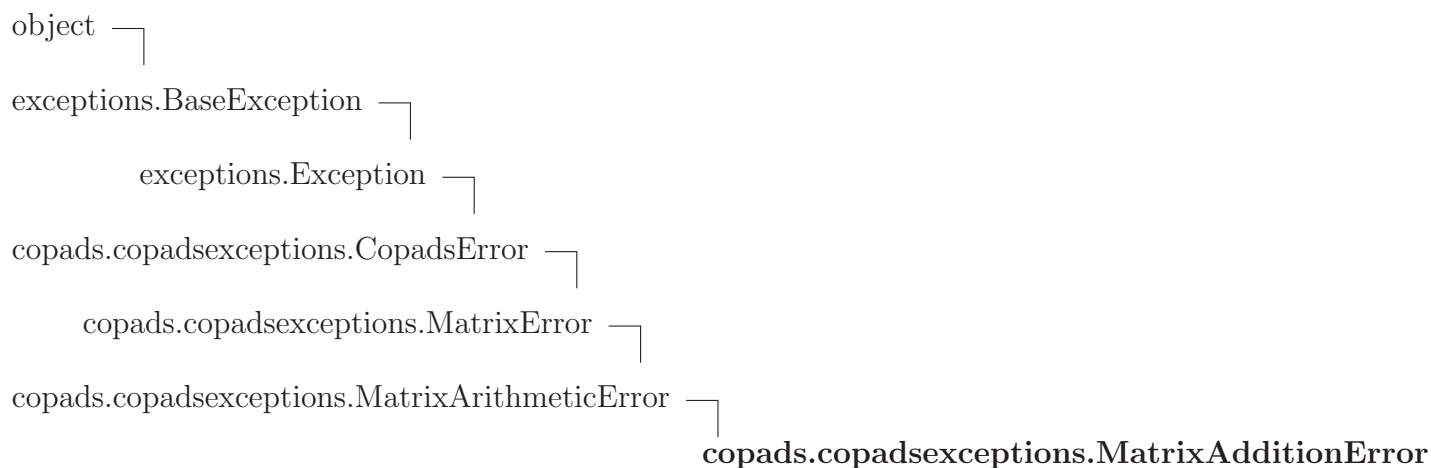
`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__unicode__()`

***Inherited from `object`***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

**6.5.2 Properties**

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

**6.6 Class `MatrixAdditionError`**

Thrown when you try to add matrices of incompatible dimensions.

**6.6.1 Methods**

`__init__(self, a, b)`  
`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` extit(inherited documentation)

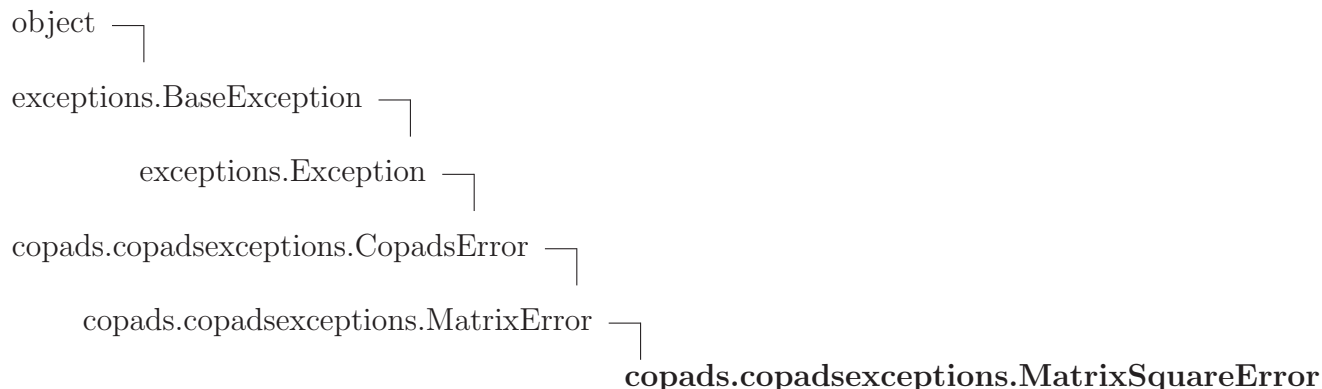
***Inherited from `copads.copadsexceptions.MatrixArithmeticError` (Section 6.4)***

`__str__()`***Inherited from `exceptions.Exception`***`__new__()`***Inherited from `exceptions.BaseException`***`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__unicode__()`***Inherited from object***`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 6.6.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

## 6.7 Class `MatrixSquareError`



**Known Subclasses:** `copads.copadsexceptions.MatrixDeterminantError`, `copads.copadsexceptions.MatrixTraceError`

Square-matrix only

This exception is thrown when you try to calculate a function that is only defined for square matrices on a non-square matrix.

**6.7.1 Methods**

```
__init__(self, func)
```

*x*.**\_\_init\_\_**(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit`(inherited documentation)

```
__str__(self)
```

`str(x)`

Overrides: `object.__str__` `exitit`(inherited documentation)

***Inherited from exceptions.Exception***

```
__new__()
```

***Inherited from exceptions.BaseException***

```
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
```

```
__setattr__(), __setstate__(), __unicode__()
```

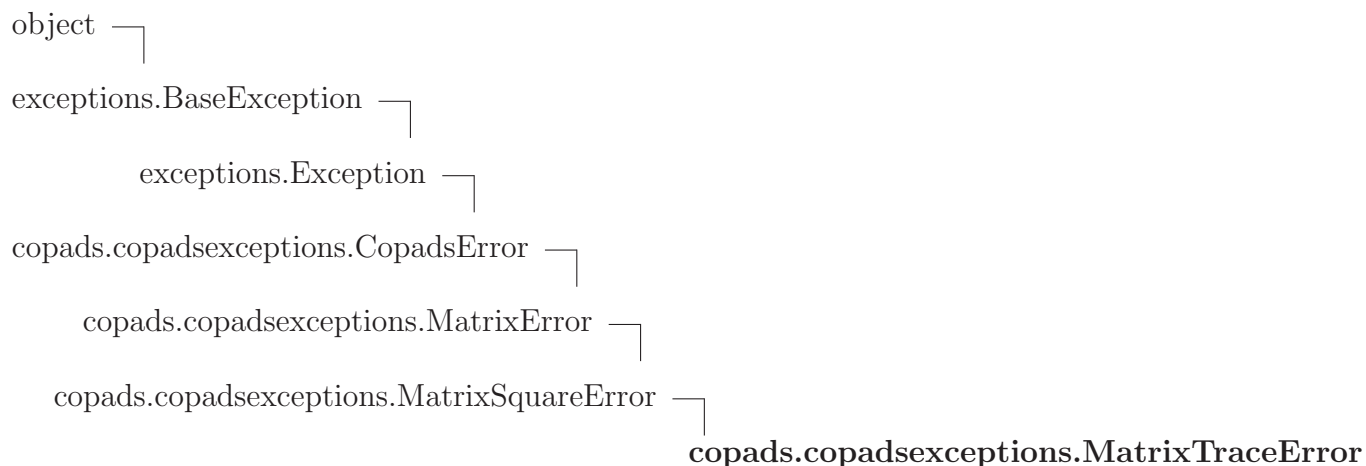
***Inherited from object***

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

**6.7.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

## 6.8 Class `MatrixTraceError`



Thrown when you try to get the trace of a non-square matrix.

### 6.8.1 Methods

`__init__(self)`  
`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` `exitit`(inherited documentation)

*Inherited from `copads.copadsexceptions.MatrixSquareError` (Section 6.7)*

`__str__()`

*Inherited from `exceptions.Exception`*

`__new__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__unicode__()`

*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

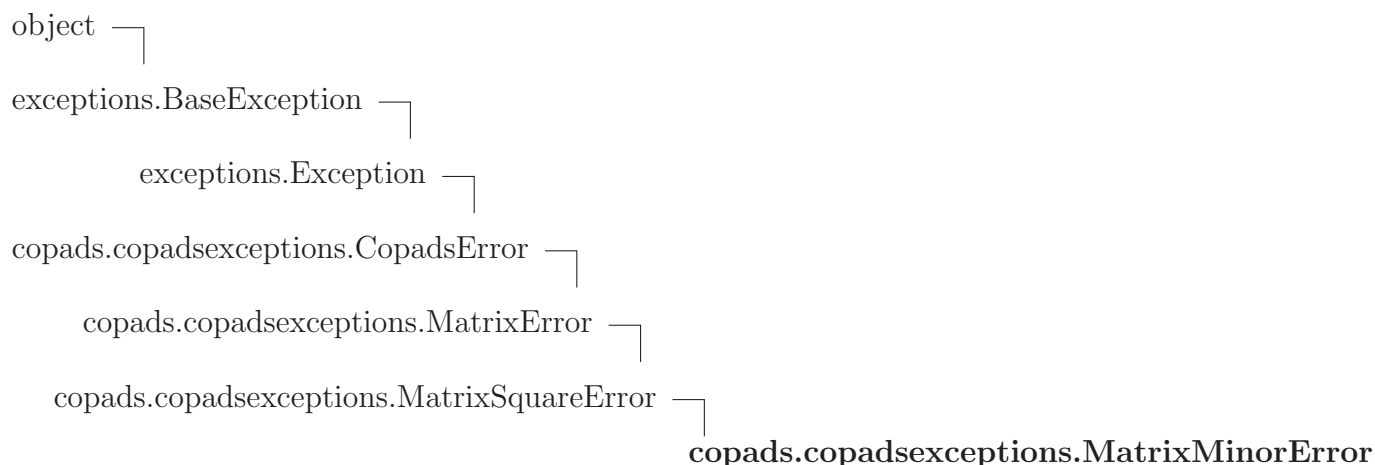
### 6.8.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	

*continued on next page*

Name	Description
args, message	
<i>Inherited from object</i>	
<code>__class__</code>	

## 6.9 Class `MatrixMinorError`



Thrown when you try to take a minor of a non-square matrix.

### 6.9.1 Methods

<b><code>__init__</code></b> ( <i>self</i> ) <code>x.__init__(...)</code> initializes <code>x</code> ; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> extit(inherited documentation)
--

*Inherited from `copads.copadsexceptions.MatrixSquareError` (Section 6.7)*

`__str__`()

*Inherited from `exceptions.Exception`*

`__new__`()

*Inherited from `exceptions.BaseException`*

`__delattr__`(), `__getattr__`(), `__getitem__`(), `__getslice__`(), `__reduce__`(), `__repr__`(),  
`__setattr__`(), `__setstate__`(), `__unicode__`()

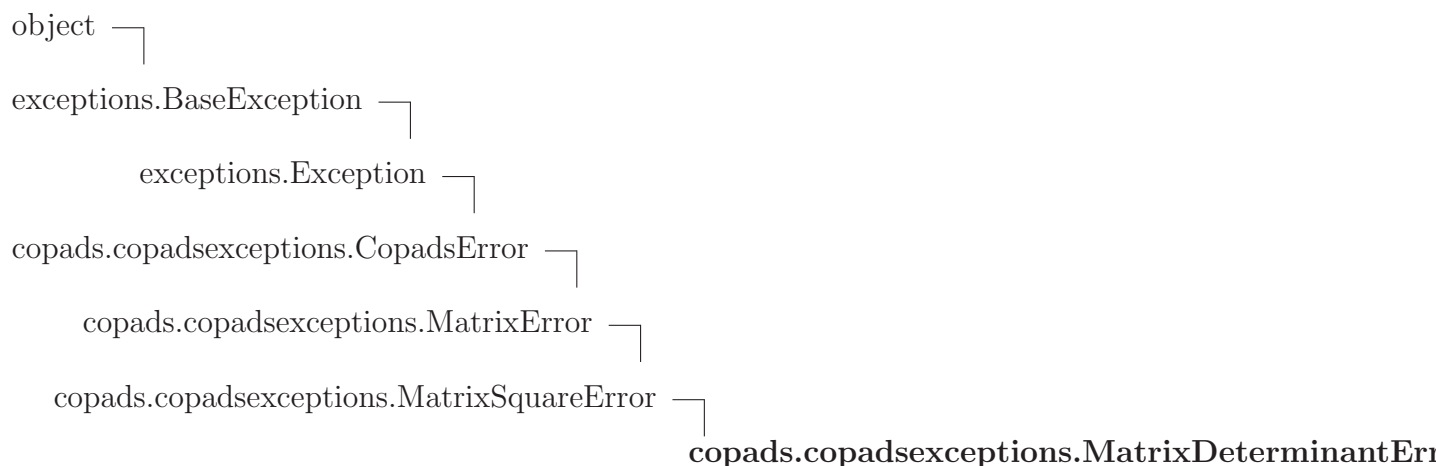
*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 6.9.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

## 6.10 Class `MatrixDeterminantError`



Thrown when you try to take the determinant of a non-square matrix.

### 6.10.1 Methods

<p><b><code>__init__(self)</code></b></p> <p><code>x.__init__(...)</code> initializes <code>x</code>; see <code>help(type(x))</code> for signature</p> <p>Overrides: <code>object.__init__</code> <code>exitit</code>(inherited documentation)</p>
--

*Inherited from `copads.copadsexceptions.MatrixSquareError`(Section 6.7)*

`__str__()`

*Inherited from `exceptions.Exception`*

`__new__()`

***Inherited from `exceptions.BaseException`***

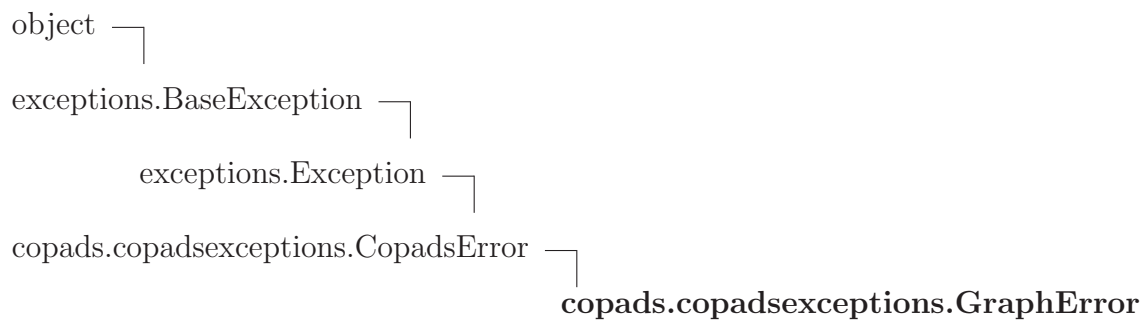
`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__unicode__()`

***Inherited from `object`***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

**6.10.2 Properties**

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

**6.11 Class `GraphError`**

**Known Subclasses:** `copads.copadsexceptions.EdgeNotFoundError`, `copads.copadsexceptions.GraphEdge`,  
`copads.copadsexceptions.GraphParameterError`, `copads.copadsexceptions.UnknownGraphMatrixError`,  
`copads.copadsexceptions.VertexNotFoundError`

Abstract parent for all graph exceptions

**6.11.1 Methods*****Inherited from `exceptions.Exception`***

`__init__()`, `__new__()`

***Inherited from `exceptions.BaseException`***

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,

`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

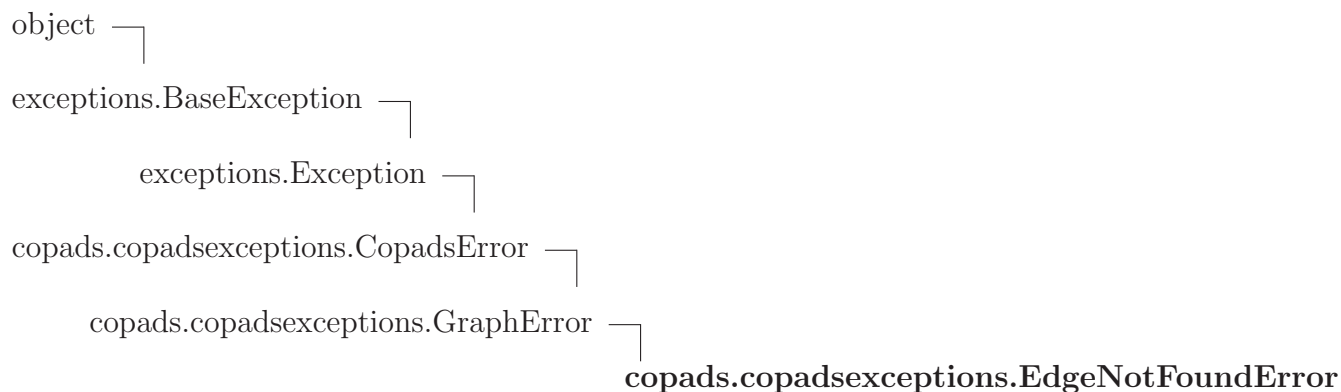
### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 6.11.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

## 6.12 Class `EdgeNotFoundError`



Exception to be thrown when trying to retrieve an edge that is not found in the graph.

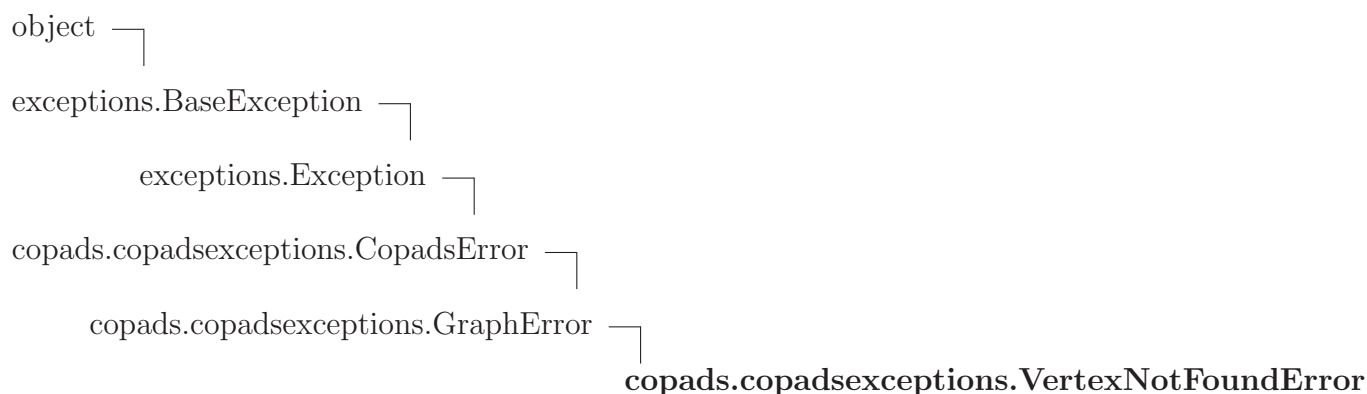
#### 6.12.1 Methods

<p><b><code>__init__(self, edge)</code></b></p> <p><code>x.__init__(...)</code> initializes <code>x</code>; see <code>help(type(x))</code> for signature</p> <p>Overrides: <code>object.__init__</code> <code>extit</code>(inherited documentation)</p>
<p><b><code>__str__(self)</code></b></p> <p><code>str(x)</code></p> <p>Overrides: <code>object.__str__</code> <code>extit</code>(inherited documentation)</p>



***Inherited from `exceptions.Exception`***`__new__()`***Inherited from `exceptions.BaseException`***`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__unicode__()`***Inherited from `object`***`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`**6.12.2 Properties**

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

**6.13 Class `VertexNotFoundError`**

Exception to be thrown when trying to retrieve a vertex that is not found in the graph.

**6.13.1 Methods**

<b><code>__init__(self, vertex)</code></b> <code>x.__init__(...)</code> initializes <code>x</code> ; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> extit(inherited documentation)
--

```
__str__(self)
str(x)
Overrides: object.__str__ extit(inherited documentation)
```

***Inherited from `exceptions.Exception`***

```
__new__()
```

***Inherited from `exceptions.BaseException`***

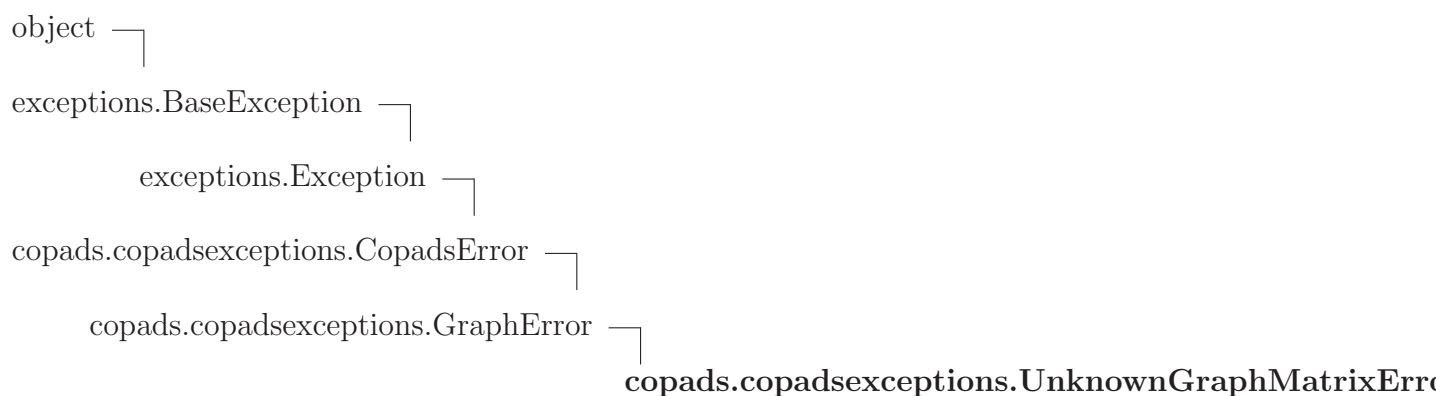
```
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __unicode__()
```

***Inherited from `object`***

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

**6.13.2 Properties**

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

**6.14 Class `UnknownGraphMatrixError`**

**Known Subclasses:** `copads.copadsexceptions.NotAdjacencyGraphMatrixError`

Exception to be thrown when trying to use an unknown type of graph matrix.

**6.14.1 Methods**

```
__init__(self, type)
```

*x*.**\_\_init\_\_**(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

```
__str__(self)
```

`str(x)`

Overrides: `object.__str__` `extit`(inherited documentation)

***Inherited from exceptions.Exception***

```
__new__()
```

***Inherited from exceptions.BaseException***

```
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
```

```
__setattr__(), __setstate__(), __unicode__()
```

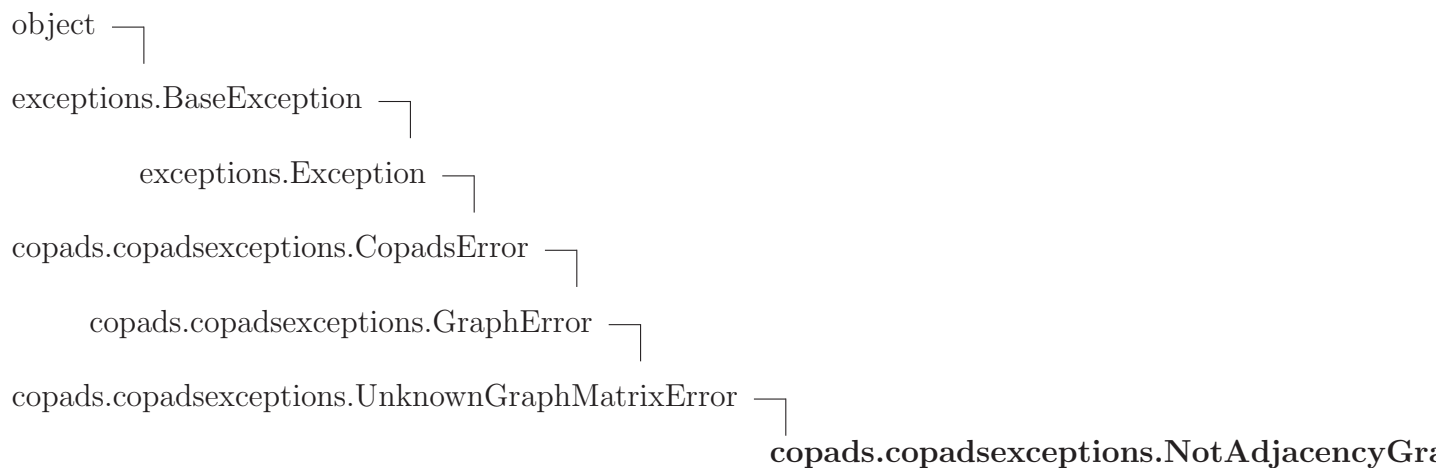
***Inherited from object***

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

**6.14.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

## 6.15 Class `NotAdjacencyGraphMatrixError`



Exception to be thrown when trying to enter a non-square matrix into `Graph.GraphAdjacencyMatrix` object or when not supplying an adjacency matrix when the calculation requires it.

### 6.15.1 Methods

```
__init__(self, type)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` `exitit`(inherited documentation)

```
__str__(self)
```

`str(x)`  
 Overrides: `object.__str__` `exitit`(inherited documentation)

*Inherited from `exceptions.Exception`*

```
__new__()
```

*Inherited from `exceptions.BaseException`*

```
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),  

__setattr__(), __setstate__(), __unicode__()
```

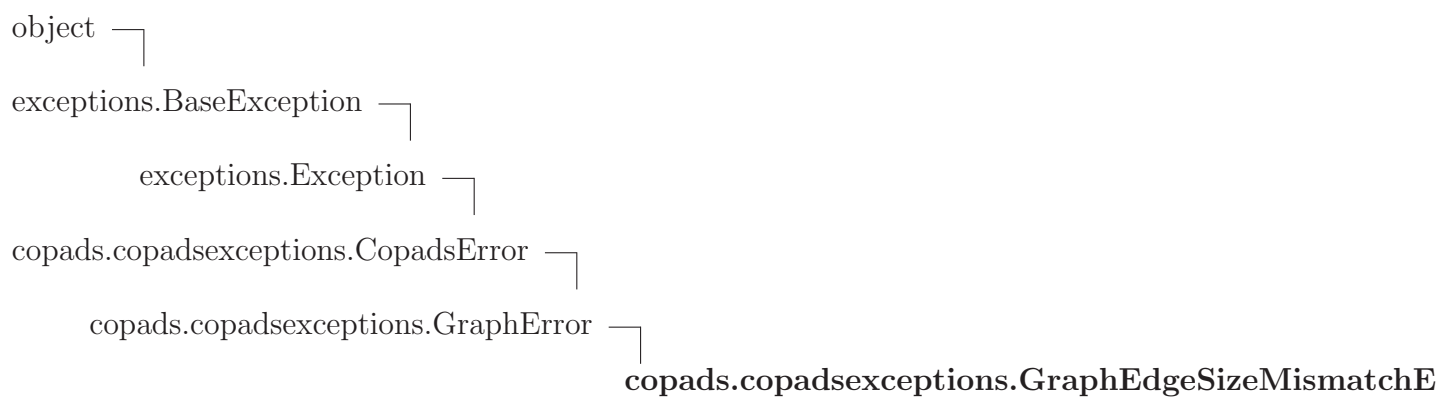
*Inherited from `object`*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

### 6.15.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

## 6.16 Class `GraphEdgeSizeMismatchError`



Exception for number of output edges do not match the number of input edges.

### 6.16.1 Methods

```
__init__(self, index, edge)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` extit(inherited documentation)

```
__str__(self)
```

`str(x)`

Overrides: `object.__str__` extit(inherited documentation)

*Inherited from `exceptions.Exception`*

```
__new__()
```

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__unicode__()`

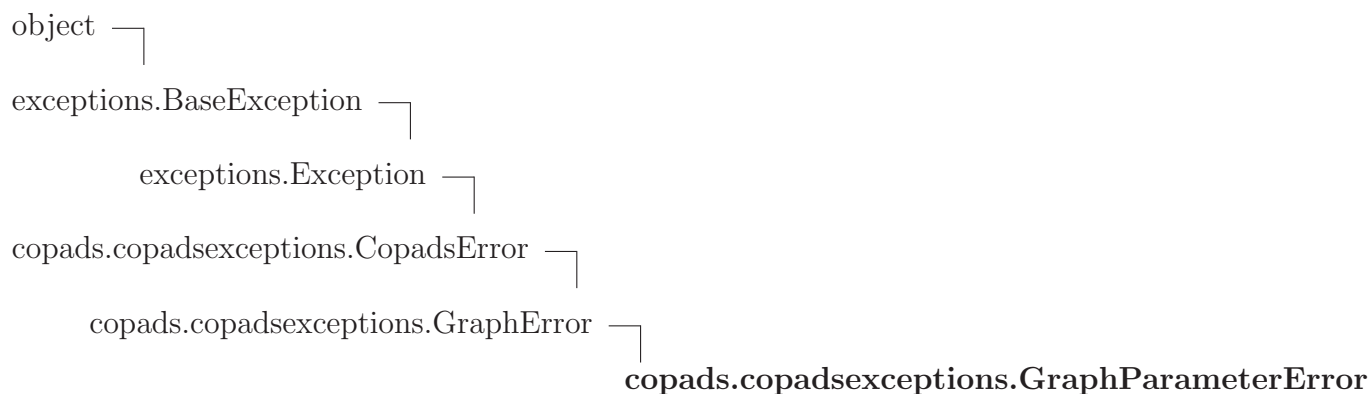
### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 6.16.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

## 6.17 Class `GraphParameterError`



Exception for parameter errors in `Graph.Graph` class.

#### 6.17.1 Methods

`__init__(self, msg)`  
`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` extit(inherited documentation)

```
__str__(self)
str(x)
Overrides: object.__str__ extit(inherited documentation)
```

***Inherited from `exceptions.Exception`***

```
__new__()
```

***Inherited from `exceptions.BaseException`***

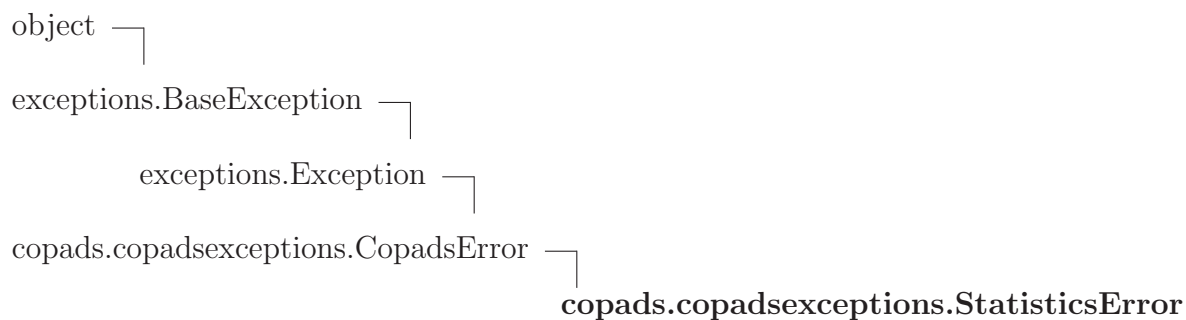
```
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __unicode__()
```

***Inherited from `object`***

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

**6.17.2 Properties**

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

**6.18 Class `StatisticsError`****Known Subclasses:** `copads.copadsexceptions.DistributionError`

Abstract parent for all statistics exceptions

### 6.18.1 Methods

*Inherited from `exceptions.Exception`*

`__init__()`, `__new__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

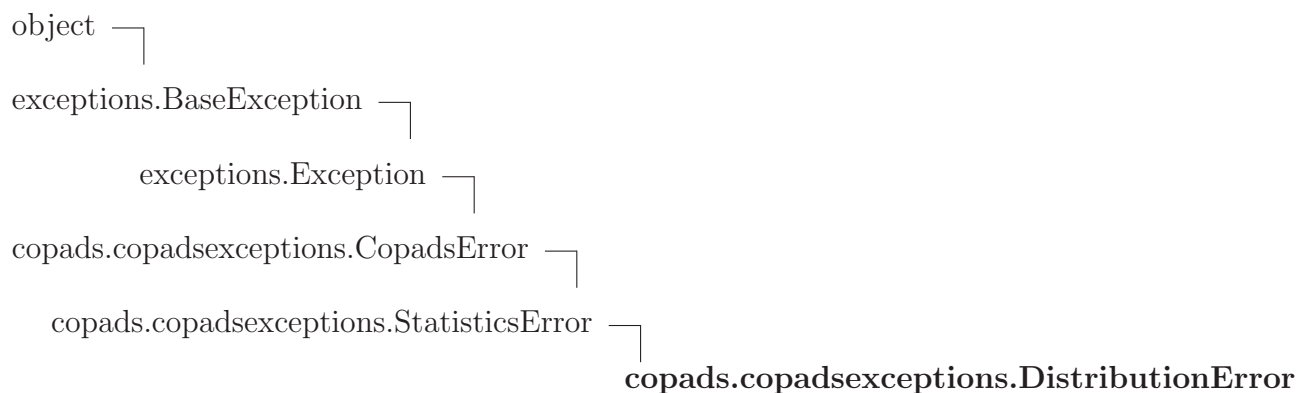
*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 6.18.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 6.19 Class `DistributionError`



**Known Subclasses:** `copads.copadsexceptions.DistributionFunctionError`, `copads.copadsexceptions.DistributionTypeError`, `copads.copadsexceptions.NormalDistributionTypeError`

Abstract parent for all exceptions pertaining to statistical distributions



### 6.19.1 Methods

*Inherited from exceptions.Exception*

`__init__()`, `__new__()`

*Inherited from exceptions.BaseException*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

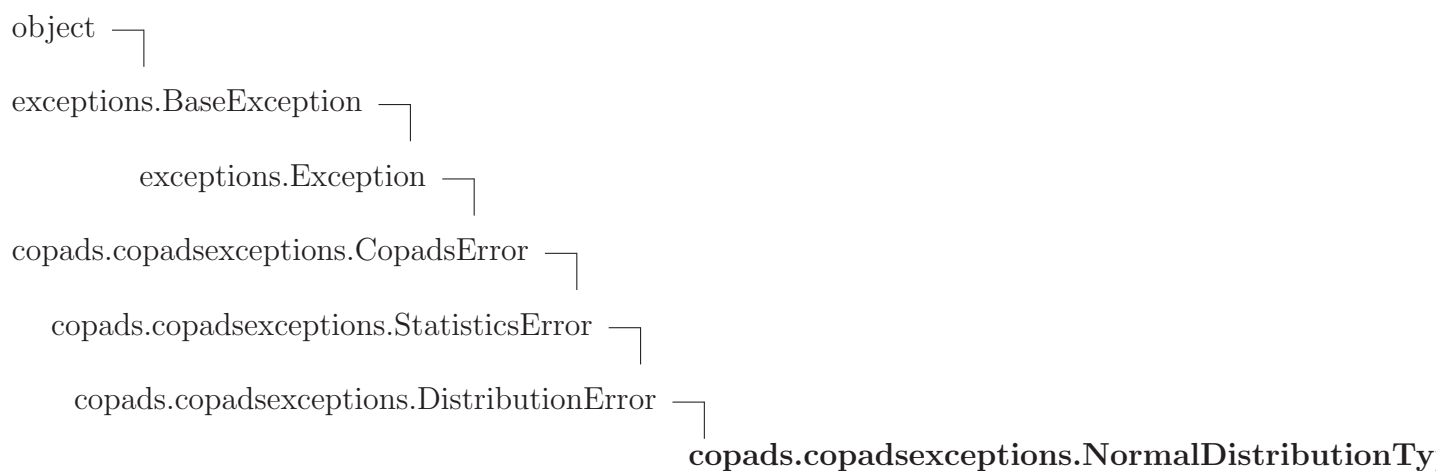
*Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 6.19.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

## 6.20 Class NormalDistributionTypeError



Exception for type errors in normal distribution (`StatisticsDistribution.NormalDistribution`).

**6.20.1 Methods**

```
__init__(self, msg)
```

*x*.**\_\_init\_\_**(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.__init__` extit(inherited documentation)

```
__str__(self)
```

`str(x)`

Overrides: `object.__str__` extit(inherited documentation)

*Inherited from exceptions.Exception*

```
__new__()
```

*Inherited from exceptions.BaseException*

```
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
```

```
__setattr__(), __setstate__(), __unicode__()
```

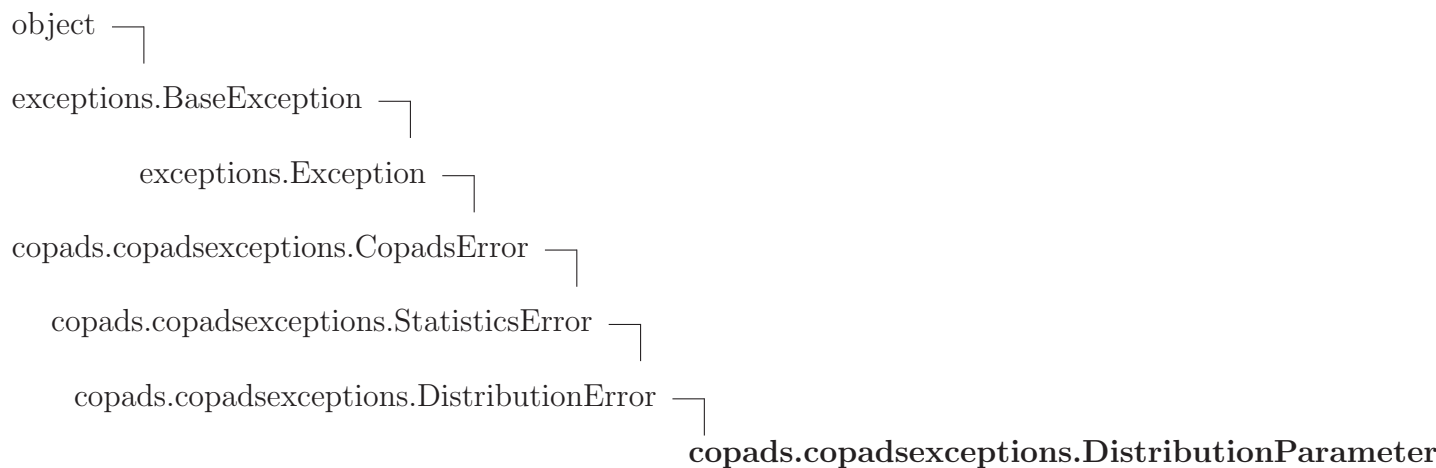
*Inherited from object*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

**6.20.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

## 6.21 Class `DistributionParameterError`



Exception for parameter errors in distributions (`StatisticsDistribution.*`).

### 6.21.1 Methods

**`__init__(self, msg)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit`(inherited documentation)

**`__str__(self)`**

`str(x)`

Overrides: `object.__str__` `exitit`(inherited documentation)

***Inherited from `exceptions.Exception`***

`__new__()`

***Inherited from `exceptions.BaseException`***

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__unicode__()`

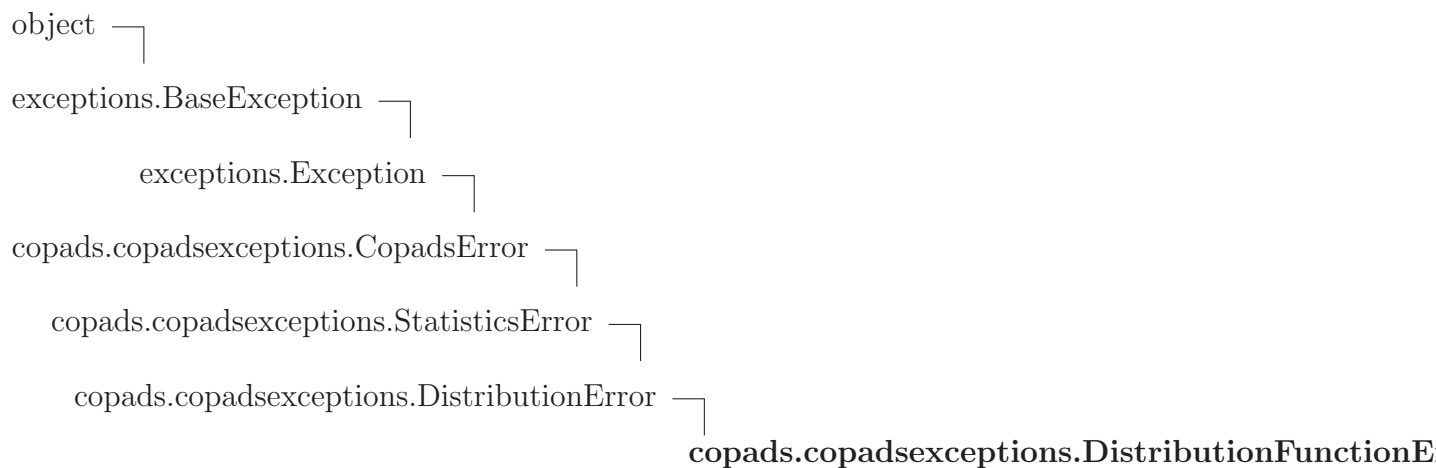
***Inherited from `object`***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 6.21.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 6.22 Class `DistributionFunctionError`



Exception for undefined functions in distributions (`StatisticsDistribution.*`).

### 6.22.1 Methods

```
__init__(self, msg)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` extit(inherited documentation)

```
__str__(self)
```

`str(x)`  
 Overrides: `object.__str__` extit(inherited documentation)

*Inherited from `exceptions.Exception`*

```
__new__()
```

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__unicode__()`

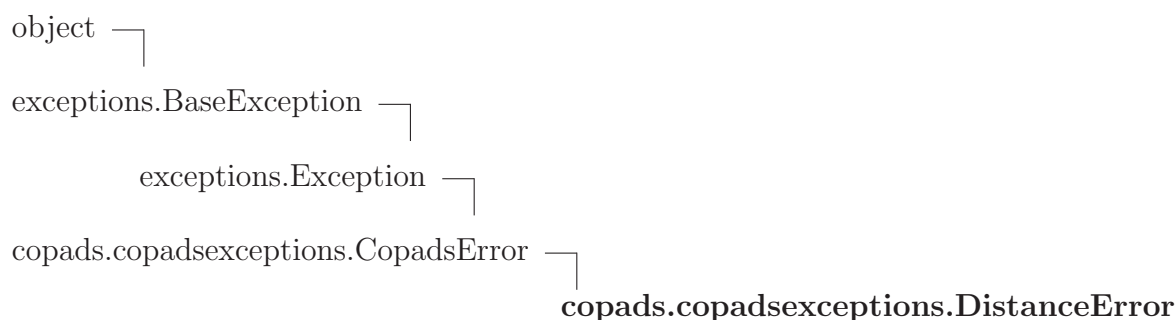
### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 6.22.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<code>__class__</code>	

### 6.23 Class *DistanceError*



**Known Subclasses:** `copads.copadsexceptions.DistanceInputSizeError`

Abstract parent for all exceptions related to calculating distances between lists.

#### 6.23.1 Methods

### *Inherited from exceptions.Exception*

`__init__()`, `__new__()`

### *Inherited from exceptions.BaseException*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

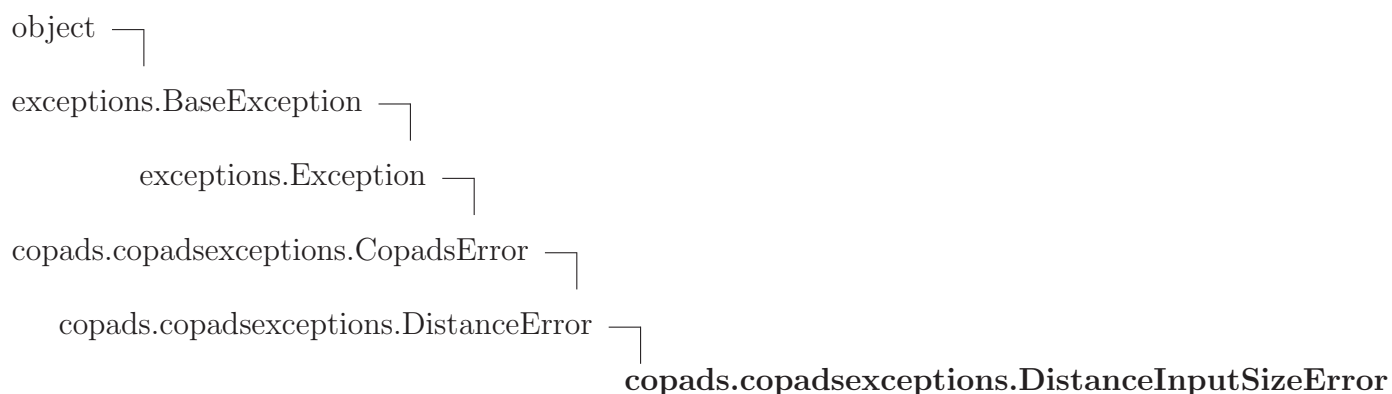
### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 6.23.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	

## 6.24 Class DistanceInputSizeError



Exception for input parameter size errors for list (object) distance routines that have specific requirements for the size of inputs.

### 6.24.1 Methods

```
__init__(self, msg)
```

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature  
 Overrides: object.\_\_init\_\_ extit(inherited documentation)

```
__str__(self)
```

str(x)  
 Overrides: object.\_\_str\_\_ extit(inherited documentation)

*Inherited from exceptions.Exception*

```
__new__()
```

*Inherited from exceptions.BaseException*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__unicode__()`

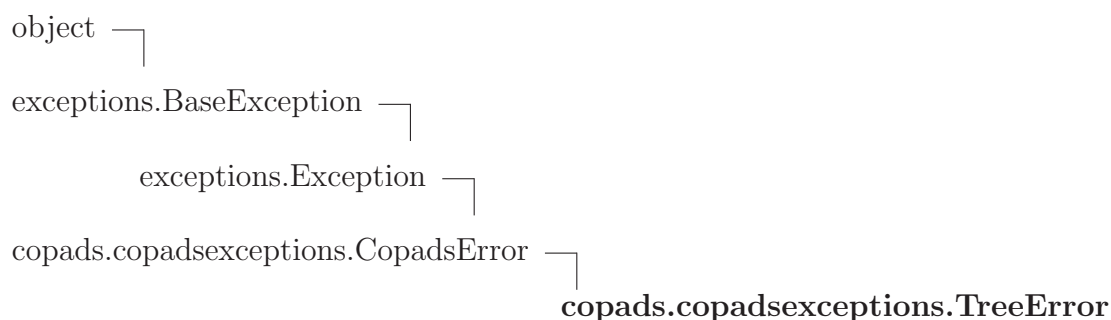
### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 6.24.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

## 6.25 Class `TreeError`



**Known Subclasses:** `copads.copadsexceptions.TreeNodeTypeError`

Abstract parent for all tree exceptions

#### 6.25.1 Methods

### *Inherited from `exceptions.Exception`*

`__init__()`, `__new__()`

### *Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

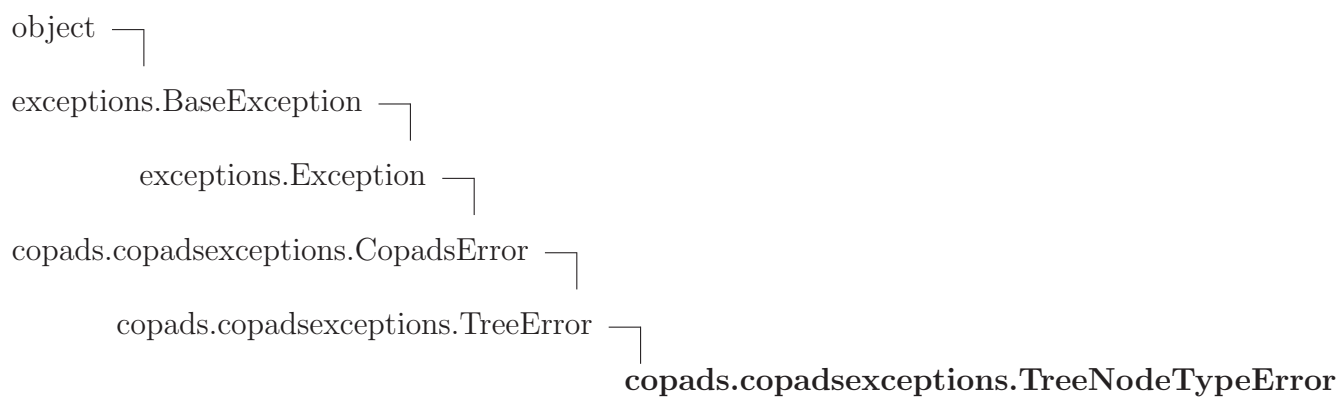
### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 6.25.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 6.26 Class `TreeNodeTypeError`



Exception to be thrown when trying to add a non-Node class object (`Tree.Node`) into a `Tree` object (`Tree.BinaryTree`)

### 6.26.1 Methods

```
__init__(self, msg)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` extit(inherited documentation)

```
__str__(self)
```

`str(x)`  
 Overrides: `object.__str__` extit(inherited documentation)

*Inherited from `exceptions.Exception`*

```
__new__()
```

*Inherited from `exceptions.BaseException`*



`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__unicode__()`

### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 6.26.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

## 6.27 Class `FunctionParameterTypeError`



Exception to be thrown when trying a function parameter is of the wrong data type

#### 6.27.1 Methods

<b><code>__init__(self, msg)</code></b> <code>x.__init__(...)</code> initializes <code>x</code> ; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> <code>extit</code> (inherited documentation)
<b><code>__str__(self)</code></b> <code>str(x)</code> Overrides: <code>object.__str__</code> <code>extit</code> (inherited documentation)

### *Inherited from `exceptions.Exception`*

`__new__()`

***Inherited from `exceptions.BaseException`***

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__unicode__()`

***Inherited from `object`***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 6.27.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 6.28 Class `FunctionParameterValueError`



Exception to be thrown when trying a function parameter is of wrong value

### 6.28.1 Methods

**`__init__(self, msg)`**  
`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` extit(inherited documentation)

```
__str__(self)

str(x)

Overrides: object.__str__ extit(inherited documentation)
```

***Inherited from `exceptions.Exception`***

```
__new__()
```

***Inherited from `exceptions.BaseException`***

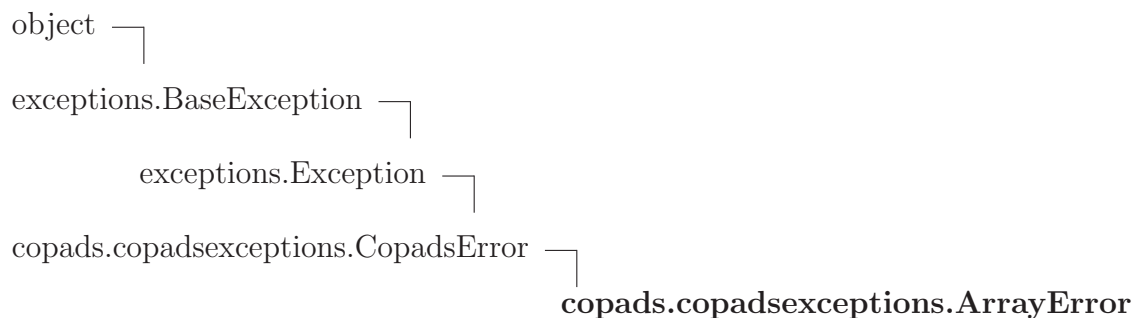
```
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __unicode__()
```

***Inherited from `object`***

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

**6.28.2 Properties**

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

**6.29 Class `ArrayError`**

Exception to be thrown when encountered error in `Array` type

**6.29.1 Methods**

```
__init__(self, msg)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` extit(inherited documentation)

```
__str__(self)
```

`str(x)`

Overrides: `object.__str__` extit(inherited documentation)

***Inherited from `exceptions.Exception`***

```
__new__()
```

***Inherited from `exceptions.BaseException`***

```
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __unicode__()
```

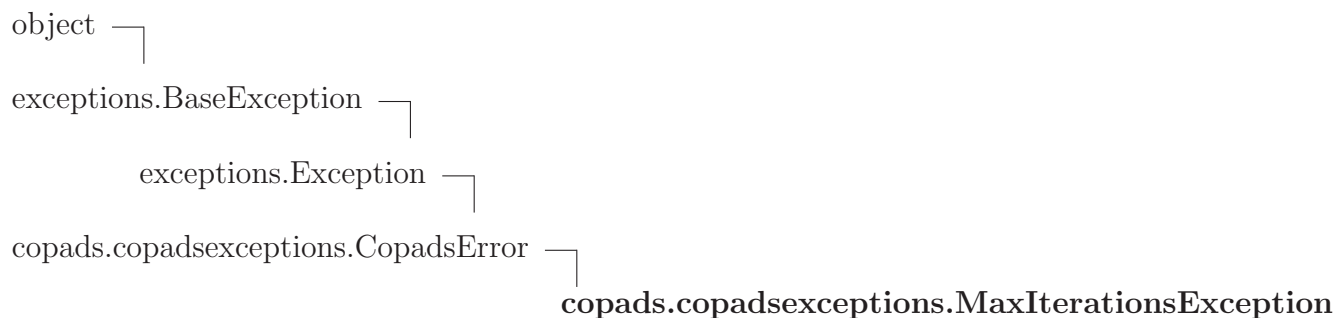
***Inherited from `object`***

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

**6.29.2 Properties**

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args, message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

### 6.30 Class `MaxIterationsException`



Exception to catch maximum looping.

#### 6.30.1 Methods

*Inherited from `exceptions.Exception`*

`__init__()`, `__new__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 6.30.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
args, message	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 7 Module `copads.dataframe`

A generic dataframe to hold data and analysis Date created: 24th September 2012 Licence: Python Software Foundation License version 2

### 7.1 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> 'copads'

### 7.2 Class `dataframe`

object └─ **`copads.dataframe.dataframe`**

A data frame is an encapsulation of one or more data series and its associated analyses.

#### 7.2.1 Methods

```
__init__(self, name='')
```

Constructor. Initialize data frame with a name.

**Parameters**

**name:** Name of this data frame  
(*type=string*)

Overrides: `object.__init__`

```
addSeries(self, name, data=[])
```

Add a data series (vector/column) to the frame.

**Parameters**

**name:** Name of data series  
(*type=string*)  
**data:** Data to be added  
(*type=list*)

**changeDataType**(*self*, *name*, *type*='float')

Change the data type for a data series

**Parameters**

**name:** Name of data series

(*type=string*)

**type:** Type of data to cast data series into. Allowable = {float | integer | string | long}. Default = float

**getSeries**(*self*, *name*)

Get values of a data series.

**Parameters**

**name:** Name of data series

(*type=string*)

**Return Value**

List of data

**has\_series**(*self*, *name*)

Check whether data frame has a certain data series (by name).

**Parameters**

**name:** Name of data series

(*type=string*)

**Return Value**

True, if the data series is present; False, if the data series is absent

**changeDatum**(*self*, *name*, *observation*, *newvalue*)

Change the value of a datum in a data series.

**Parameters**

**name:** Name of data series

(*type=string*)

**observation:** Observation number to change (starting count = 1)

(*type=integer*)

**newvalue:** new value for the observation

<b>getObservation</b> ( <i>self</i> , <i>observation</i> )
Get values of an observation
<b>Parameters</b>
<b>observation</b> : Observation number to change (starting count = 1) ( <i>type=integer</i> )
<b>Return Value</b>
Dictionary of data

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 7.2.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 7.2.3 Class Variables

Name	Description
<code>data</code>	<b>Value:</b> {}
<code>analyses</code>	<b>Value:</b> {}
<code>data.length</code>	<b>Value:</b> 0
<code>frame_name</code>	<b>Value:</b> ''



## 8 Module copads.dose\_entities

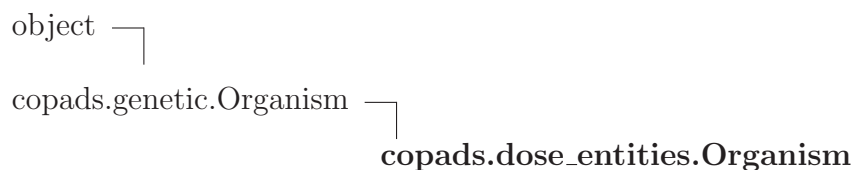
Boiler-plate codes for DOSE (digital organism simulation environment) entities Date created: 13th September 2012 Licence: Python Software Foundation License version 2

Reference: Ling, MHT. 2012. An Artificial Life Simulation Library Based on Genetic Algorithm, 3-Character Genetic Code and Biological Hierarchy. The Python Papers 7: 5.

### 8.1 Variables

Name	Description
Chromosome	<b>Value:</b> g.Chromosome(initial_chromosome, ['0', '1', '2', '3', '4']...

### 8.2 Class Organism



## 8.2.1 Methods

**`__init__(self)`**

Sets up a new organism with default status (age = 0, vitality = 100, lifespan = 100, fitness = 100, alive = True)

**Parameters**

<b>genome:</b>	list of chromosomes to inherit. Default = 'default', which will set up one default chromosome. It also allows a 'dummy' chromosome which is basically a one-base chromosome - this is for applications which does not utilize the chromosome.
<b>mutation_type:</b>	type of mutation. Accepts 'point' (point mutation), 'insert' (insert a base), 'delete' (delete a base), 'invert' (invert a stretch of the chromosome), 'duplicate' (duplicate a stretch of the chromosome), 'translocate' (translocate a stretch of chromosome to another random position). Default = point.
<b>additional_mutation_rate:</b>	probability of mutation per base above background mutation rate. Default = 0.01 (1%). No mutation event will ever happen if (rate + background_mutation) is less than zero.
<b>gender:</b>	establishes the gender of the organism which may be used for mating routines.

Overrides: object.\_\_init\_\_ extit(inherited documentation)

**`get_cytoplasm(self)`**

**fitness(*self*)**

Function to calculate the fitness of the current organism. **This function MUST be over-ridden by the inherited class or substituted as fitness function is highly dependent on utility.** The only requirement is that a fitness score must be returned.

Here, the sample implementation calculates fitness as proportion of the genome with '1's.

**Return Value**

fitness score or fitness list

Overrides: copads.genetic.Organism.fitness extit(inherited documentation)

**mutation\_scheme(*self*)**

Function to trigger mutation events in each chromosome. **This function may be over-ridden by the inherited class or substituted to cater for specific mutation schemes but not an absolute requirement to do so.**

Both type and rate must be defined at the same time, otherwise the initiated mutation\_type and additional\_mutation\_rate will be used.

**Parameters**

**type:** type of mutation. Accepts 'point' (point mutation), 'insert' (insert a base), 'delete' (delete a base), 'invert' (invert a stretch of the chromosome), 'duplicate' (duplicate a stretch of the chromosome), 'translocate' (translocate a stretch of chromosome to another random position). Default = None.

**rate:** probability of mutation per base above background mutation rate. Default = None. No mutation event will ever happen if (rate + background\_mutation) is less than zero.

Overrides: copads.genetic.Organism.mutation\_scheme extit(inherited documentation)

***Inherited from copads.genetic.Organism(Section 12.4)***

`__str__()`, `clone()`, `getStatus()`, `setStatus()`

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__subclasshook__()`

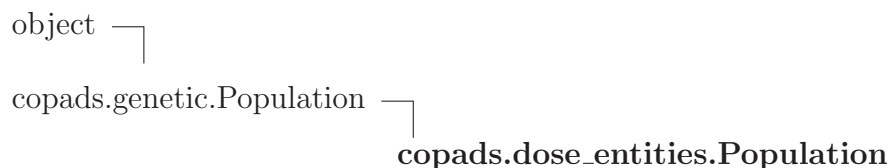
**8.2.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

### 8.2.3 Class Variables

Name	Description
cytoplasm	<b>Value:</b> <code>[0]* cytoplasm.size</code>
<i>Inherited from copads.genetic.Organism (Section 12.4)</i>	
genome, status	

## 8.3 Class Population



### 8.3.1 Methods

**`--init--`**(*self*, *pop\_size*=population\_size, *max\_gen*=maximum\_generations)

Establishes a population of organisms.

**Parameters**

**goal:** the goal to be reached by the population.

**maxgenerations:** maximum number of generations to evolve.  
Default = 'infinite'.

**agents:** organisms making up the initial population.

Overrides: `object.--init--` extit(inherited documentation)

**prepopulation\_control(*self*)**

Function to trigger population control events before mating event in each generation (For example, to simulate pre-puberty death). **This function may be over-ridden by the inherited class or substituted to cater for specific events.** Although this is not an absolute requirement, it is extremely encouraged to prevent exhaustion of memory space. Without population control, it will seems like a reproducing immortal population.

Here, the sample implementation eliminates the bottom half of the population based on fitness score unless the number of organisms is less than 20. This is to prevent extinction. However, if the number of organisms is more than 2000 (more than 100x initial population size, a random selection of 2000 will be used for the next generation.

Overrides: copads.genetic.Population.prepopulation\_control extit(inherited documentation)

**mating(*self*)**

Function to trigger mating events in each generation. **This function may be over-ridden by the inherited class or substituted to cater for specific mating schemes but not an absolute requirement to do so.** Mating schemes should include

- selection of mating partners using Organism.fitness() function and/or other status
- processes and actions of mating

Here, the sample implementation randomly selects any 2 organisms from the culled list and generate a new genome for the progeny organism by single crossover.

Overrides: copads.genetic.Population.mating extit(inherited documentation)

**postpopulation\_control(*self*)**

Function to trigger population control events after mating event in each generation(For example, to simulate old-age death). **This function may be over-ridden by the inherited class or substituted to cater for specific events.** Although this is not an absolute requirement, it is extremely encouraged to prevent exhaustion of memory space. Without population control, it will seems like a reproducing immortal population.

Overrides: copads.genetic.Population.postpopulation\_control extit(inherited documentation)

**generation\_events(*self*)**

Function to trigger other defined events in each generation. **This function may be over-ridden by the inherited class or substituted to cater for specific events but not an absolute requirement to do so.** Events and controls may include

- processes simulating disaster or other catastrophic events
- changes in mutations

Overrides: copads.genetic.Population.generation\_events extit(inherited documentation)

**report(*self*)**

Function to report the status of each generation. **This function may be over-ridden by the inherited class or substituted to cater for specific reporting schemes but not an absolute requirement to do so.** At the very least, this function should report whether the goal is reached.

**Return Value**

dictionary of status describing the current generation

Overrides: copads.genetic.Population.report extit(inherited documentation)

***Inherited from copads.genetic.Population(Section 12.5)***

add\_organism(), freeze(), generation\_step(), revive()

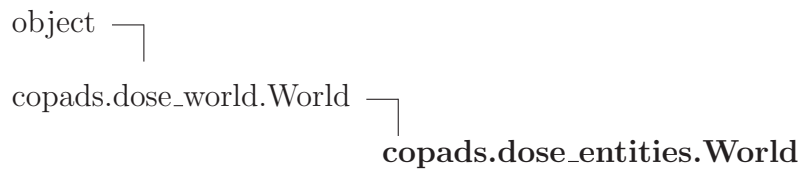
***Inherited from object***

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

**8.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
__class__	

## 8.4 Class World



### 8.4.1 Methods

```
__init__(self, world_x=world_x, world_y=world_y, world_z=world_z)
```

Setting up the world and ecosystem

**Parameters**

**world\_x**: number of ecological cells on the x-axis

**world\_y**: number of ecological cells on the y-axis

**world\_z**: number of ecological cells on the z-axis

Overrides: object.\_\_init\_\_ extit(inherited documentation)

```
organism_movement(self, x, y, z)
```

Function to trigger organism movement from current ecological cell to an adjacent ecological cell. **This function may be over-ridden by the inherited class or substituted to cater for mobility schemes but not an absolute requirement to do so.**

**Parameters**

**x**: location of current ecological cell on the x-axis

**y**: location of current ecological cell on the y-axis

**z**: location of current ecological cell on the z-axis

Overrides: copads.dose\_world.World.organism\_movement extit(inherited documentation)

**organism\_location**(*self*, *x*, *y*, *z*)

Function to trigger organism movement from current ecological cell to a distant ecological cell. **This function may be over-ridden by the inherited class or substituted to cater for mobility schemes but not an absolute requirement to do so.**

**Parameters**

**x**: location of current ecological cell on the x-axis

**y**: location of current ecological cell on the y-axis

**z**: location of current ecological cell on the z-axis

Overrides: copads.dose\_world.World.organism\_location extit(inherited documentation)

**ecoregulate**(*self*)

Function to simulate events to the entire ecosystem. **This function may be over-ridden by the inherited class or substituted to cater for ecological schemes but not an absolute requirement to do so.**

Overrides: copads.dose\_world.World.ecoregulate extit(inherited documentation)

**update\_ecology**(*self*, *x*, *y*, *z*)

Function to process temporary\_input and temporary\_output from the activities of the organisms in the current ecological cell into a local ecological cell condition, and update the ecosystem. **This function may be over-ridden by the inherited class or substituted to cater for ecological schemes but not an absolute requirement to do so.**

**Parameters**

**x**: location of current ecological cell on the x-axis

**y**: location of current ecological cell on the y-axis

**z**: location of current ecological cell on the z-axis

Overrides: copads.dose\_world.World.update\_ecology extit(inherited documentation)



**update\_local**(*self*, *x*, *y*, *z*)

Function to update local ecological cell condition from the ecosystem. **This function may be over-ridden by the inherited class or substituted to cater for ecological schemes but not an absolute requirement to do so.**

**Parameters**

**x**: location of current ecological cell on the x-axis

**y**: location of current ecological cell on the y-axis

**z**: location of current ecological cell on the z-axis

Overrides: *copads.dose\_world.World.update\_local* extit(inherited documentation)

**report**(*self*)

Function to report the status of the world and ecosystem. **This function may be over-ridden by the inherited class or substituted to cater for specific reporting schemes but not an absolute requirement to do so.**

**Return Value**

dictionary of status describing the current generation

Overrides: *copads.dose\_world.World.report* extit(inherited documentation)

***Inherited from copads.dose\_world.World(Section 11.2)***

*eco\_burial*(), *eco\_excavate*()

***Inherited from object***

*\_\_delattr\_\_*(), *\_\_format\_\_*(), *\_\_getattr\_\_*(), *\_\_hash\_\_*(), *\_\_new\_\_*(), *\_\_reduce\_\_*(), *\_\_reduce\_ex\_\_*(), *\_\_repr\_\_*(), *\_\_setattr\_\_*(), *\_\_sizeof\_\_*(), *\_\_str\_\_*(), *\_\_subclasshook\_\_*()

**8.4.2 Properties**

Name	Description
<i>Inherited from object</i>	
<i>__class__</i>	

**8.4.3 Class Variables**

Name	Description
<i>Inherited from copads.dose_world.World (Section 11.2)</i>	
<i>ecosystem</i>	

## 9 Module *copads.dose\_executor*

Default DOSE simulation runner in manuscript [1] Date created: 13th September 2012  
Licence: Python Software Foundation License version 2

[1] Ling, MHT. 2012. An Artificial Life Simulation Library Based on Genetic Algorithm, 3-Character Genetic Code and Biological Hierarchy. The Python Papers 7: 5.

### 9.1 Functions

```
set_instruction_version(ragaraja_version,  
instruction_set='ragaraja_instructions.txt')
```

Set active / usable set of Ragaraja instructions / operations for the current simulation.

#### Parameters

**ragaraja\_version:** defines the version of Ragaraja instructions to be used. Allowed values are {0 | 0.1 | 1}

**instruction\_set:** file name for user-defined instruction set usage. This will only be used when *ragaraja\_version*=0. Default = *ragaraja\_instructions.txt*. Please see *ragaraja\_instructions.txt* for format.  
(*type=string*)

#### Return Value

dictionary of ragaraja instructions as keys and instruction execution functions as values.

**Since:** version 0.4.1

```
write_parameters()
```

Write parameters into file.

**Since:** version 0.4.1

**simulate**(*entity\_module*)

Simulate the entities in DOSE.

**Parameters**

**entity\_module:** python module of entities to execute. Required classes in *entity\_module* are Chromosome (inherited from *genetic.Chromosome*), Organism (inherited from *genetic.Organism*), Population (inherited from *genetic.Population*) and World (inherited from *dose\_world.World*).

**Since:** version 0.4.1

## 9.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> 'copads'



Name	Description
population_size	Number of ecological cells in the world in (x,y,z) coordinate as (world_x, world_y, world_z). <b>Value:</b> 100
world_x	<b>Value:</b> 5
world_y	<b>Value:</b> 5
world_z	List of tuple indicating the location on ecosystem to map the population(s) and will be mapped in the same order as population_names. For example, if population_names = ['pop1', 'pop2'] and population_locations = [(0,0,0), (4,4,4)], population 'pop1' will be in (0,0,0) and population 'pop2' will be in (4,4,4). <b>Value:</b> 5
population_locations	Maximum number of generations to simulate. <b>Value:</b> [(0, 0, 0), (4, 4, 4)]
maximum_generations	Number of generations between each freezing/fossilization event. Freezing/fossilization event is similar to glycerol stocking in microbiology. <b>Value:</b> 500
fossilized_frequency	Proportion of population to freeze/fossilize. If the population size or the preserved proportion is below 100, the entire population will be preserved. <b>Value:</b> 100
fossilized_ratio	Dictionary containing prefix of file names for freezing/fossilization. The preserved sample will be written into a file with name in the following format - <prefix>.<generation count>.<sample size>.gap <b>Value:</b> 0.01
fossil_files	Number of generations between printing of reports (based on Population.report function) into files. The result file format is <UTC date time stamp> <current generation count> <output from Population.report function>. <b>Value:</b> {'pop1': 'pop1', 'pop2': 'pop2'}
print_frequency	Dictionary containing file names for result files. File name will be <file names>.result <b>Value:</b> 10

*continued on next page*

Name	Description
result_files	Version of Ragaraja instruction set to use. <b>Value:</b> {'pop1': 'pop1', 'pop2': 'pop2'}
ragaraja_version	File containing Ragaraja instruction set to be used. This option is only effective when ragaraja_version = 0. Format of file is <instruction>={Y N} where "Y" = instruction to be used and "N" = instruction not to be used. <b>Value:</b> 0.1
user_defined_instructions	Number of generations between each burial/preservation of ecosystem. <b>Value:</b> 'ragaraja_instructions.txt'
eco_buried_frequency	Prefix of file name for ecosystem burial/preservation. The preserved ecosystem will be written into a file with name in the following format - <prefix>.<generation count>.eco <b>Value:</b> 500
eco_burial_file	<b>Value:</b> 'eco'
__package__	<b>Value:</b> None

## 11 Module `copads.dose_world`

World structure for DOSE (digital organism simulation environment) Date created: 13th September 2012 Licence: Python Software Foundation License version 2

Reference: Ling, MHT. 2012. An Artificial Life Simulation Library Based on Genetic Algorithm, 3-Character Genetic Code and Biological Hierarchy. The Python Papers 7: 5.

### 11.1 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> <code>'copads'</code>

### 11.2 Class World

object └─  
           **`copads.dose_world.World`**

Representation of a 3-dimensional ecological world.

The ecosystem is made up of ecological cells. Each ecological cell is modelled as a dictionary of

- `local_input`: A list containing processed input, representing the partial local ecological condition, to be used as input to the organisms in the current ecological cell. This is updated by `World.update_local` function.
- `local_output`: A list containing processed output, representing the partial local ecological condition. This is updated by `World.update_local` function.
- `temporary_input`: A list acting as temporary holding for input after being fed to the organisms in the current ecological cell, which is to be used to update `local_input` and `local_output` lists by `World.update_local` and `World.update_ecology` functions.
- `temporary_output`: A list acting as temporary holding for output from the organisms in the current ecological cell, which is to be used to update `local_input` and `local_output` lists by `World.update_local` and `World.update_ecology` functions.
- `organisms`: The number of organisms in the current ecological cell which is updated by `World.organism_movement` and `World.organism_location` functions.

**See Also:** Ling, MHT. 2012. An Artificial Life Simulation Library Based on Genetic Algorithm, 3-Character Genetic Code and Biological Hierarchy. The Python Papers 7: 5.

### 11.2.1 Methods

**`__init__(self, world_x, world_y, world_z)`**

Setting up the world and ecosystem

**Parameters**

**world\_x:** number of ecological cells on the x-axis  
(*type=integer*)

**world\_y:** number of ecological cells on the y-axis  
(*type=integer*)

**world\_z:** number of ecological cells on the z-axis  
(*type=integer*)

Overrides: `object.__init__`

**`eco_burial(self, filename)`**

Function to preserve the entire ecosystem.

**Parameters**

**filename:** file name of preserved ecosystem.

**`eco_excavate(self, filename)`**

Function to excavate entire ecosystem.

**Parameters**

**filename:** file name of preserved ecosystem.

**`ecoregulate(self)`**

Function to simulate events to the entire ecosystem. **This function may be over-ridden by the inherited class or substituted to cater for ecological schemes but not an absolute requirement to do so.**



**organism\_movement**(*self*, *x*, *y*, *z*)

Function to trigger organism movement from current ecological cell to an adjacent ecological cell. **This function may be over-ridden by the inherited class or substituted to cater for mobility schemes but not an absolute requirement to do so.**

**Parameters**

**x**: location of current ecological cell on the x-axis

(*type=integer*)

**y**: location of current ecological cell on the y-axis

(*type=integer*)

**z**: location of current ecological cell on the z-axis

(*type=integer*)

**organism\_location**(*self*, *x*, *y*, *z*)

Function to trigger organism movement from current ecological cell to a distant ecological cell. **This function may be over-ridden by the inherited class or substituted to cater for mobility schemes but not an absolute requirement to do so.**

**Parameters**

**x**: location of current ecological cell on the x-axis

(*type=integer*)

**y**: location of current ecological cell on the y-axis

(*type=integer*)

**z**: location of current ecological cell on the z-axis

(*type=integer*)

**update\_ecology**(*self*, *x*, *y*, *z*)

Function to process temporary\_input and temporary\_output from the activities of the organisms in the current ecological cell into a local ecological cell condition, and update the ecosystem. **This function may be over-ridden by the inherited class or substituted to cater for ecological schemes but not an absolute requirement to do so.**

**Parameters**

**x**: location of current ecological cell on the x-axis

(*type=integer*)

**y**: location of current ecological cell on the y-axis

(*type=integer*)

**z**: location of current ecological cell on the z-axis

(*type=integer*)

**update\_local**(*self*, *x*, *y*, *z*)

Function to update local ecological cell condition from the ecosystem. **This function may be over-ridden by the inherited class or substituted to cater for ecological schemes but not an absolute requirement to do so.**

**Parameters**

**x**: location of current ecological cell on the x-axis

(*type=integer*)

**y**: location of current ecological cell on the y-axis

(*type=integer*)

**z**: location of current ecological cell on the z-axis

(*type=integer*)

**report**(*self*)

Function to report the status of the world and ecosystem. **This function may be over-ridden by the inherited class or substituted to cater for specific reporting schemes but not an absolute requirement to do so.**

**Return Value**

dictionary of status describing the current generation

***Inherited from object***

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

### 11.2.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

### 11.2.3 Class Variables

Name	Description
ecosystem	<b>Value:</b> {}

## 12 Module *copads.genetic*

Framework for Genetic Algorithm Applications. Date created: 23rd February 2010 Licence: Python Software Foundation License version 2

**See Also:** Lim, JZR, Aw, ZQ, Goh, DJW, How, JA, Low, SXZ, Loo, BZL, Ling, MHT. 2010. A genetic algorithm framework grounded in biology. The Python Papers Source Codes 2: 6.

### 12.1 Functions

<b>crossover</b> ( <i>chromosome1, chromosome2, position</i> )
Cross-over operator - swaps the data on the 2 given chromosomes after a given position.
<b>Parameters</b>
<i>chromosome1</i> : Chromosome object
<i>chromosome2</i> : Chromosome object
<i>position</i> : base position of the swap over ( <i>type=integer</i> )
<b>Return Value</b>
(resulting <i>chromosome1</i> , resulting <i>chromosome2</i> )
<b>Since:</b> version 0.4

```
population_constructor(data={'additional_mutation_rate': 0.01,
'background_mutation':...})
```

Function to construct a population based on a dictionary of population data.

Population data contains the following keys:

- 'nucleotide\_list' = List of allowable nucleotides (bases). Default = [1, 2, 3, 4].
- 'chromosome\_length' = Length of a chromosome. Default = 200.
- 'chromosome\_type' = Type of chromosome. Default = 'defined'.
- 'chromosome' = Initial chromosome. Default = [1] \* 200.
- 'background\_mutation' = Background mutation rate. Default = 0.0001 (0.01%).
- 'genome\_size' = Number of chromosomes per organism. Default = 1.
- 'population\_size' = Size of initial population (number of organisms). Default = 200.
- 'fitness\_function' = Fitness evaluation function. Accepts 'default' or a function. Please refer to Organism. Default = 'default'.
- 'mutation\_scheme' = Function simulating the mutation scheme of an organism. Accepts 'default' or a function. Please refer to Organism. Default = 'default'.
- 'additional\_mutation\_rate' = Mutation rate on top of background mutation rate. Default = 0.01 (1%).
- 'mutation\_type' = Type of default mutation. Default = 'point'.
- 'goal' = Goal of the population, as evaluated by fitness function. Default = 4.
- 'maximum\_generation' = Number of generations to simulate. Accepts an integer or 'infinite'. Default = 'infinite'.
- 'prepopulation\_control' = Function simulating pre-mating population control. Please refer to Population. Default = 'default'.
- 'mating' = Function simulating mating procedure (mate selection and act of mating control. Please refer to Population. Default = 'default'.
- 'postpopulation\_control' = Function simulating post-mating population control. Please refer to Population. Default = 'default'.
- 'generation\_events' = Function simulating other possible (usually rare or random) events in the generation. Please refer to Population. Default = 'default'.
- 'report' = Function to generate the status report of the generation. Please refer to Population. Default = 'default'.

### Parameters

**data:** population data

(*type=dictionary*)

77

### Return Value

Population object

**See Also:** Lim, JZR, Aw, ZQ, Goh, DJW, How, JA, Low, SXZ, Loo, BZL,

```
population_simulate(population, printfreq=100, freezefreq='never',
freezefile='pop', freezeportion=0.01, resultfile='result.txt')
```

Function to simulate the population - start the GA.

#### Parameters

**population:** Population to run.  
(*type=Population object*)

**printfreq:** Reporting intervals on screen. Default = 100.

**freezefreq:** Generation intervals to freeze population into a file. See Population.freeze method. Accepts an integer or 'never'. Default = 'never'.

**freezefile:** Prefix of file name of frozen population. See Population.freeze method. Default = 'pop'.

**freezeportion:** Proportion of population to be preserved. See Population.freeze method. Default = 0.01, preserves 1% of the population.

**resultfile:** Name of file to print out results of each generation. Format of output is dependent on reporting method of the population (Population.report). If 'None', it will print out the results into a file. Default = 'result.txt'.

**See Also:** Lim, JZR, Aw, ZQ, Goh, DJW, How, JA, Low, SXZ, Loo, BZL, Ling, MHT. 2010. A genetic algorithm framework grounded in biology. The Python Papers Source Codes 2: 6.

**Since:** version 0.4

## 12.2 Variables

Name	Description
population_data	<b>Value:</b> {'additional_mutation_rate': 0.01, 'background_mutation': ...}
__package__	<b>Value:</b> 'copads'

## 12.3 Class Chromosome

object └─  
          copads.genetic.Chromosome

Representation of a linear chromosome.

**See Also:** Lim, JZR, Aw, ZQ, Goh, DJW, How, JA, Low, SXZ, Loo, BZL, Ling, MHT. 2010. A genetic algorithm framework grounded in biology. The Python Papers Source Codes 2: 6.

**Since:** version 0.4

### 12.3.1 Methods

**`__init__(self, sequence, base, background_mutation=0.0001)`**

Sets up a chromosome.

**Parameters**

<b>sequence:</b>	a subscriptable object (list or string) representing the sequence of the chromosome.
<b>base:</b>	a subscriptable object (list or string) representing allowable entities in the sequence.
<b>background_mutation:</b>	background mutation rate represented as the probability of number of mutations per base. Default = 0.0001 (0.01%).

Overrides: `object.__init__`

**Since:** version 0.4

---

**rmutate**(*self*, *type*='point', *rate*=0.01, *start*=0, *end*=-1)

Random Mutation operator - to simulate random point, insertion, deletion, inversion, gene translocation and gene duplication events.

The start and end parameters are useful for simulating mutational hotspots in the genome.

**Parameters**

- type:** type of mutation. Accepts 'point' (point mutation), 'insert' (insert a base), 'delete' (delete a base), 'invert' (invert a stretch of the chromosome), 'duplicate' (duplicate a stretch of the chromosome), 'translocate' (translocate a stretch of chromosome to another random position). Default = point.
- rate:** probability of mutation per base above background mutation rate. Default = 0.01 (1%). No mutation event will ever happen if (rate + background\_mutation) is less than zero.
- start:** starting base on the sequence for mutation. Default = 0, start of the genome.
- end:** last base on the sequence for mutation. Default = -1, end of the genome.

**Since:** version 0.4



**kmutate**(*self*, *type*='point', *start*=0, *end*=0, *sequence*=None, *tpos*=0)

Known Mutation operator - to simulate a known point, insertion, deletion, inversion, gene translocation or gene duplication event.

The required parameters will be determined by the type of mutation:

- point requires
  - start (the position of the base to mutate)
  - sequence (the new base)
- delete requires
  - start (the position of the base to delete)
- insert requires
  - start (the position of the base to begin insertion)
  - sequence (list of sequence or a base to insert)
- invert requires
  - start (the position of the base to start inversion)
  - end (the position of the base to end inversion)
- duplicate requires
  - start (the position of the starting base to duplicate)
  - end (the position of the last base to duplicate)
- translocate requires
  - start (the position of the base to start translocation)
  - end (the position of the base to end translocation)
  - tpos (the position of the base insert the translocated sequence)

#### Parameters

- type:** type of mutation. Accepts 'point' (point mutation), 'insert' (insert a base), 'delete' (delete a base), 'invert' (invert a stretch of the chromosome), 'duplicate' (duplicate a stretch of the chromosome), 'translocate' (translocate a stretch of chromosome to another random position). Default = point.
- start:** starting base on the chromosome for mutation. Default = 0, start of the genome.
- end:** last base on the chromosome for mutation. Default = 0.
- sequence:** list of bases to change to (point mutation) or to insert (insertion mutation)
- tpos:** position of the chromosome to insert the translocated sequence.

**Since:** version 0.4

**replicate(*self*)**

Replicates (deep copy) the chromosome.

**Return Value**

a copy of current chromosome.

**Since:** version 0.4

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**12.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**12.4 Class Organism**

An organism represented by a list of chromosomes and a status table. This class should almost never be instantiated on its own but acts as an ancestor class as some functions need to be over-ridden in the inherited class for the desired use.

Each organism is identifiable by a randomly generated 32-character 'name' as `Organism.identity`.

Methods to be over-ridden in the inherited class or substituted are

- `fitness`
- `mutation_scheme`

Pre-defined status are

1. `alive` - is the organism alive? True or False.
2. `vitality` - percentage of maximum vitality.
3. `age` - the current age of the organism.
4. `lifespan` - pre-defined maximum lifespan to be set based on scenario.
5. `fitness` - how fit the organism is? Set to maximum fitness.
6. `death` - reason of death (as death code).

List of defined death codes

1. death01 - zero vitality
2. death02 - maximum age reached
3. death03 - unknown death cause

**See Also:** Lim, JZR, Aw, ZQ, Goh, DJW, How, JA, Low, SXZ, Loo, BZL, Ling, MHT. 2010. A genetic algorithm framework grounded in biology. The Python Papers Source Codes 2: 6.

**Since:** version 0.4

## 12.4.1 Methods

```
__init__(self, genome='default', mutation_type='point',
additional_mutation_rate=0.01, gender=None)
```

Sets up a new organism with default status (age = 0, vitality = 100, lifespan = 100, fitness = 100, alive = True)

**Parameters**

<b>genome:</b>	list of chromosomes to inherit. Default = 'default', which will set up one default chromosome. It also allows a 'dummy' chromosome which is basically a one-base chromosome - this is for applications which does not utilize the chromosome.
<b>mutation_type:</b>	type of mutation. Accepts 'point' (point mutation), 'insert' (insert a base), 'delete' (delete a base), 'invert' (invert a stretch of the chromosome), 'duplicate' (duplicate a stretch of the chromosome), 'translocate' (translocate a stretch of chromosome to another random position). Default = point.
<b>additional_mutation_rate:</b>	probability of mutation per base above background mutation rate. Default = 0.01 (1%). No mutation event will ever happen if (rate + background_mutation) is less than zero.
<b>gender:</b>	establishes the gender of the organism which may be used for mating routines.

Overrides: object.\_\_init\_\_

**Since:** version 0.4

**fitness(*self*)**

Function to calculate the fitness of the current organism. **This function MUST be over-ridden by the inherited class or substituted as fitness function is highly dependent on utility.** The only requirement is that a fitness score must be returned.

Here, the sample implementation calculates fitness as proportion of the genome with '1's.

**Return Value**

fitness score or fitness list

**Since:** version 0.4

**mutation\_scheme(*self*, *type*=None, *rate*=None)**

Function to trigger mutation events in each chromosome. **This function may be over-ridden by the inherited class or substituted to cater for specific mutation schemes but not an absolute requirement to do so.**

Both type and rate must be defined at the same time, otherwise the initiated mutation\_type and additional\_mutation\_rate will be used.

**Parameters**

**type:** type of mutation. Accepts 'point' (point mutation), 'insert' (insert a base), 'delete' (delete a base), 'invert' (invert a stretch of the chromosome), 'duplicate' (duplicate a stretch of the chromosome), 'translocate' (translocate a stretch of chromosome to another random position). Default = None.

**rate:** probability of mutation per base above background mutation rate. Default = None. No mutation event will ever happen if (rate + background\_mutation) is less than zero.

**Since:** version 0.4

**setStatus**(*self*, *variable*, *value*)

Sets new status or change status of the organism. However, the following status change will result in death of the organism

1. 'alive' to False
2. 'vitality' to or below zero
3. 'age' to or above lifespan

**Parameters**

**variable:** name of status to change  
**value:** new value of the status

**Since:** version 0.4

**getStatus**(*self*, *variable*)

Returns a status variable of the current organism

**Parameters**

**variable:** name of the status variable

**Return Value**

status or a KeyError if status is not found

**Since:** version 0.4

**\_\_str\_\_**(*self*)

Returns the genome of the organism

Overrides: object.\_\_str\_\_

**clone**(*self*)

Cloness (deep copy) the organism.

**Return Value**

a copy of current organism.

**Since:** version 0.4

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__subclasshook__()`

**12.4.2 Properties**

*continued on next page*

Name	Description
<b>Name</b>	<b>Description</b>
<i>Inherited from object</i>	
<code>__class__</code>	

### 12.4.3 Class Variables

Name	Description
status	<b>Value:</b> {'age': 0.0, 'alive': True, 'death': None, 'fitness': 100...
genome	<b>Value:</b> []

## 12.5 Class Population



Representation of a population as a list of organisms. The entire population is in memory.

Methods to be over-ridden in the inherited class or substituted are

- `prepopulation_control`
- `mating`
- `postpopulation_control`
- `generation_events`
- `report`

**See Also:** Lim, JZR, Aw, ZQ, Goh, DJW, How, JA, Low, SXZ, Loo, BZL, Ling, MHT. 2010. A genetic algorithm framework grounded in biology. The Python Papers Source Codes 2: 6.

**Since:** version 0.4

### 12.5.1 Methods

**\_\_init\_\_**(*self*, *goal*, *maxgenerations*=*'infinite'*, *agents*=[])

Establishes a population of organisms.

**Parameters**

**goal:** the goal to be reached by the population.  
(*type*=the return type of *Organism.fitness()*)

**maxgenerations:** maximum number of generations to evolve.  
Default = *'infinite'*.

**agents:** organisms making up the initial population.  
(*type*=list of *Organism* objects)

Overrides: object.\_\_init\_\_

**Since:** version 0.4

**prepopulation\_control**(*self*)

Function to trigger population control events before mating event in each generation (For example, to simulate pre-puberty death). **This function may be over-ridden by the inherited class or substituted to cater for specific events.** Although this is not an absolute requirement, it is extremely encouraged to prevent exhaustion of memory space. Without population control, it will seem like a reproducing immortal population.

Here, the sample implementation eliminates the bottom half of the population based on fitness score unless the number of organisms is less than 20. This is to prevent extinction. However, if the number of organisms is more than 2000 (more than 100x initial population size, a random selection of 2000 will be used for the next generation.

**Since:** version 0.4



**mating(*self*)**

Function to trigger mating events in each generation. **This function may be over-ridden by the inherited class or substituted to cater for specific mating schemes but not an absolute requirement to do so.** Mating schemes should include

- selection of mating partners using Organism.fitness() function and/or other status
- processes and actions of mating

Here, the sample implementation randomly selects any 2 organisms from the culled list and generate a new genome for the progeny organism by single crossover.

**Since:** version 0.4

**postpopulation\_control(*self*)**

Function to trigger population control events after mating event in each generation (For example, to simulate old-age death). **This function may be over-ridden by the inherited class or substituted to cater for specific events.** Although this is not an absolute requirement, it is extremely encouraged to prevent exhaustion of memory space. Without population control, it will seem like a reproducing immortal population.

**Since:** version 0.4

**generation\_events(*self*)**

Function to trigger other defined events in each generation. **This function may be over-ridden by the inherited class or substituted to cater for specific events but not an absolute requirement to do so.** Events and controls may include

- processes simulating disaster or other catastrophic events
- changes in mutations

**Since:** version 0.4

**report(*self*)**

Function to report the status of each generation. **This function may be over-ridden by the inherited class or substituted to cater for specific reporting schemes but not an absolute requirement to do so.** At the very least, this function should report whether the goal is reached.

**Return Value**

dictionary of status describing the current generation

**Since:** version 0.4

**generation\_step(*self*)**

Function to simulate events for one generation. These includes

- mating (according to the mating scheme or function)
- mutating each organism in the population
- population size control
- other events defined under generation\_events function
- increment of generation count
- reporting the population status

**Return Value**

information returned from report function.

**Since:** version 0.4

**add\_organism(*self*, *organism*)**

Add a new organism(s) to the population.

**Parameters**

**organism:** list of new Organism object(s)

**Since:** version 0.4

**freeze(*self*, *prefix*='pop', *proportion*=0.01)**

Preserves part or the entire population. If the population size or the preserved proportion is below 100, the entire population will be preserved. The preserved sample will be written into a file with name in the following format - <prefix><generation count>\_<sample size>.gap

**Parameters**

**prefix:** prefix of file name. Default = 'pop'.

**proportion:** proportion of population to be preserved. Default = 0.01, preserves 1% of the population.

**Since:** version 0.4

**revive**(*self*, *filename*, *type*='replace')

Revives a frozen population.

**Parameters**

**filename:** file name of frozen population (generated by Population.freeze() function)

**type:** type of revival. Allows 'replace' (replace the current population with the revived population) or 'add' (add the revived population to the current population). Default = 'replace'.

**Since:** version 0.4

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**12.5.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 13 Module copads.graph

Graph Data Structures and Algorithms.

Copyright (c) Maurice H.T. Ling <mauriceling@acm.org>

Date created: 17th August 2005

### 13.1 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'copads'

### 13.2 Class Graph

Graph data structure

#### 13.2.1 Methods

<b><code>--init--</code></b> ( <i>self</i> , <i>**kwarg</i> )
Initialization method. It can accept the following keyword parameters:
<b>Parameters</b>
<b>adjacency:</b> adjacency matrix, with vertices as the first row - row count is 1 more than column count <i>(type=list of list)</i>
<b>digraph:</b> state whether the input values is to construct a graph (True for directional graph). Default = False. <i>(type=boolean)</i>
<b>edges:</b> list of edges where each tuple is (<source>, <destination>). Uses digraph parameter. <i>(type=list of 2-element tuples)</i>
<b>graph:</b> graph as {<source> : <destination dictionary>} where <destination dictionary> ::= {<destination> : <attribute>}. <i>(type=dictionary of dictionary)</i>
<b>vertices:</b> vertices (nodes). <i>(type=list)</i>

**makeGraphFromAdjacency(*self*, *adj*)**

Constructs a graph from an adjacency (*adj*) matrix, which is given as a list of list (rows). The first row of the matrix contains a list of the vertices; hence, there will be *n*+1 rows and *n*-columns in the given matrix.

**Parameters**

**adj:** adjacency matrix, with vertices as the first row - row count is 1 more than column count  
*(type=list of list)*

**makeGraphFromVertices(*self*, *vertices*)**

Initialize a list of nodes (vertices) without edges.

**Parameters**

**vertices:** list of vertices

**Status:** Tested method

**Since:** version 0.1

**makeGraphFromEdges1(*self*, *edges*)**

Constructs a directional graph from edges (a list of tuple). Each tuple contains 2 vertices. For example, P -> Q is written as ('P', 'Q').

**Parameters**

**edges:** edges  
*(type=list of 2-element tuple)*

**Status:** Tested method

**Since:** version 0.1

**makeGraphFromEdges2(*self*, *edges*)**

Constructs an un-directional graph from edges (a list of tuple). Each tuple contains 2 vertices. An un-directional graph is implemented as a directional graph where each edges runs both directions.

**Parameters**

**edges:** list of edges  
*(type=list of 2-element tuples)*

**isVertices**(*self*, *vlist*)

Checks whether each element in *vlist* is a vertex (node) of the graph.

**Parameters**

**vlist:** list of vertices

**Return Value**

dictionary of <element of *vlist*> : <True | False>

**Dijkstra**(*self*, *start*, *end*=None)

Find shortest paths from the start vertex to all vertices nearer than or equal to the end.

Dijkstra's algorithm is only guaranteed to work correctly when all edge lengths are positive. This code does not verify this property for all edges (only the edges seen before the end vertex is reached), but will correctly compute shortest paths even for some graphs with negative edges, and will raise an exception if it discovers that a negative edge has caused it to make a mistake.

**Parameters**

**start:** vertex of starting point

**end:** vertex of ending point

**Status:** Tested method (by proxy from testing `shortestPath` method)

**Since:** version 0.1

**shortestPath**(*self*, *start*, *end*)

Find a single shortest path from the given start vertex to the given end vertex. The output is a list of the vertices in order along the shortest path.

**Parameters**

**start:** vertex of starting point

**end:** vertex of ending point

**Status:** Tested method

**Since:** version 0.1

**RandomGraph**(*self*, *nodes*, *edges*, *maxweight*=100.0)

Generates a graph of random edges.

**Parameters**

**nodes:** list of nodes or number of nodes in the random graph  
**edges:** number of edges to generate in the random graph  
*(type=integer)*  
**maxweight:** maximum weight of each edge. default = 100.0  
*(type=float)*

### 13.2.2 Class Variables

Name	Description
graph	Value: {}

## 14 Module *copads.hash*

Hash Generators Date created: 16th April 2013 Licence: Python Software Foundation License version 2

### 14.1 Functions

**forward\_file\_hash**(*f, fsize, start, blocksize, algorithm*)

Forward file hashing. It takes file and generate repeated hashes to the end of the file in blocks.

Algorithm:

1. block <- file from position N to N + blocksize
2. hash <- generate hash from block
3. N = N + blocksize
4. block <- file from position N to N + blocksize
5. block = block + hash
6. hash <- generate hash from block
7. Repeat steps 3 to 6 until end of file
8. Return hash

**Parameters**

**f:** file handle of file to hash

**fsize:** file size of f  
(*type=integer*)

**start:** file location to start hashing  
(*type=integer*)

**blocksize:** block size for hash generation  
(*type=integer*)

**algorithm:** algorithm to generate hash. Allowable options are {md5|sha1|sha244|sha256|sha384|sha512}  
(*type=strong*)

**Return Value**

generated hash in string



**backward\_file\_hash**(*f, end, blocksize, algorithm*)

Forward file hashing. It takes file and generate repeated hashes to the end of the file in blocks.

Algorithm:

1. block <- file from position N - blocksize to N
2. hash <- generate hash from block
3. N = N - blocksize
4. block <- file from position N - blocksize to N
5. block = block + hash
6. hash <- generate hash from block
7. Repeat steps 3 to 6 until start of file
8. Return hash

**Parameters**

- f:** file handle of file to hash
- end:** file location to start hashing  
(*type=integer*)
- blocksize:** block size for hash generation  
(*type=integer*)
- algorithm:** algorithm to generate hash. Allowable options are  
{md5|sha1|sha244|sha256|sha384|sha512}  
(*type=strong*)

**Return Value**

generated hash in string

**cfh**(*filename*, *blocksize*=1024, *startpoints*=10, *algorithm*='md5')

Circular file hasher. This function uses repeated forward and backward hash functions to generate hash for the entire file. Hence, a byte in the file will be hashed multiple times. The length of hash generated will be dependent on the block size and number of start points.

Algorithm:

1. forward hash <- hash(start) + [hash(start + 1) for start = 0 to startpoints]
2. backward hash <- hash(start) + [hash(start - 1) for start = startpoints to 0]
3. Return forward hash + backward hash

#### Parameters

- filename:** name of file to hash  
(*type=string*)
- blocksize:** block size for hash generation. Default = 1024  
(*type=integer*)
- startpoints:** defines the number of start point on the file for hash generation. Default = 10.  
(*type=integer*)
- algorithm:** algorithm to generate hash. Allowable options are {md5|sha1|sha244|sha256|sha384|sha512}  
(*type=strong*)

#### Return Value

generated hash in string

## 14.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'copads'

## 15 Module `copads.hypothesis`

Statistical Hypothesis Testing Routines.

Each routine will Return a 5-element list

critical, right result

where

- left result = True (statistic in lower critical region) or False (statistic not in lower critical region)
- left critical = lower critical value generated from 1 - confidence
- statistic = calculated statistic value
- right critical = upper critical value generated from confidence
- right result = True (statistic in upper critical region) or False (statistic not in upper critical region)

References

- Test 1-100: Gopal K. Kanji. 2006. 100 Statistical Tests, 3rd edition. Sage Publications.

Copyright (c) Maurice H.T. Ling <mauriceling@acm.org>

Date created: 1st September 2008

## 15.1 Functions

**test**(*statistic, distribution, confidence*)

Generates the critical value from distribution and confidence value using the distribution's inverseCDF method and performs 1-tailed and 2-tailed test by comparing the calculated statistic with the critical value.

Returns a 5-element list [left result, left critical, statistic, right critical, right result] where

- left result = True (statistic in lower critical region) or False (statistic not in lower critical region)
- left critical = lower critical value generated from 1 - confidence
- statistic = calculated statistic value
- right critical = upper critical value generated from confidence
- right result = True (statistic in upper critical region) or False (statistic not in upper critical region)

Therefore, null hypothesis is accepted if left result and right result are both False in a 2-tailed test.

### Parameters

**statistic:** calculated statistic (float)

**distribution:** distribution to calculate critical value  
(*type=instance of a statistics distribution*)

**confidence:** confidence level of a one-tail test (usually 0.95 or 0.99), use 0.975 or 0.995 for 2-tail test  
(*type=float of less than 1.0*)

**Z1Mean1Variance**(*smean*, *pmean*, *pvar*, *ssize*, *confidence*)

Test 1: Z-test for a population mean (variance known)

To investigate the significance of the difference between an assumed population mean and sample mean when the population variance is known.

Limitations

1. Requires population variance (use Test 7 if population variance unknown)

**Parameters**

**smean:** sample mean  
**pmean:** population mean  
**pvar:** population variance  
**ssize:** sample size  
**confidence:** confidence level

**See Also:** Ling, MHT. 2009. Ten Z-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 1:5

**Status:** Tested function

**Since:** version 0.1

**Z2Mean1Variance**(*smean1*, *smean2*, *pvar*, *ssize1*, *ssize2*, *confidence*,  
*pmean1*=0.0, *pmean2*=0.0)

Test 2: Z-test for two population means (variances known and equal)

To investigate the significance of the difference between the means of two samples when the variances are known and equal.

Limitations

1. Population variances must be known and equal (use Test 8 if population variances unknown)

#### Parameters

**smean1:** sample mean of sample #1  
**smean2:** sample mean of sample #2  
**pvar:** variances of both populations (variances are equal)  
**ssize1:** sample size of sample #1  
**ssize2:** sample size of sample #2  
**confidence:** confidence level  
**pmean1:** population mean of population #1 (optional)  
**pmean2:** population mean of population #2 (optional)

**See Also:** Ling, MHT. 2009. Ten Z-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 1:5

**Status:** Tested function

**Since:** version 0.1

**Z2Mean2Variance**(*smean1*, *smean2*, *pvar1*, *pvar2*, *ssize1*, *ssize2*, *confidence*, *pmean1*=0.0, *pmean2*=0.0)

Test 3: Z-test for two population means (variances known and unequal)

To investigate the significance of the difference between the means of two samples when the variances are known and unequal.

Limitations

1. Population variances must be known (use Test 9 if population variances unknown)

#### Parameters

**smean1:** sample mean of sample #1  
**smean2:** sample mean of sample #2  
**pvar1:** variance of population #1  
**pvar2:** variance of population #2  
**ssize1:** sample size of sample #1  
**ssize2:** sample size of sample #2  
**confidence:** confidence level  
**pmean1:** population mean of population #1 (optional)  
**pmean2:** population mean of population #2 (optional)

**See Also:** Ling, MHT. 2009. Ten Z-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 1:5

**Status:** Tested function

**Since:** version 0.4

**Z1Proportion**(*spro*, *ppro*, *ssize*, *confidence*)

Test 4: Z-test for a proportion (binomial distribution)

To investigate the significance of the difference between an assumed proportion and an observed proportion.

Limitations

1. Requires sufficiently large sample size to use Normal approximation to binomial

**Parameters**

**spro:** sample proportion  
**ppro:** population proportion  
**ssize:** sample size  
**confidence:** confidence level

**See Also:** Ling, MHT. 2009. Ten Z-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 1:5

**Status:** Tested function

**Since:** version 0.1

**Z2Proportion**(*spro1*, *spro2*, *ssize1*, *ssize2*, *confidence*)

Test 5: Z-test for the equality of two proportions (binomial distribution) To investigate the assumption that the proportions of elements from two populations are equal, based on two samples, one from each population.

Limitations

1. Requires sufficiently large sample size to use Normal approximation to binomial

**Parameters**

**spro1:** sample proportion #1  
**spro2:** sample proportion #2  
**ssize1:** sample size #1  
**ssize2:** sample size #2  
**confidence:** confidence level

**See Also:** Ling, MHT. 2009. Ten Z-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 1:5

**Status:** Tested function

**Since:** version 0.1



**Z2Count**(*time1*, *time2*, *count1*, *count2*, *confidence*)

Test 6: Z-test for comparing two counts (Poisson distribution)

To investigate the significance of the differences between two counts.

Limitations

1. Requires sufficiently large sample size to use Normal approximation to binomial

**Parameters**

**time1:** first measurement time  
**time2:** second measurement time  
**count1:** counts at first measurement time  
**count2:** counts at second measurement time  
**confidence:** confidence level

**See Also:** Ling, MHT. 2009. Ten Z-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 1:5

**Status:** Tested function

**Since:** version 0.1

**t1Mean**(*smean*, *pmean*, *svar*, *ssize*, *confidence*)

Test 7: t-test for a population mean (population variance unknown)

To investigate the significance of the difference between an assumed population mean and a sample mean when the population variance is unknown and cannot be assumed equal or not equal.

Limitations

1. Weaker form of Test 1

**Parameters**

**smean:** sample mean  
**pmean:** population mean  
**svar:** sample variance  
**ssize:** sample size  
**confidence:** confidence level

**See Also:** Chay, ZE, Ling, MHT. 2010. COPADS, II: Chi-Square test, F-Test and t-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 2:3

**Status:** Tested function

**Since:** version 0.4

**t2Mean2EqualVariance**(*smean1*, *smean2*, *svar1*, *svar2*, *ssize1*, *ssize2*, *confidence*, *pmean1*=0.0, *pmean2*=0.0)

Test 8: t-test for two population means (population variance unknown but equal)

To investigate the significance of the difference between the means of two populations when the population variances are unknown but assumed equal.

Limitations

1. Weaker form of Test 2

**Parameters**

**smean1:** sample mean of sample #1  
**smean2:** sample mean of sample #2  
**svar1:** variances of sample #1  
**svar2:** variances of sample #2  
**ssize1:** sample size of sample #1  
**ssize2:** sample size of sample #2  
**confidence:** confidence level  
**pmean1:** population mean of population #1 (optional)  
**pmean2:** population mean of population #2 (optional)

**See Also:** Chay, ZE, Ling, MHT. 2010. COPADS, II: Chi-Square test, F-Test and t-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 2:3

**Status:** Tested function

**Since:** version 0.4

**t2Mean2UnequalVariance**(*smean1*, *smean2*, *svar1*, *svar2*, *ssize1*, *ssize2*, *confidence*, *pmean1*=0.0, *pmean2*=0.0)

Test 9: t-test for two population means (population variance unknown and unequal)

To investigate the significance of the difference between the means of two populations when the population variances are unknown and unequal.

Limitations

1. Weaker form of Test 3

**Parameters**

**smean1:** sample mean of sample #1  
**smean2:** sample mean of sample #2  
**svar1:** variances of sample #1  
**svar2:** variances of sample #2  
**ssize1:** sample size of sample #1  
**ssize2:** sample size of sample #2  
**confidence:** confidence level  
**pmean1:** population mean of population #1 (optional)  
**pmean2:** population mean of population #2 (optional)

**See Also:** Chay, ZE, Ling, MHT. 2010. COPADS, II: Chi-Square test, F-Test and t-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 2:3

**Status:** Tested function

**Since:** version 0.4

**tPaired**(*smean1*, *smean2*, *svar*, *ssize*, *confidence*)

Test 10: t-test for two population means (method of paired comparisons)

To investigate the significance of the difference between two population means when no assumption is made about the population variances.

**Parameters**

**smean1:** sample mean of sample #1  
**smean2:** sample mean of sample #2  
**svar:** variance of differences between pairs  
**ssize:** sample size  
**confidence:** confidence level

**See Also:** Chay, ZE, Ling, MHT. 2010. COPADS, II: Chi-Square test, F-Test and t-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 2:3

**Status:** Tested function

**Since:** version 0.4

**tRegressionCoefficient**(*variancex*, *varianceyx*, *b*, *ssize*, *confidence*)

Test 11: t-test of a regression coefficient

To investigate the significance of the regression coefficient.

**Limitations**

1. Homoeasticity of values

**Parameters**

**variancex:** variance of x  
**varianceyx:** variance of yx  
**b:** calculated Regression Coefficient  
**ssize:** sample size  
**confidence:** confidence level

**See Also:** Chay, ZE, Ling, MHT. 2010. COPADS, II: Chi-Square test, F-Test and t-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 2:3

**Status:** Tested function

**Since:** version 0.4

**tPearsonCorrelation**(*r*, *ssize*, *confidence*)

Test 12: t-test of a correlation coefficient

To investigate whether the difference between the sample correlation coefficient and zero is statistically significant.

Limitations

1. Assumes population correlation coefficient to be zero (use Test 13 for testing other population correlation coefficient)
2. Assumes a linear relationship (regression line as  $Y = MX + C$ )
3. Independence of x-values and y-values

Use Test 59 when these conditions cannot be met

**Parameters**

**r:** calculated Pearson's product-moment correlation coefficient  
**ssize:** sample size  
**confidence:** confidence level

**See Also:** Chay, ZE, Ling, MHT. 2010. COPADS, II: Chi-Square test, F-Test and t-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 2:3

**Status:** Tested function

**Since:** version 0.4

**ZPearsonCorrelation**(*sr, pr, ssize, confidence*)

Test 13: Z-test of a correlation coefficient

To investigate the significance of the difference between a correlation coefficient and a specified value.

Limitations

1. Assumes a linear relationship (regression line as  $Y = MX + C$ )
2. Independence of x-values and y-values

Use Test 59 when these conditions cannot be met

**Parameters**

**sr:** calculated sample Pearson's product-moment correlation coefficient

**pr:** specified Pearson's product-moment correlation coefficient to test

**ssize:** sample size

**confidence:** confidence level

**See Also:** Ling, MHT. 2009. Ten Z-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 1:5

**Status:** Tested function

**Since:** version 0.1

**Z2PearsonCorrelation**(*r1, r2, ssize1, ssize2, confidence*)

Test 14: Z-test for two correlation coefficients

To investigate the significance of the difference between the correlation coefficients for a pair variables occurring from two difference populations.

**Parameters**

**r1:** Pearson correlation coefficient of sample #1

**r2:** Pearson correlation coefficient of sample #2

**ssize1:** Sample size #1

**ssize2:** Sample size #2

**confidence:** confidence level

**See Also:** Ling, MHT. 2009. Ten Z-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 1:5

**Status:** Tested function

**Since:** version 0.1

**ChiSquarePopVar**(*values, ssize, pv, confidence=0.95*)

Test 15: Chi-square test for a population variance

To investigate the difference between a sample variance and an assumed population variance.

**Parameters**

**values:** sample values  
**ssize:** sample size  
**pv:** population variance  
**confidence:** confidence level

**See Also:** Chay, ZE, Ling, MHT. 2010. COPADS, II: Chi-Square test, F-Test and t-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 2:3

**Status:** Tested function

**Since:** version 0.4

**FVarianceRatio**(*var1, var2, ssize1, ssize2, confidence*)

Test 16: F-test for two population variances (variance ratio test)

To investigate the significance of the difference between two population variances.

**Parameters**

**var1:** variance #1  
**var2:** variance #2  
**ssize1:** sample size #1  
**ssize2:** sample size #2  
**confidence:** confidence level

**See Also:** Chay, ZE, Ling, MHT. 2010. COPADS, II: Chi-Square test, F-Test and t-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 2:3

**Status:** Tested function

**Since:** version 0.4



**F2CorrelatedObs**(*r*, *var1*, *var2*, *ssize1*, *ssize2*, *confidence*)

Test 17: F-test for two population variances (with correlated observations)

To investigate the difference between two population variances when there is correlation between the pairs of observations.

**Parameters**

**r:** Sample correlation value  
**var1:** variance #1  
**var2:** variance #2  
**ssize1:** sample size #1  
**ssize2:** sample size #2  
**confidence:** confidence level

**See Also:** Chay, ZE, Ling, MHT. 2010. COPADS, II: Chi-Square test, F-Test and t-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 2:3

**Status:** Tested function

**Since:** version 0.4

**ZCorrProportion**(*ssize*, *ny*, *yn*, *confidence*)

Test 23: Z-test for correlated proportions

To investigate the significance of the difference between two correlated proportions in opinion surveys. It can also be used for more general applications.

Limitations

1. The same people are questioned both times (correlated property).
2. Sample size must be quite large.

**Parameters**

**ssize:** sample size

**ny:** number answered 'no' in first poll and 'yes' in second poll

**yn:** number answered 'yes' in first poll and 'no' in second poll

**confidence:** confidence level

**See Also:** Ling, MHT. 2009. Ten Z-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 1:5

**Status:** Tested function

**Since:** version 0.1

**Chisq2Variance**(*ssize*, *svar*, *pvar*, *confidence*)

Test 24: Chi-square test for an assumed population variance

To investigate the significance of the difference between a population variance and an assumed variance value.

Limitations

1. Sample from normal distribution

**Parameters**

**ssize:** sample size  
**svar:** sample variance  
**pvar:** population variance (assumed)  
**confidence:** confidence level

**See Also:** Chay, ZE, Ling, MHT. 2010. COPADS, II: Chi-Square test, F-Test and t-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 2:3

**Status:** Tested function

**Since:** version 0.4

**F2Count**(*count1*, *count2*, *confidence*, *time1*=0, *time2*=0, *repeat*=False)

Test 25: F-test for two counts (Poisson distribution)

To investigate the significance of the difference between two counted results (based on a Poisson distribution).

Limitations

1. Counts must satisfy a Poisson distribution
2. Samples obtained under same conditions.

**Parameters**

**count1:** count of first sample  
**count2:** count of second sample  
**repeat:** flag for repeated sampling (default = False)  
**time1:** time at which first sample is taken (only needed if repeat = True)  
**time2:** time at which second sample is taken (only needed if repeat = True)  
**confidence:** confidence level

**See Also:** Chay, ZE, Ling, MHT. 2010. COPADS, II: Chi-Square test, F-Test and t-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 2:3

**Status:** Tested function**Since:** version 0.4

**ChisqFit**(*observed*, *expected*, *confidence*)

Test 37: Chi-square test for goodness of fit

To investigate the significance of the differences between observed data arranged in K classes, and the theoretical expected frequencies in the K classes.

Limitations

1. Observed and theoretical distributions should have same number of elements
2. Same class division for both distributions
3. Expected frequency of each class should be at least 5

**Parameters**

- observed:** list of observed frequencies (index matched with expected)
- expected:** list of expected frequencies (index matched with observed)
- confidence:** confidence level

**See Also:** Chay, ZE, Ling, MHT. 2010. COPADS, II: Chi-Square test, F-Test and t-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 2:3

**Status:** Tested function**Since:** version 0.4

**tx2testofKcounts**(*T*, *V*, *confidence*)

Test 38: The x2-test for compatibility of K counts

To investigate the significance of the differences between K counts.

Limitations:

1. The counts must be obtained under comparable conditions

**Parameters**

**T:** list of time under K counts  
**V:** list of values of K counts  
**confidence:** confidence level

**See Also:** Chay, ZE, Ling, MHT. 2010. COPADS, II: Chi-Square test, F-Test and t-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 2:3

**Status:** Tested function

**Since:** version 0.4

**Chisq2x2**(*s1*, *s2*, *ssize*, *confidence*)

Test 40: Chi-square test for consistency in 2x2 table

To investigate the significance of the differences between observed frequencies for two dichotomous distributions.

Limitations

1. Total sample size (sample 1 + sample 2) must be more than 20
2. Each cell frequency more than 3

**Parameters**

**s1:** 2-element list or tuple of frequencies for sample #1  
**s2:** 2-element list or tuple of frequencies for sample #2  
**ssize:** sample size  
**confidence:** confidence level

**See Also:** Chay, ZE, Ling, MHT. 2010. COPADS, II: Chi-Square test, F-Test and t-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 2:3

**Status:** Tested function

**Since:** version 0.4

**ChisquareKx2table**(*c1*, *c2*, *k*, *confidence*)

Test 41: The x2-test for consistency in a K x 2 table

To investigate the significance of the differences between K observed frequency distributions with a dichotomous classification.

Limitations:

1. K sample sizes must be large enough.
2. It is usually assumed to be satisfied if the cell frequencies are equal to 5

**Parameters**

**c1:** class #1 values of sample k  
**c2:** class #2 values of sample k  
**k:** number of samples  
**confidence:** confidence level

**See Also:** Chay, ZE, Ling, MHT. 2010. COPADS, II: Chi-Square test, F-Test and t-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 2:3

**Status:** Tested function

**Since:** version 0.4

**Chisquare2xKtable**(*s1*, *s2*, *k*, *confidence*)

Test 43: The x2-test for consistency in a 2 x K table

To investigate the significance of the differences between two distributions based on two samples spread over K classes

Limitations:

1. The two samples are sufficiently large
2. The K classes when put together form a complete series

**Parameters**

**s1:** sample #1 of class k  
**s2:** sample #2 of class k  
**k:** number of classes  
**confidence:** confidence level

**See Also:** Chay, ZE, Ling, MHT. 2010. COPADS, II: Chi-Square test, F-Test and t-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 2:3

**Status:** Tested function

**Since:** version 0.4

**MedianTestfor2Pop**(*s1*=(9, 6), *s2*=(6, 9), *confidence*=0.95)

Test 50: The median test of two populations

To test if two random samples could have come from two populations with same frequency distribution

Limitations:

1. The two samples are assumed to be reasonably large

**Parameters**

**s1:** 2-element list or tuple of frequencies for sample #1  
**s2:** 2-element list or tuple of frequencies for sample #2  
**confidence:** confidence level

**See Also:** Chay, ZE, Ling, MHT. 2010. COPADS, II: Chi-Square test, F-Test and t-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 2:3

**Status:** Tested function

**Since:** version 0.4



**SpearmanCorrelation**(*ssize*, *confidence*, *R=None*, *series1=[]*, *series2=[]*)

Test 58: Spearman rank correlation test (paired observations) To investigate the significance of the correlation between two series of observations obtained in pairs.

Limitations

1. Assumes the two population distributions to be continuous
2. Sample size must be more than 10

**Parameters**

**R:** sum of squared ranks differences  
**ssize:** sample size  
**series1:** ranks of series #1 (not used if R is given)  
**series2:** ranks of series #2 (not used if R is given)  
**confidence:** confidence level

**See Also:** Ling, MHT. 2009. Ten Z-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 1:5

**Status:** Tested function

**Since:** version 0.1

**ZtestLogOddsRatio**(*group1*, *group2*, *confidence*)

Test 84: Z-test for comparing sequential contingencies across two groups using the 'log odds ratio'

To test the significance of the difference in sequential connections across groups

Limitations:

1. This test is applicable when a logit transformation can be used and 2 X 2 contingency tables are available

**Parameters**

**group1:** group #1 values (row = x, y) & (column a, b) Provide values in xa, xb, ya, yb format  
**group2:** group #2 values (row = x, y) & (column a, b) Provide values in xa, xb, ya, yb format  
**confidence:** confidence level

## 15.2 Variables

Name	Description
BACKHOUSE	Value: 1.45607494858
CFRACT	Value: 1.0306408341
CGOLD	Value: 0.38196601125
FRODA	Value: 6.58088599102
GAMMA	Value: 0.577215664902
GOLD	Value: 1.61803398875
GOLOMB	Value: 0.624329988544
INV2PI	Value: 0.159154943092
INVLN10	Value: 0.434294481903
INVLN2	Value: 1.44269504089
INVPI180	Value: 57.2957795131
INVSQRT2PI	Value: 0.398942280401
LEHMER	Value: 0.592632718292
LENGYEL	Value: 1.09868580553
LN10	Value: 2.30258509299
LN2	Value: 0.69314718056
LNPI	Value: 1.14472988585
PARIS	Value: 1.09864196439
PI	Value: 3.14159265359
PI180	Value: 0.0174532925199
PI2	Value: 6.28318530718
PIDIV2	Value: 1.57079632679
PORTER	Value: 1.46707807943
RABBIT	Value: 0.709803442861
SQRT2	Value: 1.41421356237
SQRT2PI	Value: 2.50662827463
ZETA9	Value: 1.00200839283
__package__	Value: 'copads'

## 16 Module *copads.lc\_bf*

Loose Circular Brainfuck (LCBF) Interpreter Date created: 15th August 2012 Licence: Python Software Foundation License version 2

LCBF uses all 8 operations in standard Brainfuck with the following differences:

1. The tape or array is circular (a ring list) instead of linear. When the pointer is at the "end" of the tape, an increment ("**>**") will move the tape to the start. Similarly, when the pointer is decremented at the "beginning" of the tape, the pointer goes to the end.
2. Operations after a start loop operator ("**[**") will only be executed provided the loop(s) are properly closed. If the loops are open, the program will terminate.
3. However, it is possible to have an end loop operator ("**]**") without a preceding start loop operator ("**[**"). In this case, the end loop operator ("**]**") will be ignored and execution continues.
4. Unclosed or unopened loops may result in non-deterministic behaviour.
5. All inputs are pre-defined at the start of the program.

**See Also:** [http://esolangs.org/wiki/Loose\\_Circular\\_Brainfuck\\_\(LCBF\)](http://esolangs.org/wiki/Loose_Circular_Brainfuck_(LCBF))

### 16.1 Functions

<b>increment</b> ( <i>array, apointer, inputdata, output, source, spointer</i> )
--

Increase value of cell by 1. Equivalent to " <b>+</b> " in Brainfuck.
---

<b>decrement</b> ( <i>array, apointer, inputdata, output, source, spointer</i> )
--

Decrease value of cell by 1. Equivalent to " <b>-</b> " in Brainfuck.
---

<b>forward</b> ( <i>array, apointer, inputdata, output, source, spointer</i> )
--

Move forward by one cell on tape. Equivalent to " <b>&gt;</b> " in Brainfuck.
---

<b>backward</b> ( <i>array, apointer, inputdata, output, source, spointer</i> )
---

Move backward by one cell on tape. Equivalent to " <b>&lt;</b> " in Brainfuck.
--

<b>call_out</b> ( <i>array, apointer, inputdata, output, source, spointer</i> )
---

Output current tape cell value and append to the end of the output list. Equivalent to " <b>.</b> " in Brainfuck.
--

**accept\_predefined**(*array, apointer, inputdata, output, source, spointer*)

Writes the first value of the input list into the current cell and remove the value from the input list. If input list is empty, "0" will be written

**cbf\_start\_loop**(*array, apointer, inputdata, output, source, spointer*)

Start loop. Operations after a start loop operator ("["") will only be executed provided the loop(s) are properly closed. If the loops are open, the program will terminate. Note that unclosed or unopened loops may result in non-deterministic behaviour.

**cbf\_end\_loop**(*array, apointer, inputdata, output, source, spointer*)

End loop. However, it is possible to have an end loop operator ("]") without a preceding start loop operator ("["). In this case, the end loop operator ("]") will be ignored and execution continues. Note that unclosed or unopened loops may result in non-deterministic behaviour.

## 16.2 Variables

Name	Description
LCBF	<b>Value:</b> {'+' : increment, '-' : decrement, '>' : forward, '<' : backw...}
__package__	<b>Value:</b> 'copads'

## 17 Module *copads.matrix*

Matrix Data Structures and Algorithms.

Copyright (c) Maurice H.T. Ling <mauriceling@acm.org>

Date created: 19th March 2008

### 17.1 Functions

<b>isVector</b> ( $x$ )
Determines if the argument is a vector class object.

<b>vZeros</b> ( $n$ )
Returns a vector of length $n$ with all ones.

<b>vOnes</b> ( $n$ )
Returns a vector of length $n$ with all ones.

<b>vRandom</b> ( $n$ , $lmin=0.0$ , $lmax=1.0$ )
Returns a random vector of length $n$ .

<b>vDot</b> ( $a$ , $b$ )
dot product of two vectors.

<b>vNorm</b> ( $a$ )
Computes the norm of vector $a$ .

<b>vSum</b> ( $a$ )
Returns the sum of the elements of $a$ .

<b>vLog10</b> ( $a$ )
log10 of each element of $a$ .

<b>vLog</b> ( $a$ )
log of each element of $a$ .

**vExp**( $a$ )

Elementwise exponential.

**vSin**( $a$ )

Elementwise sine.

**vTan**( $a$ )

Elementwise tangent.

**vCos**( $a$ )

Elementwise cosine.

**vAsin**( $a$ )

Elementwise inverse sine.

**vAtan**( $a$ )

Elementwise inverse tangent.

**vAcos**( $a$ )

Elementwise inverse cosine.

**vSqrt**( $a$ )

Elementwise sqrt.

**vSinh**( $a$ )

Elementwise hyperbolic sine.

**vTanh**( $a$ )

Elementwise hyperbolic tangent.

**vCosh**( $a$ )

Elementwise hyperbolic cosine.

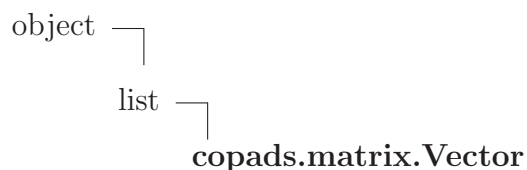
**vPow**( $a, b$ )Takes the elements of  $a$  and raises them to the  $b$ -th power

<b>vAtan2</b> ( $a, b$ )
Arc tangent
<b>isSparse</b> ( $x$ )
<b>smTranspose</b> ( $a$ )
transpose
<b>smDotDot</b> ( $y, a, x$ )
double dot product $y^+ * A * x$
<b>smDot</b> ( $a, b$ )
vector-matrix, matrix-vector or matrix-matrix product
<b>smDiag</b> ( $b$ )
<b>smIdentity</b> ( $n$ )

## 17.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'copads'

## 17.3 Class Vector



A list based vector class that supports elementwise mathematical operations

In this version, the vector call inherits from list; this requires Python 2.2 or later.

Adapted from <http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/52272> Original author: A. Pletzer

## 17.3.1 Methods

**\_\_getitem\_\_**(*self*, *i*, *j*)

*x*[*i*:*j*]

Use of negative indices is not supported.

Overrides: list.\_\_getitem\_\_ `exitit`(inherited documentation)

**\_\_add\_\_**(*self*, *other*)

Addition of current matrix to "other" matrix.

Overrides: list.\_\_add\_\_

**Status:** Tested method

**Since:** version 0.1

**\_\_neg\_\_**(*self*)

**Status:** Tested method

**Since:** version 0.1

**\_\_sub\_\_**(*self*, *other*)

**Status:** Tested method

**Since:** version 0.1

**\_\_mul\_\_**(*self*, *other*)

Element by element multiplication

Overrides: list.\_\_mul\_\_

**Status:** Tested method

**Since:** version 0.1

**\_\_rmul\_\_**(*self*, *other*)

*n*\**x*

Overrides: list.\_\_rmul\_\_ `exitit`(inherited documentation)



<b><code>--div--(self, other)</code></b>
Element by element division.
<b>Status:</b> Tested method
<b>Since:</b> version 0.1

<b><code>--rdiv--(self, other)</code></b>
The same as <code>--div--</code>

<b><code>size(self)</code></b>
--------------------------------

<b><code>conjugate(self)</code></b>
-------------------------------------

<b><code>ReIm(self)</code></b>
Return the real and imaginary parts

<b><code>AbsArg(self)</code></b>
Return modulus and phase parts

<b><code>out(self)</code></b>
Prints out the vector.

### *Inherited from list*

`--contains--()`, `--delitem--()`, `--delslice--()`, `--eq--()`, `--ge--()`, `--getattribute--()`, `--getitem--()`, `--gt--()`, `--iadd--()`, `--imul--()`, `--init--()`, `--iter--()`, `--le--()`, `--len--()`, `--lt--()`, `--ne--()`, `--new--()`, `--repr--()`, `--reversed--()`, `--setitem--()`, `--setslice--()`, `--sizeof--()`, `append()`, `count()`, `extend()`, `index()`, `insert()`, `pop()`, `remove()`, `reverse()`, `sort()`

### *Inherited from object*

`--delattr--()`, `--format--()`, `--reduce--()`, `--reduce_ex--()`, `--setattr--()`, `--str--()`, `--subclasshook--()`

## 17.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

## 17.3.3 Class Variables

Name	Description
<i>Inherited from list</i>	
<code>__hash__</code>	

## 17.4 Class Matrix

A linear algebra matrix

This class defines a generic matrix and the basic matrix operations from linear algebra. An instance of this class is a single matrix with particular values.

Arithmetic operations, trace, determinant, and minors are defined. This is a lightweight alternative to a numerical Python package for people who need to do basic linear algebra.

Vectors are implemented as 1xN and Nx1 matrices. There is no separate vector class. This implementation enforces the distinction between row and column vectors.

Indexing is zero-based, i.e. the upper left-hand corner of a matrix is element (0,0), not element (1,1).

Matrices are stored as a list of lists, where the top level lists are the rows and the sub-lists are the columns. Because of the way Python handles list references, you have be careful when copying matrix objects. If you have a matrix a, assign b=a, and then change values in b, you will change values in a as well. Matrix copying should be done with `copy.deepcopy`.

This implementation has no memory-saving optimization for sparse matrices. A derived class may implement a more sophisticated storage method by overriding the `__getitem__` and `__setitem__` functions.

Determinants are taken by expanding by minors on the top row. The private functions supplied for expansion by minors are more generic than what is needed by this implementation. They may be used by a derived class that wishes to do more efficient expansion of sparse matrices.

By default, Matrix elements are members of the complex field, but if you want to perform linear algebra on something other than numbers you may redefine `Matrix.null_element`, `Matrix.identity_element`, and `Matrix.inverse_element` and override the `is_scalar_element` function.

References George Arfken, "Mathematical Methods for Physicists", 3rd ed. San Diego. Academic Press Inc. (1985)

Adapted from: <http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/189971> Original author: Bill McNeill <billmcn@speakeasy.net>

Maintainer: Maurice H.T. Ling <mauriceling@acm.org> Copyright (c) 2005 Maurice H.T.

Ling <mauriceling@acm.org> Date: 1st May 2005

### 17.4.1 Methods

**`__init__(self, *args)`**

Matrix constructor

A matrix can be created in three ways.

1. A single integer argument is supplied. The constructor creates a null square matrix of that size. For example  
`Matrix(2)`  
 creates the following matrix  

```
0 0 0 0
```
2. Two integer arguments are supplied. The constructor creates a null matrix of size first argument x second argument. For example  
`Matrix(2, 3)`  
 creates the following matrix  

```
0 0 0 0 0 0
```
3. A list of lists is supplied. It represents a set of initial matrix values. Each element is a row and each sub-list is a column. For example  
`Matrix([[1,2,3], [4,5,6], [7,8,9]])`  
 creates the following matrix  

```
1 2 3 4 5 6 7 8 9
```

**`createNullMatrix(self, row, col)`**

Create a matrix using the null value

This is a private function called by `__init__`.

**`__str__(self)`**

**`__cmp__(self, other)`**

**`__getitem__(self, (row, col))`**

The value at (row, col)

For example, to get the value of element 1,3 say

`m[(1,3)]`

**`--setitem--`**(*self*, (*row*, *col*), *value*)

Sets the value at (row, col)

For example, to set the value of element 1,3 to 5 say

$m[(1,3)] = 5$

**`rows`**(*self*)

The number of rows in the matrix

**`cols`**(*self*)

The number of columns in the matrix

**`row`**(*self*, *i*)

The ith row of the matrix

**`col`**(*self*, *j*)

The jth row of the matrix

**`--add--`**(*self*, *other*)

Add matrix self + other

**`--neg--`**(*self*)

Negate the current matrix

**`--sub--`**(*self*, *other*)

Subtract matrix self - other

**`--mul--`**(*self*, *other*)

Multiply matrix self\*other

other can be another matrix or a scalar.

**`--rmul--`**(*self*, *other*)

Multiply other\*self

This is only called if other.`--add--` is not defined, so assume that other is a scalar.

**scalarMultiply**(*self*, *scalar*)

Multiply the matrix by a scalar value.

This is a private function called by `__mul__` and `__rmul__`.

**matrixMultiply**(*self*, *other*)

Multiply the matrix by another matrix.

This is a private function called by `__mul__`.

**isRowVector**(*self*)

Is the matrix a row vector?

**isColumnVector**(*self*)

Is the matrix a column vector?

**isSquare**(*self*)

Is the matrix square?

**transpose**(*self*)

The transpose of the matrix

**trace**(*self*)

The trace of the matrix

**determinant**(*self*)

The determinant of the matrix

**expandByMinorsOnRow**(*self*, *row*)

Calculates the determinant by expansion of minors

This function returns the determinant of the matrix by doing an expansion of minors on the specified row.

**expandByMinorsOnColumn**(*self*, *col*)

Calculates the determinant by expansion of minors

This function returns the determinant of the matrix by doing an expansion of minors on the specified column.

**minor**(*self*, *i*, *j*)

A minor of the matrix

This function returns the minor given by striking out row *i* and column *j* of the matrix.

**vectorInnerProduct**(*self*, *a*, *b*)

Takes the inner product of vectors *a* and *b*

*a* and *b* are lists. This is a private function called by `matrix_multiply`.

**isScalarElement**(*self*, *x*)

Is *x* a scalar

By default a scalar is an element in the complex number field. A class that wants to perform linear algebra on things other than numbers may override this function.

#### 17.4.2 Class Variables

Name	Description
<code>null_element</code>	<b>Value:</b> 0
<code>identity_element</code>	<b>Value:</b> 1
<code>inverse_element</code>	<b>Value:</b> -1

### 17.5 Class *SparseMatrix*



A sparse matrix class based on a dictionary, supporting matrix (dot) product and a conjugate gradient solver.

In this version, the sparse class inherits from the dictionary; this requires Python 2.2 or later.

Adapted from <http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/52275> Original author: Alexander Pletzer

Dictionary storage format { (i,j): value, ... } where (i,j) are the matrix indices

### 17.5.1 Methods

**size**(*self*)

returns # of rows and columns

**\_\_add\_\_**(*self*, *other*)

**\_\_neg\_\_**(*self*)

**\_\_sub\_\_**(*self*, *other*)

**\_\_mul\_\_**(*self*, *other*)

element by element multiplication: other can be scalar or sparse

**\_\_rmul\_\_**(*self*, *other*)

**\_\_div\_\_**(*self*, *other*)

element by element division self/other: other is scalar

**\_\_rdiv\_\_**(*self*, *other*)

element by element division other/self: other is scalar

**abs**(*self*)

**out**(*self*)

**plot**(*self*, *width\_in*=400, *height\_in*=400)

**CGsolve**(*self*, *x0*, *b*, *tol*=1e-10, *nmax*=1000, *verbose*=1)

Solve  $\text{self} \cdot x = b$  and return  $x$  using the conjugate gradient method

**biCGsolve**(*self*, *x0*, *b*, *tol*=1e-10, *nmax*=1000)

Solve  $\text{self} \cdot x = b$  and return  $x$  using the bi-conjugate gradient method

<b>save</b> ( <i>self</i> , <i>filename</i> , <i>OneBased</i> =0)
---

Save matrix in file <filename> using format OneBased, nrow, ncol, nnonzeros [ii, jj, data]
---

***Inherited from dict***

\_\_cmp\_\_(), \_\_contains\_\_(), \_\_delitem\_\_(), \_\_eq\_\_(), \_\_ge\_\_(), \_\_getattr\_\_(), \_\_getitem\_\_(),  
 \_\_gt\_\_(), \_\_init\_\_(), \_\_iter\_\_(), \_\_le\_\_(), \_\_len\_\_(), \_\_lt\_\_(), \_\_ne\_\_(), \_\_new\_\_(), \_\_repr\_\_(),  
 \_\_setitem\_\_(), \_\_sizeof\_\_(), clear(), copy(), fromkeys(), get(), has\_key(), items(),  
 iteritems(), iterkeys(), itervalues(), keys(), pop(), popitem(), setdefault(), update(),  
 values(), viewitems(), viewkeys(), viewvalues()

***Inherited from object***

\_\_delattr\_\_(), \_\_format\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_setattr\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

**17.5.2 Properties**

Name	Description
<i>Inherited from object</i>	
__class__	

**17.5.3 Class Variables**

Name	Description
<i>Inherited from dict</i>	
__hash__	



## 18 Module copads.n\_bf

NucleotideBF (nBF) Interpreter Date created: 15th August 2012 Licence: Python Software Foundation License version 2

NucleotideBF is a derivative of Brainfuck based on IUPAC nucleotide code. It uses only 5 of the 8 operations in Brainfuck, and there is no loop operations.

The commands for nBF can be divided into 2 classes - deterministic operations and random operations. The deterministic operations are A (increment, equivalent to '+'), T (decrement, equivalent to '-'), C (backward, equivalent to '<'), G (forward, equivalent to '>') and '.' (call out).

Based on the same interpreter as Loose Circular Brainfuck (LCBF), the tape or array is circular (a ring list) instead of linear. When the pointer is at the "end" of the tape, an increment ("A") will move the tape to the start. Similarly, when the pointer is decremented at the "beginning" of the tape, the pointer goes to the end.

**See Also:** [http://esolangs.org/wiki/NucleotideBF\\_\(nBF\)](http://esolangs.org/wiki/NucleotideBF_(nBF))

### 18.1 Functions

**random\_op**(array, apointer, inputdata, output, source, spointer)

Random operations / commands to simulate ambiguous DNA nucleotides.  
 Allowable ambiguous nucleotides are: R: Random between A or G Y: Random between C or T S: Random between G or C W: Random between A or T K: Random between G or T M: Random between A or C B: Random between C or G or T D: Random between A or G or T H: Random between A or C or T V: Random between A or C or G N: Random between A or T or C or G

### 18.2 Variables

Name	Description
nBF	<b>Value:</b> {'A': increment, 'T': decrement, 'G': forward, 'C': backw...
--package--	<b>Value:</b> 'copads'

## 19 Module copads.neural

Framework for Neural Network Applications Date created: 10th May 2013 License: Python Software Foundation License version 2

### 19.1 Functions

<b>tf_linear</b> ( $x$ )
Linear transfer function. Equation: $y = x$

<b>itf_linear</b> ( $y$ )
Linear transfer function - inverse. Equation: $x = y$

<b>dtf_linear</b> ( $x$ )
Linear transfer function - derivative. Equation: $df = 1.0$

### 19.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'copads'

### 19.3 Class Neuron

Class for a Neuron.

1. Each neuron is identified by a unique name, which will be used for mapping between different neurons.
2. A dictionary, named "cellbody", is provided to contain any other information needed. For example, it can be used to contain a timer for time-delayed neuron activation or it can be used for neuronal memory.

## 19.3.1 Methods

```
__init__(self, name=None, transfer_function=<function tf_linear at
0x10f96a320>, itransfer_function=<function itf_linear at
0x10f96a398>, dtransfer_function=<function dtf_linear at
0x10f96a410>)
```

Constructor method for Neuron class.

**Parameters**

<b>name:</b>	unique name of the neuron. Default = a string of 60 to 75 numbers. ( <i>type=string</i> )
<b>transfer_function:</b>	a function to convert the consolidated weighted input for the neuron and generate an output signal. Default = tf_linear; that is, output signal = consolidated weighted input. ( <i>type=function</i> )
<b>itransfer_function:</b>	inverse of transfer_function, used by some learning algorithms, Default = itf_linear ( <i>type=function</i> )
<b>dtransfer_function:</b>	differential of transfer_function, used by some learning algorithms, Default = dtf_linear ( <i>type=function</i> )

```
generate_name(self)
```

Generate a new random name (a string of 60 to 75 numbers) for the current neuron.

**Return Value**

new name of neuron.

```
set_transfer_functions(self, transfer_function, itransfer_function=None,
dtransfer_function=None)
```

Set transfer function, its inverse and differential functions.

#### Parameters

- transfer\_function:** a function to convert the consolidated weighted input for the neuron and generate an output signal.  
(*type=function*)
- itransfer\_function:** inverse of transfer\_function, used by some learning algorithms, Default = None  
(*type=function*)
- dtransfer\_function:** differential of transfer\_function, used by some learning algorithms, Default = None  
(*type=function*)

```
execute(self, synapses, activations)
```

Execute/activate the neuron into action.

#### Parameters

- synapses:** dictionary of reverse synaptic connections and its synaptic weights in the format of {name of signal receiving neuron>: {<name of signal receiving neuron>: <synaptic weight>}}
- (*type=dictionary*)
- activations:** a dictionary of current activations of the entire network where keys are the names of neurons and values are the activation state or current output signal.  
(*type=dictionary*)

#### Return Value

activations dictionary with the updated output signal or activation state from the current neuron.

### 19.3.2 Class Variables

Name	Description
name	<b>Value:</b> None
weights	<b>Value:</b> {}
cellbody	<b>Value:</b> {}

*continued on next page*

Name	Description
transfer_function	<b>Value:</b> None
itransfer_function	<b>Value:</b> None
dtransfer_function	<b>Value:</b> None

## 19.4 Class Brain

Class for the neural network.

1. A brain consists of a set of neurons (uniquely named or uniquely randomly named) held in "neuron\_pool" dictionary where key is the name and value is the neuron object.
2. The corresponding names of neurons are entered into "activations" dictionary as keys. This dictionary holds the current activation states or output signals (as values) of all neurons.
3. A sequence of neuronal activity is given as activation\_sequence, which is a list of list. This suggests that neurons listed in activation\_sequence[0] will be activated before those neurons listed in activation\_sequence[1], and neuron in activation\_sequence[0][0] will be activated before neuron in activation\_sequence[0][1]. Inherited class can over-ride Brain.run() method to cater for parallel activations of all neurons in activation\_sequence[0] or even parallel activation of all neurons regardless of sequence.
4. In sequential activation (that is activation\_sequence[0] before activation\_sequence[1]), neurons in activation\_sequence[0] will be the input neurons to traditional neural networks.
5. The connections and synaptic weights are registered in "synapses" dictionary as a reverse connection - {<name of signal receiving neuron>: <dictionary of signal receiving neurons and its weights>} where <dictionary of signal receiving neurons and its weights> is {<name of signal receiving neuron>: <synaptic weight>}. Hence, synapses dictionary is a nested dictionary of {name of signal receiving neuron>: {<name of signal receiving neuron>: <synaptic weight>}}.
6. A dictionary, called "brainmatter", is provided to contain any other information needed.
7. The brain will also a learning algorithm to train / learn by itself.

## 19.4.1 Methods

---

```
__init__(self, number_of_neurons=0, original_neuron=None,
list_of_neuron_names=[], learning_algorithm=None)
```

---

Constructor method for Brain class.

A brain or neural network can be constructed by a predefined list of unique names for neurons or by stating the number of neurons required. In both cases, a pre-created neuron may be used but is not mandatory. However, in event where both the list of unique names for neurons and the number of neurons required are given, the list of unique names for neurons takes precedence - for example, if 5 neuron names but 10 neurons are required, then only 5 neurons (which are named) will be created.

**Parameters**

<b>number_of_neurons:</b>	number of neurons requested (takes a lower precedence than the names of neurons), default = 0. ( <i>type=integer</i> )
<b>original_neuron:</b>	a pre-created neuron to duplicate, default = None ( <i>type=Neuron object</i> )
<b>list_of_neuron_names:</b>	names for neurons to be created (takes a higher precedence than the number of neurons requested), default = [] (empty list). ( <i>type=list</i> )
<b>learning_algorithm:</b>	learning mechanism for brain ( <i>type=function</i> )

---

```
connect_neurons(self, originating_neuron, destination_neuron)
```

---

Establish synaptic connection between 2 existing neurons.

**Parameters**

<b>originating_neuron:</b>	name of neuron to connect from ( <i>type=string</i> )
<b>destination_neuron:</b>	name of neuron to connect to ( <i>type=string</i> )

---

**disconnect\_neurons**(*self*, *originating\_neuron*, *destination\_neuron*)

---

Disconnect between 2 connected neurons.

**Parameters**

**originating\_neuron:** name of neuron to disconnect from  
*(type=string)*

**destination\_neuron:** name of neuron to disconnect to  
*(type=string)*

---

**set\_activation\_sequence**(*self*, *activation\_sequence*)

---

Set neuron activation sequence or replace current neuron activation sequence.

**Parameters**

**activation\_sequence:** list of list of activation sequence

---

**add\_neuron\_to\_activation\_sequence**(*self*, *neuron\_name*, *position*)

---

Add neuron to activation sequence.

**Parameters**

**neuron\_name:** name of neuron to add  
*(type=string)*

**position:** position to add neuron, where position > 0  
*(type=integer)*

---

**remove\_neuron\_from\_activation\_sequence**(*self*, *neuron\_name*,  
*position='all'*)

---

Remove neuron from activation sequence.

**Parameters**

**neuron\_name:** name of neuron to remove  
*(type=string)*

**position:** position to add neuron, where position > 0 (integer)  
or 'all' (string). If position = 'all', this method will  
remove neuron from the entire activation sequence

**add\_neuron**(*self*, *neuron*=None)

Add a neuron into the brain.

**Parameters**

**neuron:** the neuron to be added. If None, then a generic neuron will be created and added. Default = None.  
(*type*=*Neuron object*)

**remove\_neuron**(*self*, *neuron\_name*)

Removing a neuron from the brain.

**Parameters**

**neuron\_name:** name of neuron to remove  
(*type*=*string*)

**empty\_brain**(*self*)

Empty/clear the entire brain of all neurons, connections and states.

#### 19.4.2 Class Variables

Name	Description
activations	<b>Value:</b> {}
neuron_pool	<b>Value:</b> {}
synapses	<b>Value:</b> {}
activation_sequence	<b>Value:</b> []
brainmatter	<b>Value:</b> {}
learning_algorithm	<b>Value:</b> None



## 20 Module *copads.nrpy*

Numerical Recipes in Python.

References:

- Press, William H., Flannery, Brian P., Teukolsky, Saul A., and Vetterling, William T. 1989. Numerical Recipes in Pascal. Cambridge University Press, Cambridge (ISBN 978-0521375160)
- Press, William H., Flannery, Brian P., Teukolsky, Saul A., and Vetterling, William T. 1992. Numerical Recipes in C, 2nd edition. Cambridge University Press, Cambridge (ISBN 978-0521431088)

Numerical Recipes in C, 2nd edition is freely browsable online at <http://www.nrbook.com/a/bookcpdf.php> but is not intended as a substitution for purchasing the book.

Functions will be named as in the references and will be referred to section number. For example, the reference "NRP 5.2" refers to Numerical Recipes in Pascal chapter 5 section 2.

The authors of Numerical Recipes in Pascal (NRP) and Numerical Recipes in C, 2nd edition (NRC2) explicitly allows the reader to analyze the mathematical ideas in the codes within the book and owns the re-implemented functions as stated in NRP and NRC2 that "If you analyze the ideas contained in a program, and then express those ideas in your own distinct implementation, then that new program implementation belongs to you" (page xv of NRP; page xviii of NRC2). Not mentioned in NRP, NRC2 allows the reader a "free licence" (page xviii of NRC2) which allows the reader to make one machine-readable copy of the C codes in the book for his/her own use (not distribution) in his/her work, provided that the source codes are not distributed. As such, the codes in this file will be called by other functions in COPADS but not for direct use by users of COPADS - if you intend to call these functions directly, the simplest way is to own a copy of both NRP and NRC2.

Copyright (c) Maurice H.T. Ling <[mauriceling@acm.org](mailto:mauriceling@acm.org)>

Date created: 19th March 2008

## 20.1 Functions

**bessi**( $n, x$ )Modified Bessel function I-sub- $n$ ( $x$ ).**Parameters** $n$ : integer, more than 1 - modified  $n$ -th Bessel function $x$ : positive integer**Return Value**modified  $n$ -th Bessel function of  $x$ **See Also:** NRP 6.5**Status:** Tested function**Since:** version 0.1**bessi0**( $x$ )Modified Bessel function I-sub-0( $x$ ).**Parameters** $x$ : float number**Return Value**modified Bessel function base 0 of  $x$ **See Also:** NRP 6.5**Status:** Tested function**Since:** version 0.1**bessi1**( $x$ )Bessel function I-sub-1( $x$ ).**Parameters** $x$ : float number**Return Value**

float number

**See Also:** NRP 6.5**Status:** Tested function**Since:** version 0.1

**bessj**( $n, x$ )

Bessel function J-sub-n(x).

**Parameters**

**x**: float number

**Return Value**

float number

**See Also:** NRP 6.5

**Status:** Tested function

**Since:** version 0.1

**bessj0**( $x$ )

Bessel function J-sub-0(x).

**Parameters**

**x**: float number

**Return Value**

float number

**See Also:** NRP 6.4

**Status:** Tested function

**Since:** version 0.1

**bessj1**( $x$ )

Bessel function J-sub-1(x).

**Parameters**

**x**: float number

**Return Value**

float number

**See Also:** NRP 6.4

**Status:** Tested function

**Since:** version 0.1

**bessk**( $n, x$ )Bessel function K-sub- $n$ ( $x$ ). @see: NRP 6.5**Parameters** $n$ : integer, more than 1 - modified  $n$ -th Bessel function $x$ : positive integer**Return Value**modified  $n$ -th Bessel function of  $x$ **bessk0**( $x$ )Bessel function K-sub-0( $x$ ).**Parameters** $x$ : positive integer**Return Value** $n$ -th Bessel function of  $x$ **See Also:** NRP 6.5**bessk1**( $x$ )Bessel function K-sub-1( $x$ ).**Parameters** $x$ : positive integer**Return Value** $n$ -th Bessel function of  $x$ **See Also:** NRP 6.5**bessy**( $n, x$ )Bessel function Y-sub- $n$ ( $x$ ).**Parameters** $n$ : integer, more than 1 -  $n$ -th Bessel function $x$ : positive integer**Return Value** $n$ -th Bessel function of  $x$ **See Also:** NRP 6.4**Status:** Tested function**Since:** version 0.1

**bessy0**( $x$ )

Bessel function  $Y_{\text{-sub-0}}(x)$ .

**Parameters**

**x**: float number

**Return Value**

float number

**See Also:** NRP 6.4 Depend: `bessj0`

**Status:** Tested function

**Since:** version 0.1

**bessy1**( $x$ )

Bessel function  $Y_{\text{-sub-1}}(x)$ .

**Parameters**

**x**: float number

**Return Value**

float number

**See Also:** NRP 6.4 Depend: `bessj1`

**Status:** Tested function

**Since:** version 0.1

**beta**( $z, w$ )

Beta function. Depend: `gammln`

**Parameters**

**z**: float number

**w**: float number

**Return Value**

float number

**See Also:** NRP 6.1, Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's  $t$ , and Uniform. The Python Papers Source Codes 1:4

**Status:** Tested function

**Since:** version 0.1

**betacf**( $a, b, x$ )

Continued fraction for incomplete beta function. Adapted from `salstat_stats.py` of SalStat ([www.sf.net/projects/salstat](http://www.sf.net/projects/salstat))

**See Also:** NRP 6.3, Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

**betai**( $a, b, x$ )

Incomplete beta function

$I_{-x}(a,b) = 1/B(a,b) * (\text{Integral}(0,x) \text{ of } t^{(a-1)}(1-t)^{(b-1)} dt)$

where  $a,b > 0$  and  $B(a,b) = G(a)*G(b)/(G(a+b))$  where  $G(a)$  is the gamma function of  $a$ .

Adapted from `salstat_stats.py` of SalStat ([www.sf.net/projects/salstat](http://www.sf.net/projects/salstat))

Depend: `betacf`, `gammln`

**See Also:** NRP 6.3, Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

**Status:** Tested function

**Since:** version 0.1

**bico**( $n, k$ )

Binomial coefficient. Returns  $n!/(k!(n-k)!)$  Depend: `factln`, `gammln`

#### Parameters

**n:** total number of items

**k:** required number of items

#### Return Value

floating point number

**See Also:** NRP 6.1, Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

**Status:** Tested function

**Since:** version 0.1

**chebev**(*a*, *b*, *c*, *m*, *x*)

Chebyshev evaluation.

**Parameters****a**: float number**b**: float number**c**: list of Chebyshev coefficients produced by `chebft` with the same 'a' and 'b'**m**:**x**:**Return Value**

float number - function value

**See Also:** NRP 5.6**erf**(*x*)

Error function (a special incomplete gamma function) equivalent to `gammp(0.5, x^2)` for  $x \geq 0$ . In this routine, `gammp` is by-passed and `gser` and `gcf` are used directly. Depend: `gser`, `gcf`, `gammln`

**Parameters****x**: float number**Return Value**

float number

**See Also:** NRP 6.2**erfc**(*x*)

Complementary error function (a special incomplete gamma function) equivalent to `gammq(0.5, x^2)` which is equivalent to  $1 - \text{gammp}(0.5, x^2)$  for  $x \geq 0.0$ . Depend: `gammp`, `gammq`, `gser`, `gcf`, `gammln`

**Parameters****x**: float number**Return Value**

float number

**See Also:** NRP 6.2**Status:** Tested function**Since:** version 0.1

**erfcc**( $x$ )

Complementary error function similar to `erfc(x)` but with fractional error lesser than  $1.2e-7$ .

**Parameters**

**x**: float number

**Return Value**

float number

**See Also:** NRP 6.2

**Status:** Tested function

**Since:** version 0.1

**expdev**( $x$ )

Depends: `ran3`

**See Also:** NRP 7.2

**factln**( $n$ )

Natural logarithm of factorial:  $\ln(n!)$

**Parameters**

**n**: positive integer

**Return Value**

natural logarithm of factorial of  $n$

**See Also:** NRP 6.1, Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's  $t$ , and Uniform. The Python Papers Source Codes 1:4



**gammln**( $n$ )

Complete Gamma function.

**Parameters**

**n**: float number

**Return Value**

float number

**See Also:** NRP 6.1,

<http://mail.python.org/pipermail/python-list/2000-June/671838.html>, Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

**Status:** Tested function

**Since:** version 0.1

**gammp**( $a, x$ )

Gamma incomplete function,  $P(a,x)$ .  $P(a,x) = (1/\text{gammln}(a)) * \text{integral}(0, x, (e^{-t}) * (t^{(a-1)})), dt$  Depend: gser, gcf, gammln

**Parameters**

**a**: float number

**x**: float number

**Return Value**

float number

**See Also:** NRP 6.2, Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

**Status:** Tested function

**Since:** version 0.1

**gammq**(*a*, *x*)

Incomplete gamma function:  $Q(a, x) = 1 - P(a, x) = 1 - \text{gammq}(a, x)$  Also commonly known as Q-equation.

**See Also:**

<http://mail.python.org/pipermail/python-list/2000-June/671838.html>, Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

**Status:** Tested function

**Since:** version 0.1

**gcf**(*a*, *x*, *itmax*=200, *eps*=3e-07)

Continued fraction approx'n of the incomplete gamma function.

**See Also:**

<http://mail.python.org/pipermail/python-list/2000-June/671838.html>, Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

**Status:** Tested function

**Since:** version 0.1

**gser**(*a*, *x*, *itmax*=700, *eps*=3e-07)

Series approximation to the incomplete gamma function.

**See Also:**

<http://mail.python.org/pipermail/python-list/2000-June/671838.html>, Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

**Status:** Tested function

**Since:** version 0.1

**mdian1**(*data*)

Calculates the median of a list of numerical values using sorting.

**Parameters**

**data:** a 1-dimensional list of numerical data

**Return Value**

value of median

**See Also:** NRP 13.2

**moment**(*data*)

Calculates moment from a list of numerical data. @see: NRP 13.1

**Parameters**

**data:** a 1-dimensional list of numerical values

**Return Value**

(ave, adev, sdev, var, skew, kurt) where

- ave = mean
- adev = average deviation
- sdev = standard deviation
- var = variance
- skew = skew
- kurt = kurtosis

**qgaus**(*a, b, func*)

**See Also:** NRP 4.5

**cdf\_binomial**(*k, n, p*)

Cummulative density function of Binomial distribution. No reference implementation. Depend: betai, betacf, gammln

**Parameters**

**k:** number of times of event occurrence in n trials

**n:** total number of trials

**p:** probability of event occurrence per trial

**Return Value**

float number - Binomial probability

**See Also:** NRP 6.3, Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

**cdf\_poisson**( $k, x$ )

Cumulative density function of Poisson distribution from 0 to  $k - 1$  inclusive.  
No reference implementation. Depend: gammq, gser, gcf, gammln

**Parameters**

**k**: number of times of event occurrence

**x**: mean of Poisson distribution

**Return Value**

float number - Poisson probability of  $k - 1$  times of occurrence with  
the mean of  $x$

**See Also:** NRP 6.2, Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's  $t$ , and Uniform. The Python Papers Source Codes 1:4

**20.2 Variables**

Name	Description
--package--	<b>Value:</b> 'copads'

## 21 Module *copads.objectdistance*

Functions for Calculating Similarity Coefficients between 2 Objects.

Generally, the lower boundary of similar coefficient signifies complete difference (no similarity) while the upper boundary (if any) signifies complete similarity (no difference).

In the following formulae, the following notations will be used

- A = found in both 'original' and 'test'
- B = found in 'original' only
- C = found in 'test' only
- D = not found in either 'original' or 'test'

Copyright (c) Maurice H.T. Ling <mauriceling@acm.org>

Date created: 17th August 2005

### 21.1 Functions

**binarize**(*data*, *absent*=0)

Converts input data in a list of presence or absence of values. For example,  
 binarize([1, 2, 0, 3, 4, 0], 0) -> [1, 1, 0, 1, 1, 0] binarize([1, 2, 0, 3, 4, 0], 2) ->  
 [1, 0, 1, 1, 1, 1]

**compare**(*original*, *test*, *absent*, *type*='Set')

Used for processing list-based (positional) or set-based (non-positional) distance of categorical data.

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** indicator to define absent data  
**type:** {List | Set}

**Jaccard**(*original*, *test*, *absent*=0, *type*='Set')

Jaccard coefficient for nominal or ordinal data.

Coefficient:  $A / (A + B + C)$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.1

**Sokal\_Michener**(*original*, *test*, *absent*=0, *type*='Set')

Sokal and Michener coefficient for nominal or ordinal data.

Coefficient:  $(A + D) / (A + B + C + D)$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.1

**Matching**(*original*, *test*, *absent*=0, *type*='Set')

Matching coefficient for nominal or ordinal data

Coefficient:  $(A + D) / (2A + B + C)$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Dice**(*original*, *test*, *absent*=0, *type*='Set')

Dice coefficient for nominal or ordinal data.

Coefficient:  $2A / (2A + B + C)$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.1

**Ochiai**(*original*, *test*, *absent*=0, *type*='Set')

Ochiai coefficient for nominal or ordinal data.

Coefficient:  $2A / \sqrt{(A + B)(A + C)}$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.1

**Ochiai2**(*original*, *test*, *absent*=0, *type*='Set')

Ochiai 2 coefficient for nominal or ordinal data, and requires the presence of regions whereby both original and test are not present.

Coefficient:  $(A * D) / \sqrt{(A + B)(A + C)(D + B)(D + C)}$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3



**Anderberg**(*original, test, absent=0, type='Set'*)

Anderberg coefficient for nominal or ordinal data.

Coefficient:  $A / (A + 2(B + C))$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Kulczynski2**(*original, test, absent=0, type='Set'*)

Second Kulczynski coefficient for nominal or ordinal data.

Coefficient:  $((A / (A + B)) + (A / (A + C))) / 2$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Kulczynski**(*original*, *test*, *absent*=0, *type*='Set')

First Kulczynski coefficient for nominal or ordinal data.

Coefficient:  $A / (B + C)$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.1

**Forbes**(*original*, *test*, *absent*=0, *type*='Set')

Forbes coefficient for nominal or ordinal data.

Coefficient:  $A(A + B + C + D) / ((A + B)(A + C))$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Hamann**(*original*, *test*, *absent*=0, *type*='Set')

Hamann coefficient for nominal or ordinal data.

Coefficient:  $((A + D) - (B + C)) / (A + B + C + D)$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Simpson**(*original*, *test*, *absent*=0, *type*='Set')

Simpson coefficient for nominal or ordinal data.

Coefficient:  $A / \min(A + B, A + C)$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Russel\_Rao**(*original*, *test*, *absent=0*, *type='Set'*)

Russel and Rao coefficient for nominal or ordinal data.

Coefficient:  $A / (A + B + C + D)$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Roger\_Tanimoto**(*original*, *test*, *absent=0*, *type='Set'*)

Roger and Tanimoto coefficient for nominal or ordinal data.

Coefficient:  $(A + D) / (A + 2B + 2C + D)$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Sokal\_Sneath**(*original, test, absent=0, type='Set'*)

Sokal and Sneath coefficient for nominal or ordinal data.

Coefficient:  $A / (A + 2B + 2C)$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Sokal\_Sneath2**(*original, test, absent=0, type='Set'*)

Sokal and Sneath 2 coefficient for nominal or ordinal data.

Coefficient:  $(2A + 2D) / (2A + B + C + 2D)$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Sokal\_Sneath3**(*original*, *test*, *absent*=0, *type*='Set')

Sokal and Sneath 3 coefficient for nominal or ordinal data.

Coefficient:  $(A + D) / (B + C)$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Buser**(*original*, *test*, *absent*=0, *type*='Set')

Buser coefficient (also known as Baroni-Urbani coefficient) for nominal or ordinal data.

Coefficient:  $(\sqrt{A * D} + A) / (\sqrt{A * D} + A + B + C)$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Fossum**(*original*, *test*, *absent*=0, *type*='Set')

Fossum coefficient for nominal or ordinal data.

Coefficient:  $((A + B + C + D)(A - 0.5)^2) / ((A + B)(A + C))$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**YuleQ**(*original*, *test*, *absent*=0, *type*='Set')

Yule Q coefficient (also known as First Yule coefficient) for nominal or ordinal data.

Coefficient:  $((A * D) - (B * C)) / ((A * D) + (B * C))$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**YuleY**(*original*, *test*, *absent*=0, *type*='Set')

Yule Y coefficient (also known as Second Yule coefficient) for nominal or ordinal data.

Coefficient:  $(\text{sqrt}(A * D) - \text{sqrt}(B * C)) / (\text{sqrt}(A * D) + \text{sqrt}(B * C))$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Mcconnaughey**(*original*, *test*, *absent*=0, *type*='Set')

McConnaughey coefficient for nominal or ordinal data.

Coefficient:  $(A^2 - (B * C)) / (A + B)(A + C)$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3



**Stiles**(*original*, *test*, *absent*=0, *type*='Set')

Stiles coefficient for nominal or ordinal data.

Coefficient:  $\log_{10}(((A + B + C + D)(|(A * D) - (B * C)| - ((A + B + C + D) / 2)) ^ 2) / (A + B)(A + C)(B + D)(C + D))$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Pearson**(*original*, *test*, *absent*=0, *type*='Set')

Pearson coefficient for nominal or ordinal data.

Coefficient:  $((A * D) - (B * C)) / (A + B)(A + C)(B + D)(C + D)$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Dennis**(*original*, *test*, *absent*=0, *type*='Set')

Dennis coefficient for nominal or ordinal data.

Coefficient:  $((A * D) - (B * C)) / (A + B + C + D)(A + B)(A + C)$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Gower\_Legendre**(*original*, *test*, *absent*=0, *type*='Set')

Gower and Legendre coefficient for nominal or ordinal data.

Coefficient:  $(A + D) / ((0.5 * (B + C)) + A + D)$

**Parameters**

**original:** list of original data  
**test:** list of data to test against original  
**absent:** user-defined identifier for absent of region, default = 0  
**type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Tulloss**(*original*, *test*, *absent*=0, *type*='Set')

Tulloss coefficient for nominal or ordinal data.

Coefficient:  $\sqrt{U * S * R}$

- $U = \log(1 + ((\min(B, C) + A) / (\max(B, C) + A))) / \log 2$
- $S = 1 / \sqrt{\log(2 + (\min(B, C) / (A + 1))) / \log 2}$
- $R = \log(1 + A / (A + B)) \log(1 + A / (A + C)) / \log 2 \log 2$

**Parameters**

- original:** list of original data
- test:** list of data to test against original
- absent:** user-defined identifier for absent of region, default = 0
- type:** {Set | List}, define whether use Set comparison (non-positional) or list comparison (positional), default = Set

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.3

**Hamming**(*original*, *test*)

Hamming coefficient for ordinal data - only for positional data.

Coefficient: number of mismatches with respect to position

**Parameters**

- original:** list of original data
- test:** list of data to test against original

**See Also:** Ling, MHT. 2010. COPADS, I: Distances Measures between Two Lists or Sets. The Python Papers Source Codes 2:2.

**Status:** Tested function

**Since:** version 0.1

**Euclidean**(*original*, *test*)

Euclidean coefficient for interval or ratio data.

Coefficient:  $\text{sqrt}(\sum(((A + B)(i) - (A + C)(i)) ^ 2))$

euclidean(original, test) -> euclidean distance between original and test.  
Adapted from BioPython

**Parameters**

**original:** list of original data

**test:** list of data to test against original

**Status:** Tested function

**Since:** version 0.1

**Minkowski**(*original*, *test*, *power=3*)

Minkowski coefficient for interval or ratio data.

Coefficient:  $\text{power-th root}(\sum(((A + B)(i) - (A + C)(i)) ^ \text{power}))$

Minkowski Distance is a generalized absolute form of Euclidean Distance.  
Minkowski Distance = Euclidean Distance when power = 2

**Parameters**

**original:** list of original data

**test:** list of data to test against original

**power:** exponential variable  
(*type=integer*)

**Status:** Tested function

**Since:** version 0.4

**Manhattan**(*original*, *test*)

Manhattan coefficient for interval or ratio data.

Coefficient:  $\sum (abs((A + B)(i) - (A + C)(i)))$

Manhattan Distance is also known as City Block Distance. It is essentially summation of the absolute difference between each element.

**Parameters**

**original:** list of original data

**test:** list of data to test against original

**See Also:** Krause, Eugene F. 1987. Taxicab Geometry. Dover. ISBN 0-486-25202-7.

**Status:** Tested function

**Since:** version 0.4

**Canberra**(*original*, *test*)

Canberra coefficient for interval or ratio data.

Coefficient:  $\sum (abs((A + B)(i) - (A + C)(i)) / abs((A + B)(i) + (A + C)(i)))$

**Parameters**

**original:** list of original data

**test:** list of data to test against original

**See Also:** Lance GN and Williams WT. 1966. Computer programs for hierarchical polythetic classification. Computer Journal 9: 60-64.

**Status:** Tested function

**Since:** version 0.4

**Bray\_Curtis**(*original, test*)

Complement Bray and Curtis coefficient for interval or ratio data. Lower boundary of Bray and Curtis coefficient represents complete similarity (no difference).

Coefficient:  $1 - \sum(\text{abs}((A + B)(i) - (A + C)(i))) / (\sum((A + B)(i)) + \sum((A + C)(i)))$

**Parameters**

**original:** list of original data

**test:** list of data to test against original

**See Also:** Bray JR and Curtis JT. 1957. An ordination of the upland forest communities of S. Wisconsin. Ecological Monographs 27: 325-349.

**Status:** Tested function

**Since:** version 0.4

**Cosine**(*original, test*)

Cosine coefficient for interval or ratio data.

Coefficient:  $\sum(\text{abs}((A + B)(i) * (A + C)(i))) / (\sum((A + B) ^ 2) * \sum((A + C) ^ 2))$

**Parameters**

**original:** list of original data

**test:** list of data to test against original

**Status:** Tested function

**Since:** version 0.4

**Tanimoto**(*original, test*)

Tanimoto coefficient for interval or ratio data.

Coefficient:  $\sum(\text{abs}((A + B)(i) * (A + C)(i))) / (\sum((A + B) ^ 2) + \sum((A + C) ^ 2) - \sum(\text{abs}((A + B)(i) * (A + C)(i))))$

**Parameters**

**original:** list of original data

**test:** list of data to test against original

**Status:** Tested function

**Since:** version 0.4

## 21.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'copads'</code>

## 22 Module *copads.operations*

Mathematical Operation Routines.

Copyright (c) Maurice H.T. Ling <mauriceling@acm.org>

Date created: 19th March 2008

### 22.1 Functions

<b><code>asdict(<i>l</i>)</code></b>
Return a dictionary where the keys are the items in the list, with arbitrary values. This is useful for quick testing of membership. Adapted from BioPython.
<b>Return Value</b> dictionary

<b><code>items(<i>l</i>)</code></b>
Generate a list of one of each item in <i>l</i> . The items are returned in arbitrary order. Adapted from BioPython.
<b>Return Value</b> list of items

<b><code>count(<i>items</i>)</code></b>
Count the number of times each item appears in a list of data. Adapted from BioPython.
<b>Return Value</b> dict of counts of each item

<b><code>contents(<i>items</i>)</code></b>
Summarize the contents of the list in terms of the percentages of each item. For example, if an item appears 3 times in a list with 10 items, it is in 0.3 of the list. Adapted from BioPython.
<b>Return Value</b> dict of item:percentage



**itemindex**(*l*)

Make an index of the items in the list. The dictionary contains the items in the list as the keys, and the index of the first occurrence of the item as the value. Adapted from BioPython.

**Return Value**

dict of item : index of item

**indexesof**(*l*, *fn*)

Return a list of indexes *i* where *fn*(*l*[*i*]) is true. Adapted from BioPython.

**Return Value**

list of indexes

**take**(*l*, *indexes*)

Adapted from BioPython.

**Return Value**

list of just the indexes from *l*

**take\_byfn**(*l*, *fn*, *opposite*=0)

Adapted from BioPython.

**fcmp**(*x*, *y*, *precision*)**Return Value**

-1, 0, or 1

**intd**(*x*, *digits\_after\_decimal*=...)

Represent a floating point number with some digits after the decimal point as an integer. This is useful when floating point comparisons are failing due to precision problems. e.g. `intd(5.35, 1) -> 54`. Adapted from BioPython.

**Return Value**

int *x*, rounded

**safe\_log**(*n*, *zero*=None, *neg*=None)

Calculate the log of *n*. If *n* is 0, returns the value of *zero*. If *n* is negative, returns the value of *neg*. Adapted from BioPython.

**Return Value**

log(*n*)

**safe\_log2**(*n*, *zero*=None, *neg*=None)

Calculate the log base 2 of *n*. If *n* is 0, returns the value of *zero*. If *n* is negative, returns the value of *neg*. Adapted from BioPython.

**Return Value** $\log(n)$ **safe\_exp**(*n*, *under*=None, *over*=None)

Guaranteed not to overflow. Instead of overflowing, it returns the values of 'under' for underflows or 'over' for overflows. Adapted from BioPython.

**Return Value** $e^{**n}$ **factorial**(*n*)

Calculates and return  $n!$  where *n* is an integer, or will be casted as an integer.

**fibonacci**(*n*)

Calculates and return the sum of the first *n*-th Fibonacci number

**permutation**(*items*, *n*=None)

Generates permutation of *n*-elements from the list of input items. For example, if *n* = 2, this function will generate permutations of 2-elements.

Adapted from Raymond Hettinger's comment to  
<http://code.activestate.com/recipes/474124/>

**combination**(*items*, *n*=None)

Generates combination of *n*-elements from the list of input items. For example, if *n* = 2, this function will generate combinations of 2-elements.

Adapted from Raymond Hettinger's comment to  
<http://code.activestate.com/recipes/474124/>

**sample\_wr**(*population*, *k*)

Chooses *k* random elements (with replacement) from a population. Adapted from Raymond Hettinger's comment to  
<http://code.activestate.com/recipes/273085/>

**Since:** version 0.2

<b>sample</b> ( <i>population</i> , <i>k</i> )
--

Chooses k random elements (without replacement) from a population.
--

<b>Since:</b> version 0.2
---------------------------

<b>summation</b> ( <i>x</i> )
-------------------------------

Sum of all the elements of x
------------------------------

<b>product</b> ( <i>x</i> )
-----------------------------

Product of all the elements of x, also known as series product
--

## 22.2 Variables

Name	Description
LOG2	<b>Value:</b> 0.69314718056
__package__	<b>Value:</b> 'copads'

## 22.3 Class Modulus2

Class for Modulus 2 arithmetics

**Status:** Tested methods

**Since:** version 0.1

### 22.3.1 Methods

<b>__init__</b> ( <i>self</i> , <i>input</i> =0)
--

<b>__add__</b> ( <i>self</i> , <i>other</i> )
---

<b>__mul__</b> ( <i>self</i> , <i>other</i> )
---

<b>__str__</b> ( <i>self</i> )
--------------------------------

## 22.4 Class Boolean

Class for Boolean arithmetics

**Status:** Tested methods

**Since:** version 0.1

#### 22.4.1 Methods

<code>--init--(<i>self</i>, <i>input</i>=0)</code>
--

<code>--add--(<i>self</i>, <i>other</i>)</code>
---

<code>--mul--(<i>self</i>, <i>other</i>)</code>
---

<code>--str--(<i>self</i>)</code>
-----------------------------------

## 23 Module *copads.prioritydictionary*

Priority Dictionary and Algorithms.

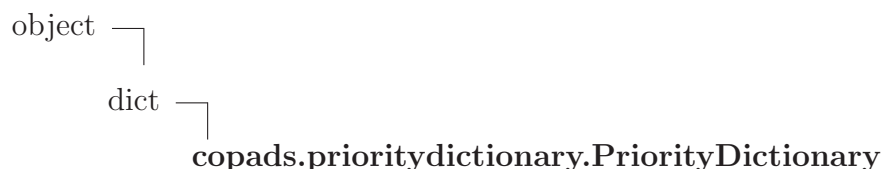
Copyright (c) Maurice H.T. Ling <mauriceling@acm.org>

Date created: 19th March 2008

### 23.1 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'copads'</code>

### 23.2 Class *PriorityDictionary*



This data structure acts almost like a dictionary, with two modifications: First, `D.smallest()` returns the value `x` minimizing `D[x]`. For this to work correctly, all values `D[x]` stored in the dictionary must be comparable. Second, iterating `'for x in D'` finds and removes the items from `D` in sorted order. Each item is not removed until the next item is requested, so `D[x]` will still return a useful value until the next iteration of the for-loop.

Adapted from: <http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/117228> Original author: David Eppstein

#### 23.2.1 Methods

<b><code>--init--(self)</code></b>
Initialize <code>priorityDictionary</code> by creating binary heap of pairs (value,key). Note that changing or removing a dict entry will not remove the old pair from the heap until it is found by <code>smallest()</code> or until the heap is rebuilt.
<b>Return Value</b>
new empty dictionary
Overrides: <code>object.__init__</code>

<b>smallest</b> ( <i>self</i> )
---------------------------------

Find smallest item after removing deleted items from heap.
--

<b>__iter__</b> ( <i>self</i> )
---------------------------------

Create destructive sorted iterator of priorityDictionary.
---

Overrides: dict.__iter__
--------------------------

<b>__setitem__</b> ( <i>self</i> , <i>key</i> , <i>val</i> )
--

Change value stored in dictionary and add corresponding pair to heap. Rebuilds the heap if the number of deleted items grows too large, to avoid memory leakage.
--

Overrides: dict.__setitem__
-----------------------------

<b>setDefault</b> ( <i>self</i> , <i>key</i> , <i>val</i> )
---

Reimplement setdefault to call our customized __setitem__.
--

### *Inherited from dict*

\_\_cmp\_\_(), \_\_contains\_\_(), \_\_delitem\_\_(), \_\_eq\_\_(), \_\_ge\_\_(), \_\_getattr\_\_(), \_\_getitem\_\_(),  
\_\_gt\_\_(), \_\_le\_\_(), \_\_len\_\_(), \_\_lt\_\_(), \_\_ne\_\_(), \_\_new\_\_(), \_\_repr\_\_(), \_\_sizeof\_\_(), clear(),  
copy(), fromkeys(), get(), has\_key(), items(), iteritems(), iterkeys(), itervalues(),  
keys(), pop(), popitem(), setdefault(), update(), values(), viewitems(), viewkeys(),  
viewvalues()

### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_setattr\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

#### 23.2.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

#### 23.2.3 Class Variables

Name	Description
<i>Inherited from dict</i>	
__hash__	

## 24 Module `copads.ragaraja`

Ragaraja Interpreter Date created: 16th August 2012 Licence: Python Software Foundation License version 2

Ragaraja is a derivative and massive extension of Brainfuck. This work is influenced by a large number of Brainfuck derivatives, other esoteric programming languages, and even assembly languages. Probably the most critical difference between Ragaraja and other Brainfuck derivatives is the large number of commands / instructions - 1000 possible commands / instructions, inspired by Nandi (follower of Lord Shiva) who was supposed to be the first author of Kama Sutra and wrote it in 1000 chapters.

Etymology: Ragaraja is the name of a Mahayana Buddhist deity from Esoteric traditions. The Chinese calls him Ai Ran Ming Wang. Ragaraja is one of the Wisdom Kings (a group of Bodhisattvas) and represents the state at which sexual excitement or agitation can be channeled towards enlightenment and passionate love can become compassion for all living things. Hence, I name this compilation/derivative/extension of Brainfuck in 1000 commands/instructions/opcode to signify the epitome, a channeling of raw urge to the love and compassion for and towards every being. May really be viewed as Brainfuck attaining enlightenment or Nirvana. Whoever that can remember all 1000 commands and use it, really deserves an award.

The interpreter environment consists of the following elements:

1. Array/Tape: A circular tape initialized with 30 thousand cells each with zero. This can be visualized as a 30,000 cell register machine. The number of cells can increase or decrease during runtime.
2. Source: The program
3. Input List: A list of data given to the execution environment at initialization.
4. Output List: A list of output from the execution. This may also be used as a secondary tape.

When the program terminates, all 4 elements are returned, and the interpreter terminates itself.

**See Also:** <http://esolangs.org/wiki/Ragaraja>

### 24.1 Functions

`instruction_padding(inst)`

**loop\_start**(*array, apointer, inputdata, output, source, spointer*)

Start loop. Will only enter loop if current cell is more than "0". If current cell is "0" or less, it will go to the end of the loop (command 015). if the loop is not closed, it will go to the end of the source.

**loop\_end**(*array, apointer, inputdata, output, source, spointer*)

End loop. However, it is possible to have an end loop operator (command 015) without a preceding start loop operator (command 014). In this case, the end loop operator (command 015) will be ignored and execution continues.



**tape\_move**(*array, apointer, inputdata, output, source, spointer*)

Moving tape pointer for more than one increment or decrement.

Instructions handled: 001: Move forward by 5 cells on tape. Equivalent to 5 times of "000". 002: Move forward by 10 cells on tape. Equivalent to 10 times of "000". 003: Move forward by  $N \times N$  cells on tape where  $N$  is the value of the current cell. If  $N$  is a decimal, it will move forward by the floor of  $N \times N$ . For example, if  $N$  is 4.2, this operation will tape pointer forward by 17 cells. As  $N \times N$  is always a positive number, it does not matter if the value of the current cell is positive or negative. 005: Move backward by 5 cells on tape. Equivalent to 5 times of "004". 006: Move backward by 10 cells on tape. Equivalent to 10 times of "004". 007: Move backward by  $N \times N$  cells on tape where  $N$  is the value of the current cell. If  $N$  is a decimal, it will move backward by the floor of  $N \times N$ . For example, if  $N$  is 4.2, this operation will tape pointer backward by 17 cells. As  $N \times N$  is always a positive number, it does not matter if the value of the current cell is positive or negative. 043: Move the tape cell pointer to the first cell. 044: Move the tape cell pointer to the last cell. 045: Move the tape cell pointer to the location determined by the last value of the output list. If the last value of the output list is more than the length of the tape, it will take the modulus of the length of the tape. For example, the last value of the output list is 5, the tape cell pointer will point to the 5th cell on the tape. 061: Move forward by the number of cells signified by the current cell. 062: Move backward by the number of cells signified by the current cell. 140: Move tape pointer to the centre of the tape. If the tape has odd number cells, it will move to the lower cell. For example, this instruction will move the tape pointer to the 500th cell of a 1000-cell tape, or 142nd of a 285-cell tape. 141: Move tape pointer to  $1/4$  the of the tape. If the tape has odd number cells, it will move to the lower cell. For example, this instruction will move the tape pointer to the 250th cell of a 1000-cell tape, or 71st of a 285-cell tape. 142: Move tape pointer to  $3/4$  the of the tape. If the tape has odd number cells, it will move to the lower cell. For example, this instruction will move the tape pointer to the 750th cell of a 1000-cell tape, or 213rd of a 285-cell tape. 143: Move tape pointer to the position as the integer value in the current cell. If the value of the cell is larger than the length of the tape, it will move to the modulus of the integer value in the current cell.

**accumulations**(*array, apointer, inputdata, output, source, spointer*)

Accumulate the tape cell by more than one increment or decrement.

Instructions handled: 009: Increase value of cell by 5. Equivalent to 5 times of "008". 010: Increase value of cell by 10. Equivalent to 10 times of "008". 012: Decrease value of cell by 5. Equivalent to 5 times of "011". 013: Decrease value of cell by 10. Equivalent to 10 times of "011". 032: Double current tape cell value. 033: Half current tape cell value.

**nBF\_random\_op**(*array, apointer, inputdata, output, source, spointer*)

NucleotideBF (nBF) random operations - to simulate ambiguous DNA bases.

Instructions handled: 050: Randomly execute "008" (increment by 1) or "000" (move forward by 1). Equivalent to "R" in NucleotideBF (nBF). 051: Randomly execute "011" (decrement by 1) or "004" (move backward by 1). Equivalent to "Y" in NucleotideBF (nBF). 052: Randomly execute "000" (move forward by 1) or "004" (move backward by 1). Equivalent to "S" in NucleotideBF (nBF). 053: Randomly execute "008" (increment by 1) or "011" (decrement by 1). Equivalent to "W" in NucleotideBF (nBF). 054: Randomly execute "000" (move forward by 1) or "011" (decrement by 1). Equivalent to "K" in NucleotideBF (nBF). 055: Randomly execute "004" (move backward by 1) or "008" (increment by 1). Equivalent to "M" in NucleotideBF (nBF). 056: Randomly execute "000" (move forward by 1) or "004" (move backward by 1) or "011" (decrement by 1). Equivalent to "B" in NucleotideBF (nBF). 057: Randomly execute "000" (move forward by 1) or "008" (increment by 1) or "011" (decrement by 1). Equivalent to "D" in NucleotideBF (nBF). 058: Randomly execute "004" (move backward by 1) or "008" (Increment by 1) or "011" (decrement by 1). Equivalent to "H" in NucleotideBF (nBF). 059: Randomly execute "000" (move forward by 1) or "004" (move backward by 1) or "008" (increment by 1). Equivalent to "V" in NucleotideBF (nBF). 060: Randomly execute "000" (move forward by 1) or "004" (move backward by 1) or "008" (increment by 1) or "011" (decrement by 1). Equivalent to "N" in NucleotideBF (nBF)

**tape.size**(*array, apointer, inputdata, output, source, spointer*)

Change the length of the tape during runtime.

Instructions handled: 016: Add one cell to the end of the tape. 017: Add 10 cells to the end of the tape. 018: Remove one cell from the end of the tape. If original tape pointer is at the last cell before removal operation, the tape pointer will point to the last cell after removal. 019: Remove 10 cells from the end of the tape. If original tape pointer is at the last cell before removal operation, the tape pointer will point to the last cell after removal. 034: Insert a cell after the current tape cell. For example, if current tape cell is 35, a cell initialized to zero will be added as cell 36. As a result, the tape is 1 cell longer. 035: Delete the current cell. As a result, the tape is 1 cell shorter. 036: Delete the current and append to the end of the output list. As a result, the tape is 1 cell shorter.

**source.move**(*array, apointer, inputdata, output, source, spointer*)

Moving the source without execution.

Instructions handled: 023: Move source pointer forward by one instruction without execution if the source pointer does not point beyond the length of the source after the move, otherwise, does not move the source pointer. 024: Move source pointer forward by 5 instruction without execution if the source pointer does not point beyond the length of the source after the move, otherwise, does not move the source pointer. 025: Move source pointer forward by 10 instruction without execution if the source pointer does not point beyond the length of the source after the move, otherwise, does not move the source pointer. 026: Move source pointer backward by one instruction without execution if the source pointer does not point beyond the length of the source after the move, otherwise, does not move the source pointer. 027: Move source pointer backward by 5 instruction without execution if the source pointer does not point beyond the length of the source after the move, otherwise, does not move the source pointer. 028: Move source pointer backward by 10 instruction without execution if the source pointer does not point beyond the length of the source after the move, otherwise, does not move the source pointer. 082: Skip next instruction if current cell is "0". Equivalent to "/" in [[Minimal]]. However, this operation will only execute if there is at least 1 more instruction from the current instruction. 083: Skip the number of instructions equivalent to the absolute integer value of the current cell if the source pointer does not point beyond the length of the source after the move, otherwise, does not move the source pointer. For example, if current cell is "5.6" or "5", the next 5 instructions will be skipped.

**set\_tape\_value**(*array*, *apointer*, *inputdata*, *output*, *source*, *spointer*)

Set values into tape cell by over-writing the original value.

Instructions handled: 084: Set current tape cell to "0". 085: Set current tape cell to "-1". 086: Set current tape cell to "1". 097: Set the value of the current cell to pi (3.14159265358979323846) 098: Set the value of the current cell to e (2.718281828459045) 187: Set all the cell values in the cells after the current cell to "0" (current cell is not affected). 188: Set all the cell values in the cells before the current cell to "0" (current cell is not affected). 189: Set all values in tape to "0". 190: Set all the cell values in the tape to the value of current cell. 191: Set all the cell values in the tape to the tape position of current cell. 192: Set all the cell values in the cells after the current cell to the value of current cell. 193: Set all the cell values in the cells before the current cell to the value of current cell. 194: Set all the cell values in the cells after the current cell to the tape position of current cell. 195: Set all the cell values in the cells before the current cell to the tape position of current cell.

**mathematics**(*array, apointer, inputdata, output, source, spointer*)

Performs mathematical and arithmetical operations.

Instructions handled: 065: Add the value of the current cell (n) and (n+1)th cell, and store the value in the current cell.  $\text{Array}[n] = \text{Array}[n] + \text{Array}[n+1]$  066: Add the value of the current cell (n) and first of the input list, and store the value in the current cell. 067: Add the value of the current cell (n) and last of the input list, and store the value in the current cell. 068: Subtract the value of the current cell (n) from (n+1)th cell, and store the value in the current cell.  $\text{Array}[n] = \text{Array}[n+1] - \text{Array}[n]$  069: Subtract the value of the current cell (n) from the first of the input list, and store the value in the current cell.  $\text{Array}[n] = \text{InputList}[0] - \text{Array}[n]$  070: Subtract the value of the current cell (n) from the last of the input list, and store the value in the current cell.  $\text{Array}[n] = \text{InputList}[-1] - \text{Array}[n]$  071: Multiply the value of the current cell (n) and (n+1)th cell, and store the value in the current cell.  $\text{Array}[n] = \text{Array}[n+1] * \text{Array}[n]$  072: Multiply the value of the current cell (n) and first of the input list, and store the value in the current cell. 073: Multiply the value of the current cell (n) and last of the input list, and store the value in the current cell. 074: Divide the value of the current cell (n) from (n+1)th cell, and store the value in the current cell.  $\text{Array}[n] = \text{Array}[n+1] / \text{Array}[n]$  075: Divide the value of the current cell (n) from the first of the input list, and store the value in the current cell.  $\text{Array}[n] = \text{InputList}[0] / \text{Array}[n]$  076: Divide the value of the current cell (n) from the last of the input list, and store the value in the current cell.  $\text{Array}[n] = \text{InputList}[-1] / \text{Array}[n]$  077: Modulus (remainder after division) the value of the current cell (n) from (n+1)th cell, and store the value in the current cell.  $\text{Array}[n] = \text{Array}[n+1] \% \text{Array}[n]$  078: Modulus (remainder after division) the value of the current cell (n) from the first of the input list, and store the value in the current cell.  $\text{Array}[n] = \text{InputList}[0] \% \text{Array}[n]$  079: Modulus (remainder after division) the value of the current cell (n) from the last of the input list, and store the value in the current cell.  $\text{Array}[n] = \text{InputList}[-1] \% \text{Array}[n]$  080: Floor the value of the current cell. For example, if the value of the current cell is 6.7, it will become 6. 087: Negate the value of the current cell. Positive value will be negative. Negative value will be positive. Equivalent to "-" in L00P 088: Calculate the sine of the value of the current cell (measured in radians) and replace. Equivalent to "s" in Grin.  $\text{Array}[n] = \text{sine}(\text{Array}[n])$  089: Calculate the cosine of the value of the current cell (measured in radians) and replace. Equivalent to "c" in Grin.  $\text{Array}[n] = \text{cosine}(\text{Array}[n])$  090: Calculate the tangent of the value of the current cell (measured in radians) and replace. Equivalent to "t" in Grin.  $\text{Array}[n] = \text{tangent}(\text{Array}[n])$  091: Calculate the arc sine of the value of the current cell (measured in radians) and replace. Equivalent to "S" in Grin.  $\text{Array}[n] = \text{arcsine}(\text{Array}[n])$  092: Calculate the arc cosine of the value of the current cell (measured in radians) and replace. Equivalent to "C" in Grin.  $\text{Array}[n] = \text{arccosine}(\text{Array}[n])$  093: Calculate the arc tangent of the value of the current cell (measured in radians) and replace. Equivalent to "T" in Grin.  $\text{Array}[n] = \text{arctangent}(\text{Array}[n])$  094: Calculate the reciprocal of the value of the current cell (measured in radians) and replace. Equivalent to "1" in Grin.  $\text{Array}[n] = 1/\text{Array}[n]$  095: Calculate the square root of the value of the current cell (measured in radians) and replace. Equivalent to "q" in Grin.  $\text{Array}[n] = \text{sqrt}(\text{Array}[n])$  096: Calculate the natural logarithm of the value of

**output\_IO**(*array, apointer, inputdata, output, source, spointer*)

Using output list as output storage or secondary tape, write and accept values from output list.

Instructions handled: 021: Output current tape cell location and append to the end of the output list. 022: Output current source location and append to the end of the output list. 037: Replace the current tape cell value with the last value of the output list, and delete the last value from the output list. 038: Replace the current tape cell value with the last value of the output list, without deleting the last value from the output list. 039: Replace the current tape cell value with the first value of the output list, and delete the first value from the output list. 040: Replace the current tape cell value with the first value of the output list, without deleting the first value from the output list. 041: Remove first value from the output list. 042: Remove last value from the output list. 172: Append all the cell values in the cells after the current cell to the end of output list (current cell is not appended to output list). 173: Append all the cell values in the cells after the current cell to the end of output list (current cell is not appended to output list), then set the values of all the cells after the current cell to "0" (current cell value is not affected). 174: Append all the cell values in the cells to the end of output list. 175: Append all the cell values in the cells to the end of output list and set the tape to "0".

**logic**(*array, apointer, inputdata, output, source, spointer*)

Logical operations

Instructions handled: 120: AND operator: Given positive numbers ( $>0$ ) as True and zero or negative numbers ( $\leq 0$ ) as False, store  $\text{Array}[\text{current}] \text{ AND } \text{Array}[\text{current}+1]$  in the current cell ( $\text{Array}[\text{current}]$ ) where "0" is False and "1" is True. 121: OR operator: Given positive numbers ( $>0$ ) as True and zero or negative numbers ( $\leq 0$ ) as False, store  $\text{Array}[\text{current}] \text{ OR } \text{Array}[\text{current}+1]$  in the current cell ( $\text{Array}[\text{current}]$ ) where "0" is False and "1" is True. 122: NOT operator: Given positive numbers ( $>0$ ) as True and zero or negative numbers ( $\leq 0$ ) as False, store NOT  $\text{Array}[\text{current}]$  in the current cell ( $\text{Array}[\text{current}]$ ) where "0" is False and "1" is True. 123: LESS-THAN operator: Store  $\text{Array}[\text{current}] < \text{Array}[\text{current}+1]$  in the current cell ( $\text{Array}[\text{current}]$ ) where "0" is False and "1" is True. 124: MORE-THAN operator: Store  $\text{Array}[\text{current}] > \text{Array}[\text{current}+1]$  in the current cell ( $\text{Array}[\text{current}]$ ) where "0" is False and "1" is True. 125: EQUAL operator: Store  $\text{Array}[\text{current}] = \text{Array}[\text{current}+1]$  in the current cell ( $\text{Array}[\text{current}]$ ) where "0" is False and "1" is True. 126: NOT-EQUAL operator: Store  $\text{Array}[\text{current}] \neq \text{Array}[\text{current}+1]$  in the current cell ( $\text{Array}[\text{current}]$ ) where "0" is False and "1" is True. 127: LESS-THAN-OR-EQUAL operator: Store  $\text{Array}[\text{current}] \leq \text{Array}[\text{current}+1]$  in the current cell ( $\text{Array}[\text{current}]$ ) where "0" is False and "1" is True. 128: MORE-THAN-OR-EQUAL operator: Store  $\text{Array}[\text{current}] \geq \text{Array}[\text{current}+1]$  in the current cell ( $\text{Array}[\text{current}]$ ) where "0" is False and "1" is True. 129: NAND operator: Given positive numbers ( $>0$ ) as True and zero or negative numbers ( $\leq 0$ ) as False, store  $\text{Array}[\text{current}] \text{ NAND } \text{Array}[\text{current}+1]$  in the current cell ( $\text{Array}[\text{current}]$ ) where "0" is False and "1" is True.  $\text{Array}[\text{current}] \text{ NAND } \text{Array}[\text{current}+1]$  is equivalent to NOT ( $\text{Array}[\text{current}] \text{ AND } \text{Array}[\text{current}+1]$ ) 130: NOR operator: Given positive numbers ( $>0$ ) as True and zero or negative numbers ( $\leq 0$ ) as False, store  $\text{Array}[\text{current}] \text{ NOR } \text{Array}[\text{current}+1]$  in the current cell ( $\text{Array}[\text{current}]$ ) where "0" is False and "1" is True.  $\text{Array}[\text{current}] \text{ NOR } \text{Array}[\text{current}+1]$  is equivalent to NOT ( $\text{Array}[\text{current}] \text{ OR } \text{Array}[\text{current}+1]$ )

**flipping**(*array, apointer, inputdata, output, source, spointer*)

Flipping of execution elements.

Instructions handled: 046: Flip the tape. The original first cell becomes the last cell but the tape pointer does not flip in location. 047: Flip the output list. 048: Flip the instruction list (source) but the source pointer does not flip in location.

**input\_IO**(*array, apointer, inputdata, output, source, spointer*)

Write to and accept values from input list.

Instructions handled: 064: Writes the first value of the input list into the current cell and without removing the value from the input list. If input list is empty, "0" will be written.

**tape\_manipulate**(*array, apointer, inputdata, output, source, spointer*)

Manipulating the tape.

Instructions handled: 081: Swap the value of the current cell (n) and (n+1)th cell. 133: Flip the tape from the cell after the current cell to the end of the tape (temporarily breaking the circularity of the tape). 161: Cut the tape before the current cell (n) and append it to the end of the tape and set tape pointer to 0.  $\langle -A- \rangle n \langle -B- \rangle \implies n \langle -B- \rangle \langle -A- \rangle$  162: Cut the tape after the current cell (n) and append it to the start of the tape and set tape pointer to the last cell.  $\langle -A- \rangle n \langle -B- \rangle \implies \langle -B- \rangle \langle -A- \rangle n$  163: Cut out the current cell and append it to the front of the tape and set tape pointer to 0.  $\langle -A- \rangle n \langle -B- \rangle \implies n \langle -A- \rangle \langle -B- \rangle$  164: Cut out the current cell and append it to the end of the tape and set tape pointer to the last cell.  $\langle -A- \rangle n \langle -B- \rangle \implies \langle -A- \rangle \langle -B- \rangle n$



**register\_IO**(*array, apointer, inputdata, output, source, spointer*)

Read from and write to register from tape cell.

Instructions handled: 201: Store value of current tape cell to register #1 202: Store value of current tape cell to register #2 203: Store value of current tape cell to register #3 204: Store value of current tape cell to register #4 205: Store value of current tape cell to register #5 206: Store value of current tape cell to register #6 207: Store value of current tape cell to register #7 208: Store value of current tape cell to register #8 209: Store value of current tape cell to register #9 210: Store value of current tape cell to register #10 211: Store value of current tape cell to register #11 212: Store value of current tape cell to register #12 213: Store value of current tape cell to register #13 214: Store value of current tape cell to register #14 215: Store value of current tape cell to register #15 216: Store value of current tape cell to register #16 217: Store value of current tape cell to register #17 218: Store value of current tape cell to register #18 219: Store value of current tape cell to register #19 220: Store value of current tape cell to register #20 221: Store value of current tape cell to register #21 222: Store value of current tape cell to register #22 223: Store value of current tape cell to register #23 224: Store value of current tape cell to register #24 225: Store value of current tape cell to register #25 226: Store value of current tape cell to register #26 227: Store value of current tape cell to register #27 228: Store value of current tape cell to register #28 229: Store value of current tape cell to register #29 230: Store value of current tape cell to register #30 231: Store value of current tape cell to register #31 232: Store value of current tape cell to register #32 233: Store value of current tape cell to register #33 234: Store value of current tape cell to register #34 235: Store value of current tape cell to register #35 236: Store value of current tape cell to register #36 237: Store value of current tape cell to register #37 238: Store value of current tape cell to register #38 239: Store value of current tape cell to register #39 240: Store value of current tape cell to register #40 241: Store value of current tape cell to register #41 242: Store value of current tape cell to register #42 243: Store value of current tape cell to register #43 244: Store value of current tape cell to register #44 245: Store value of current tape cell to register #45 246: Store value of current tape cell to register #46 247: Store value of current tape cell to register #47 248: Store value of current tape cell to register #48 249: Store value of current tape cell to register #49 250: Store value of current tape cell to register #50 251: Store value of current tape cell to register #51 252: Store value of current tape cell to register #52 253: Store value of current tape cell to register #53 254: Store value of current tape cell to register #54 255: Store value of current tape cell to register #55 256: Store value of current tape cell to register #56 257: Store value of current tape cell to register #57 258: Store value of current tape cell to register #58 259: Store value of current tape cell to register #59 260: Store value of current tape cell to register #60 261: Store value of current tape cell to register #61 262: Store value of current tape cell to register #62 263: Store value of current tape cell to register #63 264: Store value of current tape cell to register #64 265: Store value of current tape cell to register #65 266: Store value of current tape cell to register #66 267: Store value of current tape cell to register #67 268: Store value of current tape cell to register #68 269: Store value of current tape cell to register #69 270: Store value of current tape cell to register #70 271:

---

**jump\_identifier**(*array, apointer, inputdata, output, source, spointer*)

---

Defines jump location within the source tape. These instructions acts as pure identifiers and do not perform any operations.

Instructions handled: 200, 300, 400, 500, 600, 700, 800, 900

---



---

**not\_used**(*array, apointer, inputdata, output, source, spointer*)

---

Default do-nothing handler for not implemented instructions.

---



---

**source\_filter**(*source, sfilter=['000', '001', '002', '003', '004', '005', '006', '007', ...]*)

---

Checks a Ragaraja source code string and removes any instructions are not found in the filter

**Parameters**

**source:** Ragaraja source code string  
(*type=string*)

**sfilter:** list of instructions allowed. Default = `ragaraja_v1`. Other defined list are `nBF_instructions` (NucleotideBF) and `tested_ragaraja_instructions`.

**Return Value**

Ragaraja source code string

**Since:** version 0.4

---



---

**nBF\_to\_Ragaraja**(*source*)

---

Converts NucleotideBF (nBF) source code to Ragaraja source code

**Parameters**

**source:** NucleotideBF (nBF) source code  
(*type=string*)

**Return Value**

Ragaraja source code string

**Since:** version 0.4

---

**activate\_version**(*version=1*)

Function to only set tested instructions as usable.

**Parameters**

**version:** Define the version to activate. Default = 1. Allowable versions are

- 0 (all currently tested instructions)
- 0.1 (using NucleotideBF instructions)
- 1 (as defined in Ling, MHT. 2012. An Artificial Life Simulation Library Based on Genetic Algorithm, 3-Character Genetic Code and Biological Hierarchy. The Python Papers.)

**Since:** version 0.4

## 24.2 Variables

Name	Description
register	<b>Value:</b> [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...]
ragaraja	<b>Value:</b> {'000': forward, '001': tape_move, '002': tape_move, '003': ...}
ragaraja_v1	<b>Value:</b> ['000', '001', '002', '003', '004', '005', '006', '007', ...]
tested_ragaraja_instructions	<b>Value:</b> ['000', '001', '002', '003', '004', '005', '006', '007', ...]
nBF_instructions	<b>Value:</b> ['000', '004', '008', '011', '020', '050', '051', '052', ...]
--package--	<b>Value:</b> 'copads'

## 25 Module `copads.register_machine`

One dimensional tape/register machine Date created: 15th August 2012 Licence: Python Software Foundation License version 2

The machine consists of the following elements:

1. Array/Tape: A circular tape for operations to occur
2. Source: The program
3. Input List: A list of data given to the machine at initialization.
4. Output List: A list of output from the execution. This may also be used as a secondary tape.

When the program terminates, all 4 elements are returned, and the machine terminates itself.

### 25.1 Functions

```
interpret(source, functions, function_size=1, inputdata=[], array=None,
size=30, max_instructions=1000)
```

Interpreter loop.

#### Parameters

<b>source:</b>	Instructions to execute. ( <i>type=string</i> )
<b>functions:</b>	Dictionary of functions / operations.
<b>function_size:</b>	Length of each instruction. Default = 1 ( <i>type=integer</i> )
<b>inputdata:</b>	Any input data that the function may need. ( <i>type=list</i> )
<b>array:</b>	The endless tape in a Turing machine which is implemented as a circular list, making it virtually limitless. ( <i>type=list</i> )
<b>size:</b>	Length of the type (array). Default = 30 ( <i>type=integer</i> )
<b>max_instructions:</b>	The maximum number of instructions to execute. Default = 1000 ( <i>type=integer</i> )

## 25.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> None

## 26 Module copads.ring

Ring Data Structures and Algorithms.

Copyright (c) Maurice H.T. Ling <mauriceling@acm.org>

Date created: 19th March 2008

### 26.1 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> None

### 26.2 Class RingList

The RingList is a class implementing a circular list. The ring have a fixed size and when it is full and you append a new element, the first one will be deleted. The class lets you access to the data like a python list or like a string.

Adapted from: <http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/435902> Original author: Flavio Catalani

#### 26.2.1 Methods

<b><code>--init--</code></b> ( <i>self</i> , <i>length</i> )
Initializes an empty RingList of maximum given length as given in length parameter
<b><code>append</code></b> ( <i>self</i> , <i>x</i> )
Append parameter x to the data structure.
<b><code>get</code></b> ( <i>self</i> )
Retrieves the data.
<b><code>remove</code></b> ( <i>self</i> )
Removes the last element of the ring.

**size**(*self*)

Returns the current size of the ring.

**maxSize**(*self*)

Returns the maximum allowed size of the ring.

**\_\_str\_\_**(*self*)

Returns the ring as a string.

## 27 Module *copads.samplestatistics*

Data Structures and Algorithms for Data Collected from One or More Samples.

The following functions were adapted from [http://www.nmr.mgh.harvard.edu/Neural\\_Systems\\_Group/gary](http://www.nmr.mgh.harvard.edu/Neural_Systems_Group/gary) (assumes 1-dimensional list as input):

- *geometricMean*
- *harmonicMean*
- *arithmeticMean*
- *median*
- *medianScore*
- *mode*
- *moment*
- *variation*
- *skew*
- *kurtosis*

Copyright (c) Maurice H.T. Ling <[mauriceling@acm.org](mailto:mauriceling@acm.org)>

### 27.1 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'copads'

### 27.2 Class *SingleSample*

Class to hold a single sample, and provides calculations on the sample

#### 27.2.1 Methods

<code>--init--(<i>self</i>, <i>data</i>, <i>name</i>='Sample 1')</code>
<b><code>geometricMean(<i>self</i>)</code></b> Calculates the geometric mean of the data <b>Status:</b> Tested method <b>Since:</b> version 0.1



**harmonicMean**(*self*)

Calculates the harmonic mean of the data

**Status:** Tested method

**Since:** version 0.1

**arithmeticMean**(*self*)

Returns the arithmetic mean of the data

**Status:** Tested method

**Since:** version 0.1

**moment**(*self*, *moment*=1)

Calculates the nth moment about the mean for the data

**skew**(*self*)

Returns the skewness of the data, as defined in Numerical Recipies (alternate defn in CRC Standard Probability and Statistics, p.6.)

**Status:** Tested method

**Since:** version 0.1

**kurtosis**(*self*)

Returns the kurtosis of the data, as defined in Numerical Recipies (alternate defn in CRC Standard Probability and Statistics, p.6.)

**Status:** Tested method

**Since:** version 0.1

**variation**(*self*)

Returns the coefficient of variation in percentage, as defined in CRC Standard Probability and Statistics, p.6. Ref:  
[http://en.wikipedia.org/wiki/Coefficient\\_of\\_variation](http://en.wikipedia.org/wiki/Coefficient_of_variation)

**Status:** Tested method

**Since:** version 0.1

**range**(*self*)

Returns the range of the data (maximum - minimum)

**Status:** Tested method

**Since:** version 0.1

**variance**(*self*)

Returns the variance of the data

**Status:** Tested method**Since:** version 0.1**\_\_str\_\_**(*self*)**fullSummary**(*self*)

### 27.2.2 Class Variables

Name	Description
data	<b>Value:</b> None
rowcount	<b>Value:</b> 0
name	<b>Value:</b> None
summary	<b>Value:</b> {}

## 27.3 Class *SampleDistribution*

copads.statisticsdistribution.Distribution —  
copads.samplestatistics.SampleDistribution

### 27.3.1 Methods

**\_\_init\_\_**(*self*, *sampleData*)

Constructor method. The parameters are used to construct the probability distribution.

Overrides: copads.statisticsdistribution.Distribution.\_\_init\_\_ extit(inherited documentation)

***Inherited from copads.statisticsdistribution.Distribution(Section 29.3)***

CDF(), PDF(), inverseCDF(), kurtosis(), mean(), mode(), skew(), variance()

## 27.4 Class *TwoSample*

Class to hold a two samples, and provides calculations on the samples

## 27.4.1 Methods

```
__init__(self, data1, name1, data2, name2)
```

```
getSample(self, name)
```

```
listSamples(self)
```

```
covariance(self)
```

Calculates covariance using the formula:  $\text{Cov}(xy) = E(xy) - E(x)E(y)$

**Status:** Tested method

**Since:** version 0.3

```
linear_regression(self)
```

Calculates the first order linear regression model in the form of "y = mx + c" from the 2 samples where the first sample (data1 and name1 in initialization method) is taken as "X" and the second sample (data2 and name2 in initialization method is taken as "Y".

**Return Value**

Tuple of (gradient, intercept)

**Status:** Tested method

**Since:** version 0.1

```
pearson(self)
```

Calculates the Pearson's product-moment coefficient by the formula

$$\frac{(N * \text{sum\_xy}) - (\text{sum\_x} * \text{sum\_y})}{((N * \text{sum\_x2} - (\text{sum\_x})^{**2}) * (N * \text{sum\_y2} - (\text{sum\_y})^{**2}))^{**0.5}}$$

**Status:** Tested method

**Since:** version 0.1

## 27.4.2 Class Variables

Name	Description
sample	<b>Value:</b> {}
sample_name	<b>Value:</b> []

## 28 Module copads.set

Sets Data Structure and Algorithms.

Copyright (c) Maurice H.T. Ling <mauriceling@acm.org>

Date created: 19th March 2008

### 28.1 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> None

### 28.2 Class Set

The set class. It can contain mutable objects.

Adapted from: <http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/106469>

#### 28.2.1 Methods

<b><code>--init--(self, seq=None)</code></b>
--

The constructor. It can take any object giving an iterator as an optional argument to populate the new set.
---

<b><code>--str--(self)</code></b>
-----------------------------------

<b><code>copy(self)</code></b>
--------------------------------

Shallow copy of a set object.
-------------------------------

<b><code>--contains--(self, elem)</code></b>
--

<b><code>--len--(self)</code></b>
-----------------------------------

<b><code>items(self)</code></b>
---------------------------------

Returns a list of the elements in the set.
--

**add**(*self*, *elem*)

Add one element to the set.

**remove**(*self*, *elem*)

Remove an element from the set. Return an error if elem is not in the set.

**discard**(*self*, *elem*)

Remove an element from the set. Do nothing if elem is not in the set.

**\_\_iter\_\_**(*self*)**\_\_or\_\_**(*self*, *other*)

Union of two sets.

**\_\_sub\_\_**(*self*, *other*)

Difference of two sets.

**\_\_and\_\_**(*self*, *other*)

Intersection of two sets.

**\_\_add\_\_**(*self*, *other*)

Symmetric difference of two sets.

**\_\_mul\_\_**(*self*, *other*)

Cartesian product of two sets.

**\_\_lt\_\_**(*self*, *other*)

Returns 1 if the lhs set is contained but not equal to the rhs set.

**\_\_le\_\_**(*self*, *other*)

Returns 1 if the lhs set is contained in the rhs set.

**\_\_eq\_\_**(*self*, *other*)

Returns 1 if the sets are equal.

## 29 Module `copads.statisticsdistribution`

Classes for Various Statistical Distributions.

References:

- `Regress+` A compendium of common probability distributions (version 2.3) by Michael P. McLaughlin (mpmcl@mitre.org) [http://www.causascientia.org/math\\_stat/Dists/Compendium.pdf](http://www.causascientia.org/math_stat/Dists/Compendium.pdf)
- Hand-book on statistical distributions for experimentalists Internal report SUF-PFY/96-01. University of Stockholms by Christian Walck (walck@physto.se)

Distributions:

- `BetaDistribution(location, scale, p, q)`
  - `PowerFunctionDistribution(shape)`
- `BinomialDistribution(success, trial)`
  - `BernoulliDistribution(success)`
- `BradfordDistribution`
- `BurrDistribution`
- `CauchyDistribution(location=0.0, scale=1.0)`
  - `LorentzDistribution` (alias of `CauchyDistribution`)
- `ChiDistribution`
  - `HalfNormalDistribution(location, scale)`
  - `MaxwellDistribution(scale)`
  - `RayleighDistribution(scale)`
- `CosineDistribution(location=0.0, scale=1.0)`
- `DoubleGammaDistribution`
- `DoubleWeibullDistribution`
- `ExponentialDistribution(location=0.0, scale=1.0)`
  - `NegativeExponentialDistribution` (alias of `ExponentialDistribution`)
- `ExtremeLBDistribution`
- `FDistribution`
- `FiskDistribution`
  - `LogLogisticDistribution` (alias of `FiskDistribution`)
- `FoldedNormalDistribution`
- `GammaDistribution`
  - `ChiSquareDistribution(df)`
  - `ErlangDistribution(shape)`
  - `FurryDistribution` (alias of `GammaDistribution`)

- `GenLogisticDistribution`
- `GeometricDistribution(success=0.5)`
- `GumbelDistribution(location, scale)`
  - `FisherTippettDistribution` (alias of `GumbelDistribution`)
  - `GompertzDistribution` (alias of `GumbelDistribution`)
  - `LogWeibullDistribution` (alias of `GumbelDistribution`)
- `HyperbolicSecantDistribution`
- `HypergeometricDistribution`
- `InverseNormalDistribution`
  - `WaldDistribution` (alias of `InverseNormalDistribution`)
- `LaplaceDistribution`
  - `BilateralExponentialDistribution` (alias of `LaplaceDistribution`)
  - `DoubleExponentialDistribution` (alias of `LaplaceDistribution`)
- `LogarithmicDistribution(shape)`
- `LogisticDistribution`
  - `SechSquaredDistribution` (alias of `LogisticDistribution`)
- `LogNormalDistribution`
  - `AntiLogNormalDistribution` (alias of `LogNormalDistribution`)
  - `CobbDouglasDistribution` (alias of `LogNormalDistribution`)
- `NakagamiDistribution`
- `NegativeBinomialDistribution(success, target)`
  - `PascalDistribution(success, target)`
  - `PolyaDistribution` (alias of `NegativeBinomialDistribution`)
- `NormalDistribution()`
- `ParetoDistribution(location=1.0, shape=1.0)`
- `PoissonDistribution(expectation)`
- `RademacherDistribution()`
- `ReciprocalDistribution`
- `SemicircularDistribution(location=0.0, scale=1.0)`
- `TDistribution(location=0.0, scale=1.0, shape=2)`
- `TriangularDistribution`
- `UniformDistribution(location, scale)`
  - `RectangularDistribution` (alias of `UniformDistribution`)
- `WeibullDistribution`
  - `FrechetDistribution` (alias of `WeibullDistribution`)

Copyright (c) Maurice H.T. Ling <mauriceling@acm.org>

Date created: 17th August 2005

## 29.1 Functions

### **ErlangDistribution**(*location, scale, shape*)

Erlang distribution is an alias of Gamma distribution where the shape parameter is an integer.

#### **Parameters**

location:

scale:

shape:

**Status:** Tested method

**Since:** version 0.2

### **FurryDistribution**(*location, scale, shape*)

Furry distribution is an alias of Gamma distribution.

#### **Parameters**

location:

scale:

shape:

**Status:** Tested method

**Since:** version 0.2

### **FrechetDistribution**(\*\**parameters*)

Frechet distribution is an alias of Weibull distribution.

### **AntiLogNormalDistribution**(\*\**parameters*)

Anti-Lognormal distribution is an alias of Lognormal distribution.

### **BilateralExponentialDistribution**(\*\**parameters*)

Bilateral Exponential distribution is an alias of Laplace distribution.

### **CobbDouglasDistribution**(\*\**parameters*)

Cobb-Douglas distribution is an alias of Lognormal distribution.



**DoubleExponentialDistribution**(\*\**parameters*)

Double Exponential distribution is an alias of Laplace distribution.

**FisherTippettDistribution**(*location*, *scale*)

Fisher-Tippett distribution is an alias of Gumbel distribution.

**Parameters**

location:  $\eta$

scale:  $\theta$

**GompertzDistribution**(*location*, *scale*)

Gompertz distribution is an alias of Gumbel distribution.

**Parameters**

location:  $\eta$

scale:  $\theta$

**LogLogisticDistribution**(\*\**parameters*)

Log-Logistic distribution is an alias of Fisk distribution.

**LogWeibullDistribution**(*location*, *scale*)

Log-Weibull distribution is an alias of Gumbel distribution.

**Parameters**

location:  $\eta$

scale:  $\theta$

**LorentzDistribution**(\*\**parameters*)

Lorentz distribution is an alias of Cauchy distribution.

**NegativeExponentialDistribution**(\*\**parameters*)

Negative-exponential distribution is an alias of Exponential distribution.

**PolyaDistribution**(*success*, *target*)

Polya distribution is an alias of Negative Binomial distribution.

**Parameters**

success: probability of success;  $0 \leq \text{success} \leq 1$

target: a constant, target number of successes

<b>RectangularDistribution</b> (** <i>parameters</i> )
--

Rectangular distribution is an alias of Uniform distribution.
---

<b>SechSquaredDistribution</b> (** <i>parameters</i> )
--

Sech-squared distribution is an alias of Logistic distribution.
---

<b>WaldDistribution</b> (** <i>parameters</i> )
---

Wald distribution is an alias of Inverse Normal distribution.
---

## 29.2 Variables

Name	Description
BACKHOUSE	<b>Value:</b> 1.45607494858
CFRACT	<b>Value:</b> 1.0306408341
CGOLD	<b>Value:</b> 0.38196601125
FRODA	<b>Value:</b> 6.58088599102
GAMMA	<b>Value:</b> 0.577215664902
GOLD	<b>Value:</b> 1.61803398875
GOLOMB	<b>Value:</b> 0.624329988544
INV2PI	<b>Value:</b> 0.159154943092
INVLN10	<b>Value:</b> 0.434294481903
INVLN2	<b>Value:</b> 1.44269504089
INVPI180	<b>Value:</b> 57.2957795131
INVSQRT2PI	<b>Value:</b> 0.398942280401
LEHMER	<b>Value:</b> 0.592632718292
LENGYEL	<b>Value:</b> 1.09868580553
LN10	<b>Value:</b> 2.30258509299
LN2	<b>Value:</b> 0.69314718056
LNPI	<b>Value:</b> 1.14472988585
PARIS	<b>Value:</b> 1.09864196439
PI	<b>Value:</b> 3.14159265359
PI180	<b>Value:</b> 0.0174532925199
PI2	<b>Value:</b> 6.28318530718
PIDIV2	<b>Value:</b> 1.57079632679
PORTER	<b>Value:</b> 1.46707807943
RABBIT	<b>Value:</b> 0.709803442861
SQRT2	<b>Value:</b> 1.41421356237
SQRT2PI	<b>Value:</b> 2.50662827463
ZETA9	<b>Value:</b> 1.00200839283
__package__	<b>Value:</b> 'copads'

## 29.3 Class Distribution

**Known Subclasses:** `copads.samplestatistics.SampleDistribution`, `copads.statisticsdistribution.BernoulliDistribution`, `copads.statisticsdistribution.BetaDistribution`, `copads.statisticsdistribution.BinomialDistribution`, `copads.statisticsdistribution.BradfordDistribution`, `copads.statisticsdistribution.BurrDistribution`, `copads.statisticsdistribution.CauchyDistribution`, `copads.statisticsdistribution.ChiDistribution`, `copads.statisticsdistribution.GammaDistribution`, `copads.statisticsdistribution.CosineDistribution`, `copads.statisticsdistribution.DoubleGammaDistribution`, `copads.statisticsdistribution.DoubleWeibullDistribution`, `copads.statisticsdistribution.ExponentialDistribution`, `copads.statisticsdistribution.ExtremeLBDistribution`, `copads.statisticsdistribution.FDistribution`, `copads.statisticsdistribution.FiskDistribution`, `copads.statisticsdistribution.FoldedNormalDistribution`, `copads.statisticsdistribution.GenLogisticDistribution`, `copads.statisticsdistribution.GeometricDistribution`, `copads.statisticsdistribution.GumbelDistribution`, `copads.statisticsdistribution.HalfNormalDistribution`, `copads.statisticsdistribution.HyperbolicSecantDistribution`, `copads.statisticsdistribution.HypergeometricDistribution`, `copads.statisticsdistribution.LaplaceDistribution`, `copads.statisticsdistribution.LogNormalDistribution`, `copads.statisticsdistribution.LogarithmicDistribution`, `copads.statisticsdistribution.LogisticDistribution`, `copads.statisticsdistribution.MaxwellDistribution`, `copads.statisticsdistribution.NakagamiDistribution`, `copads.statisticsdistribution.NegativeBinomialDistribution`, `copads.statisticsdistribution.NormalDistribution`, `copads.statisticsdistribution.ParetoDistribution`, `copads.statisticsdistribution.PascalDistribution`, `copads.statisticsdistribution.PoissonDistribution`, `copads.statisticsdistribution.PowerFunctionDistribution`, `copads.statisticsdistribution.RademacherDistribution`, `copads.statisticsdistribution.RayleighDistribution`, `copads.statisticsdistribution.ReciprocalDistribution`, `copads.statisticsdistribution.SemicircularDistribution`, `copads.statisticsdistribution.TDistribution`, `copads.statisticsdistribution.TriangularDistribution`, `copads.statisticsdistribution.UniformDistribution`, `copads.statisticsdistribution.WeibullDistribution`

Abstract class for all statistical distributions. Due to the large variations of parameters for each distribution, it is unlikely to be able to standardize a parameter list for each method that is meaningful for all distributions. Instead, the parameters to construct each distribution is to be given as keyword arguments.

**See Also:** Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

### 29.3.1 Methods

<code>__init__(self, **parameters)</code>
---

Constructor method. The parameters are used to construct the probability distribution.
--

**CDF**(*self*, *x*)

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability. CDF is also known as density function.

**PDF**(*self*, *x*)

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

**inverseCDF**(*self*, *probability*, *start*=0.0, *step*=0.01)

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.

**mean**(*self*)

Gives the arithmetic mean of the sample.

**mode**(*self*)

Gives the mode of the sample, if closed-form is available.

**kurtosis**(*self*)

Gives the kurtosis of the sample.

**skew**(*self*)

Gives the skew of the sample.

**variance**(*self*)

Gives the variance of the sample.

## 29.4 Class *BetaDistribution*

copads.statisticsdistribution.Distribution

copads.statisticsdistribution.**BetaDistribution**

Class for Beta Distribution.

**See Also:** Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes

1:4

**Status:** Tested method**Since:** version 0.2**29.4.1 Methods****`__init__(self, location, scale, p, q)`**

Constructor method. The parameters are used to construct the probability distribution.

**Parameters****location:****scale:** upper bound**p:** shape parameter. Although no upper bound but seldom exceed 10.**q:** shape parameter. Although no upper bound but seldom exceed 10.Overrides: *copads.statisticsdistribution.Distribution.\_\_init\_\_***`CDF(self, x)`**

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: *copads.statisticsdistribution.Distribution.CDF***`PDF(self, x)`**

Partial Distribution Function, which gives the probability for particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: *copads.statisticsdistribution.Distribution.PDF***`inverseCDF(self, probability, start=0.0, step=0.01)`**

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: *copads.statisticsdistribution.Distribution.inverseCDF*

**mean**(*self*)

Gives the arithmetic mean of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mean`**mode**(*self*)

Gives the mode of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mode`**kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: `copads.statisticsdistribution.Distribution.kurtosis`**skew**(*self*)

Gives the skew of the sample.

Overrides: `copads.statisticsdistribution.Distribution.skew`**variance**(*self*)

Gives the variance of the sample.

Overrides: `copads.statisticsdistribution.Distribution.variance`**moment**(*self*, *r*)

Gives the r-th moment of the sample.

**random**(*self*)

Gives a random number based on the distribution.

## 29.5 Class **BinomialDistribution**

`copads.statisticsdistribution.Distribution``copads.statisticsdistribution.BinomialDistribution`

Class for Binomial Distribution.

**See Also:** Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

**Status:** Tested method

**Since:** version 0.2

### 29.5.1 Methods

**`__init__(self, success=0.5, trial=1000)`**

Constructor method. The parameters are used to construct the probability distribution.

**Parameters**

**success:** probability of success;  $0 \leq \text{success} \leq 1$

**trial:** number of Bernoulli trials

Overrides: *copads.statisticsdistribution.Distribution.\_\_init\_\_*

**`CDF(self, x)`**

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: *copads.statisticsdistribution.Distribution.CDF*

**`PDF(self, x)`**

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: *copads.statisticsdistribution.Distribution.PDF*

**`inverseCDF(self, probability, start=0, step=1)`**

It does the reverse of `CDF()` method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: *copads.statisticsdistribution.Distribution.inverseCDF*

**`mean(self)`**

Gives the arithmetic mean of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mean*

**mode**(*self*)

Gives the mode of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mode`**kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: `copads.statisticsdistribution.Distribution.kurtosis`**skew**(*self*)

Gives the skew of the sample.

Overrides: `copads.statisticsdistribution.Distribution.skew`**variance**(*self*)

Gives the variance of the sample.

Overrides: `copads.statisticsdistribution.Distribution.variance`

## 29.6 Class *CauchyDistribution*

`copads.statisticsdistribution.Distribution``copads.statisticsdistribution.CauchyDistribution`

Class for Cauchy Distribution.

**Status:** Tested method**Since:** version 0.4

### 29.6.1 Methods

**\_\_init\_\_**(*self*, *location*=0.0, *scale*=1.0)

Constructor method. The parameters are used to construct the probability distribution.

**Parameters****location:** the mean; default = 0.0**scale:** spread of the distribution,  $\lambda$ ; default = 1.0Overrides: `copads.statisticsdistribution.Distribution.__init__`



**CDF**(*self*, *x*)

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: *copads.statisticsdistribution.Distribution.CDF*

**PDF**(*self*, *x*)

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: *copads.statisticsdistribution.Distribution.PDF*

**inverseCDF**(*self*, *probability*, *start*=0.0, *step*=0.01)

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: *copads.statisticsdistribution.Distribution.inverseCDF*

**mean**(*self*)

Gives the arithmetic mean of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mean*

**mode**(*self*)

Gives the mode of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mode*

**median**(*self*)

Gives the median of the sample.

**quantile1**(*self*)

Gives the 1st quantile of the sample.

**quantile3**(*self*)

Gives the 3rd quantile of the sample.

**qmode**(*self*)

Gives the quantile of the mode of the sample.

<b>random</b> ( <i>self</i> , <i>seed</i> )
---

Gives a random number based on the distribution.
--

*Inherited from copads.statisticsdistribution.Distribution(Section 29.3)*

kurtosis(), skew(), variance()

## 29.7 Class CosineDistribution

copads.statisticsdistribution.Distribution

copads.statisticsdistribution.CosineDistribution

Cosine distribution is sometimes used as a simple approximation to Normal distribution.

**Status:** Tested method

**Since:** version 0.4

### 29.7.1 Methods

<b>__init__</b> ( <i>self</i> , <i>location</i> =0.0, <i>scale</i> =1.0)
--

Constructor method. The parameters are used to construct the probability distribution.
--

**Parameters**

**location:** the mean; default = 0.0

**scale:** spread of the distribution,  $\lambda$ ; default = 1.0

Overrides: copads.statisticsdistribution.Distribution.\_\_init\_\_

<b>CDF</b> ( <i>self</i> , <i>x</i> )
---------------------------------------

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.
--

Overrides: copads.statisticsdistribution.Distribution.CDF

<b>PDF</b> ( <i>self</i> , <i>x</i> )
---------------------------------------

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.
---

Overrides: copads.statisticsdistribution.Distribution.PDF

**inverseCDF**(*self*, *probability*, *start*=0.0, *step*=0.01)

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: `copads.statisticsdistribution.Distribution.inverseCDF`

**mean**(*self*)

Gives the arithmetic mean of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mean`

**mode**(*self*)

Gives the mode of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mode`

**median**(*self*)

Gives the median of the sample.

**kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: `copads.statisticsdistribution.Distribution.kurtosis`

**skew**(*self*)

Gives the skew of the sample.

Overrides: `copads.statisticsdistribution.Distribution.skew`

**variance**(*self*)

Gives the variance of the sample.

Overrides: `copads.statisticsdistribution.Distribution.variance`

**quantile1**(*self*)

Gives the 1st quantile of the sample.

**quantile3**(*self*)

Gives the 13rd quantile of the sample.

<b>qmean</b> ( <i>self</i> )
------------------------------

Gives the quantile of the arithmetic mean of the sample.
--

<b>qmode</b> ( <i>self</i> )
------------------------------

Gives the quantile of the mode of the sample.
---

## 29.8 Class **ExponentialDistribution**

copads.statisticsdistribution.Distribution

└─ copads.statisticsdistribution.**ExponentialDistribution**

Exponential distribution is the continuous version of Geometric distribution. It is also a special case of Gamma distribution where shape = 1

**Status:** Tested method

**Since:** version 0.4

### 29.8.1 Methods

<b>__init__</b> ( <i>self</i> , <i>location</i> =0.0, <i>scale</i> =1.0)
--

Constructor method. The parameters are used to construct the probability distribution.
--

**Parameters**

**location:** position of the distribution, default = 0.0

**scale:** spread of the distribution,  $\lambda$ ; default = 1.0

Overrides: copads.statisticsdistribution.Distribution.\_\_init\_\_

<b>CDF</b> ( <i>self</i> , <i>x</i> )
---------------------------------------

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.
--

Overrides: copads.statisticsdistribution.Distribution.CDF
---

**PDF**(*self*, *x*)

Partial Distribution Function, which gives the probability for the particular value of *x*, or the area under probability distribution from *x-h* to *x+h* for continuous distribution.

Overrides: `copads.statisticsdistribution.Distribution.PDF`

**inverseCDF**(*self*, *probability*, *start*=0.0, *step*=0.01)

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: `copads.statisticsdistribution.Distribution.inverseCDF`

**mean**(*self*)

Gives the arithmetic mean of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mean`

**mode**(*self*)

Gives the mode of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mode`

**median**(*self*)

Gives the median of the sample.

**kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: `copads.statisticsdistribution.Distribution.kurtosis`

**skew**(*self*)

Gives the skew of the sample.

Overrides: `copads.statisticsdistribution.Distribution.skew`

**variance**(*self*)

Gives the variance of the sample.

Overrides: `copads.statisticsdistribution.Distribution.variance`

**quantile1**(*self*)

Gives the 1st quantile of the sample.

**quantile3**(*self*)

Gives the 3rd quantile of the sample.

**qmean**(*self*)

Gives the quantile of the arithmetic mean of the sample.

**qmode**(*self*)

Gives the quantile of the mode of the sample.

**random**(*self*)

Gives a random number based on the distribution.

## 29.9 Class **FDistribution**

`copads.statisticsdistribution.Distribution``copads.statisticsdistribution.FDistribution`

Class for F Distribution.

**See Also:** Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

**Status:** Tested method**Since:** version 0.2

**29.9.1 Methods**


---

**`__init__(self, df1=1, df2=1)`**


---

Constructor method. The parameters are used to construct the probability distribution.

**Parameters**

**df1**: degrees of freedom for numerator

**df2**: degrees of freedom for denominator

Overrides: *copads.statisticsdistribution.Distribution.\_\_init\_\_*

---



---

**`CDF(self, x)`**


---

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: *copads.statisticsdistribution.Distribution.CDF*

---



---

**`PDF(self, x)`**


---

Partial Distribution Function, which gives the probability for particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: *copads.statisticsdistribution.Distribution.PDF*

---



---

**`inverseCDF(self, probability, start=0.0, step=0.01)`**


---

It does the reverse of `CDF()` method, it takes a probability value and the corresponding value on the x-axis.

Overrides: *copads.statisticsdistribution.Distribution.inverseCDF*

---



---

**`mean(self)`**


---

Gives the arithmetic mean of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mean*

---

***Inherited from *copads.statisticsdistribution.Distribution*(Section 29.3)***

`kurtosis()`, `mode()`, `skew()`, `variance()`

## 29.10 Class *GammaDistribution*

`copads.statisticsdistribution.Distribution`

`copads.statisticsdistribution.GammaDistribution`

**Known Subclasses:** `copads.statisticsdistribution.ChiSquareDistribution`

Class for Gamma Distribution.

**See Also:** Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

**Status:** Tested method

**Since:** version 0.2

### 29.10.1 Methods

**`__init__(self, location, scale, shape)`**

Constructor method. The parameters are used to construct the probability distribution.

**Parameters**

**location:**

**scale:**

**shape:**

Overrides: `copads.statisticsdistribution.Distribution.__init__`

**`CDF(self, x)`**

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: `copads.statisticsdistribution.Distribution.CDF`

**`inverseCDF(self, probability, start=0.0, step=0.01)`**

It does the reverse of `CDF()` method, it takes a probability value and the corresponding value on the x-axis.

Overrides: `copads.statisticsdistribution.Distribution.inverseCDF`



**mean**(*self*)

Gives the arithmetic mean of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mean***mode**(*self*)

Gives the mode of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mode***kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: *copads.statisticsdistribution.Distribution.kurtosis***skew**(*self*)

Gives the skew of the sample.

Overrides: *copads.statisticsdistribution.Distribution.skew***variance**(*self*)

Gives the variance of the sample.

Overrides: *copads.statisticsdistribution.Distribution.variance***qmean**(*self*)

Gives the quantile of the arithmetic mean of the sample.

**qmode**(*self*)

Gives the quantile of the mode of the sample.

*Inherited from copads.statisticsdistribution.Distribution(Section 29.3)*

PDF()

## 29.11 Class *ChiSquareDistribution*

*copads.statisticsdistribution.Distribution**copads.statisticsdistribution.GammaDistribution**copads.statisticsdistribution.ChiSquareDistribution*

Chi-square distribution is a special case of Gamma distribution where location = 0, scale = 2 and shape is twice that of the degrees of freedom.

**See Also:** Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

**Status:** Tested method

**Since:** version 0.2

### 29.11.1 Methods

**\_\_init\_\_**(*self*, *df*=2)

Constructor method. The parameters are used to construct the probability distribution.

**Parameters**

*df*: degrees of freedom

Overrides: *copads.statisticsdistribution.Distribution.\_\_init\_\_*

*Inherited from copads.statisticsdistribution.GammaDistribution(Section 29.10)*

CDF(), inverseCDF(), kurtosis(), mean(), mode(), qmean(), qmode(), skew(), variance()

*Inherited from copads.statisticsdistribution.Distribution(Section 29.3)*

PDF()

## 29.12 Class *GeometricDistribution*

*copads.statisticsdistribution.Distribution*

*copads.statisticsdistribution.GeometricDistribution*

Geometric distribution is the discrete version of Exponential distribution.

**See Also:** Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

**Status:** Tested method

**Since:** version 0.2

**29.12.1 Methods**


---

**`__init__(self, success=0.5)`**


---

Constructor method. The parameters are used to construct the probability distribution.

**Parameters**

**success:** probability of success;  $0 \leq \text{success} \leq 1$ ; default = 0.5

Overrides: *copads.statisticsdistribution.Distribution.\_\_init\_\_*

---



---

**`CDF(self, x)`**


---

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: *copads.statisticsdistribution.Distribution.CDF*

---



---

**`PDF(self, x)`**


---

Partial Distribution Function, which gives the probability for particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: *copads.statisticsdistribution.Distribution.PDF*

---



---

**`inverseCDF(self, probability, start=1, step=1)`**


---

It does the reverse of `CDF()` method, it takes a probability value and the corresponding value on the x-axis.

Overrides: *copads.statisticsdistribution.Distribution.inverseCDF*

---



---

**`mean(self)`**


---

Gives the arithmetic mean of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mean*

---



---

**`mode(self)`**


---

Gives the mode of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mode*

---

**variance**(*self*)

Gives the variance of the sample.

Overrides: `copads.statisticsdistribution.Distribution.variance`

*Inherited from copads.statisticsdistribution.Distribution(Section 29.3)*

kurtosis(), skew()

### 29.13 Class HypergeometricDistribution

copads.statisticsdistribution.Distribution

copads.statisticsdistribution.HypergeometricDistribut

Class for Hypergeometric distribution

**Status:** Tested method

**Since:** version 0.4

### 29.13.1 Methods

```
__init__(self, sample_size, population_size=100, population_success=50)
```

Constructor method. The parameters are used to construct the probability distribution.

## Parameters

**sample\_size:** sample size (not more than population size)  
(*type=integer*)

`population_size`: population size; default = 100  
(*type=integer*)

**population\_success:** number of successes in the population  
(cannot be more than population size);  
default = 10  
(*type=integer*)

Overrides: `copads.statisticsdistribution.Distribution.__init__`

**CDF**(*self*, *sample\_success*)

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value (*sample\_success*, an integer that is not more than sample size) on the x-axis where y-axis is the probability.

Overrides: *copads.statisticsdistribution.Distribution.CDF*

**PDF**(*self*, *sample\_success*)

Partial Distribution Function, which gives the probability for the particular value of x (*sample\_success*, an integer that is not more than sample size), or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: *copads.statisticsdistribution.Distribution.PDF*

**inverseCDF**(*self*, *probability*, *start*=1, *step*=1)

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: *copads.statisticsdistribution.Distribution.inverseCDF*

**mean**(*self*)

Gives the arithmetic mean of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mean*

**mode**(*self*)

Gives the mode of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mode*

**variance**(*self*)

Gives the variance of the sample.

Overrides: *copads.statisticsdistribution.Distribution.variance*

***Inherited from copads.statisticsdistribution.Distribution(Section 29.3)***

*kurtosis()*, *skew()*

## 29.14 Class **LogarithmicDistribution**



Class for Logarithmic Distribution.

**Status:** Tested method

**Since:** version 0.4

### 29.14.1 Methods

**`__init__(self, shape)`**

Constructor method. The parameters are used to construct the probability distribution.

**Parameters**

**shape:** the spread of the distribution

Overrides: `copads.statisticsdistribution.Distribution.__init__`

**`CDF(self, x)`**

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: `copads.statisticsdistribution.Distribution.CDF`

**`PDF(self, x)`**

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: `copads.statisticsdistribution.Distribution.PDF`

**`inverseCDF(self, probability, start=0.0, step=0.01)`**

It does the reverse of `CDF()` method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: `copads.statisticsdistribution.Distribution.inverseCDF`

**mean**(*self*)

Gives the arithmetic mean of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mean***mode**(*self*)

Gives the mode of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mode***variance**(*self*)

Gives the variance of the sample.

Overrides: *copads.statisticsdistribution.Distribution.variance**Inherited from copads.statisticsdistribution.Distribution(Section 29.3)**kurtosis()*, *skew()*

## 29.15 Class *NormalDistribution*

*copads.statisticsdistribution.Distribution**copads.statisticsdistribution.NormalDistribution*

Class for standardized normal distribution (area under the curve = 1)

**See Also:** Ling, MHT. 2009. Ten Z-Test Routines from Gopal Kanji's 100 Statistical Tests. The Python Papers Source Codes 1:5**Status:** Tested method**Since:** version 0.1

### 29.15.1 Methods

**\_\_init\_\_**(*self*)

Constructor method. The parameters are used to construct the probability distribution.

Overrides: *copads.statisticsdistribution.Distribution.\_\_init\_\_* extit(inherited documentation)

**CDF**(*self*, *x*)

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: *copads.statisticsdistribution.Distribution.CDF*

**PDF**(*self*, *x*)

Calculates the density (probability) at *x* by the formula  $f(x) = 1/(\sqrt{2\pi}) \sigma e^{-((x-\mu)^2/(2\sigma^2))}$  where  $\mu$  is the mean of the distribution and  $\sigma$  the standard deviation.

**Parameters**

**x**: probability at *x*

Overrides: *copads.statisticsdistribution.Distribution.PDF*

**inverseCDF**(*self*, *probability*, *start*=-10.0, *end*=10.0, *error*=1e-07)

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis, together with the cumulative probability.

**Parameters**

**probability**: probability under the curve from -infinity

**start**: lower boundary of calculation (default = -10)

**end**: upper boundary of calculation (default = 10)

**error**: error between the given and calculated probabilities (default = 10e-8)

**Return Value**

Returns a tuple (*start*, *cprob*) where 'start' is the standard deviation for the area under the curve from -infinity to the given 'probability' (+/- step). 'cprob' is the calculated area under the curve from -infinity to the returned 'start'.

Overrides: *copads.statisticsdistribution.Distribution.inverseCDF*

**mean**(*self*)

Gives the arithmetic mean of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mean* extit(inherited documentation)



**mode**(*self*)

Gives the mode of the sample, if closed-form is available.

Overrides: `copads.statisticsdistribution.Distribution.mode` extit(inherited documentation)

**kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: `copads.statisticsdistribution.Distribution.kurtosis` extit(inherited documentation)

**skew**(*self*)

Gives the skew of the sample.

Overrides: `copads.statisticsdistribution.Distribution.skew` extit(inherited documentation)

**variance**(*self*)

Gives the variance of the sample.

Overrides: `copads.statisticsdistribution.Distribution.variance` extit(inherited documentation)

**random**(*self*)

Gives a random number based on the distribution.

## 29.16 Class **PoissonDistribution**

`copads.statisticsdistribution.Distribution`

└─ **`copads.statisticsdistribution.PoissonDistribution`**

Class for Poisson Distribution. Poisson distribution is binomial distribution with very low success - that is, for rare events.

**See Also:** Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

**Status:** Tested method

**Since:** version 0.2

**29.16.1 Methods**

**`__init__(self, expectation=0.001)`**

Constructor method. The parameters are used to construct the probability distribution.

**Parameters**

**expectation:** mean success probability;  $\lambda$

Overrides: *copads.statisticsdistribution.Distribution.\_\_init\_\_*

**`CDF(self, x)`**

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: *copads.statisticsdistribution.Distribution.CDF*

**`PDF(self, x)`**

Partial Distribution Function, which gives the probability for particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: *copads.statisticsdistribution.Distribution.PDF*

**`inverseCDF(self, probability, start=0.001, step=1)`**

It does the reverse of `CDF()` method, it takes a probability value and the corresponding value on the x-axis.

Overrides: *copads.statisticsdistribution.Distribution.inverseCDF*

**`mean(self)`**

Gives the arithmetic mean of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mean*

**`mode(self)`**

Gives the mode of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mode*

<b>variance</b> ( <i>self</i> )
---------------------------------

Gives the variance of the sample.
-----------------------------------

Overrides: <code>copads.statisticsdistribution.Distribution.variance</code>
---

*Inherited from copads.statisticsdistribution.Distribution(Section 29.3)*

`kurtosis()`, `skew()`

## 29.17 Class *SemicircularDistribution*

`copads.statisticsdistribution.Distribution`

`copads.statisticsdistribution.SemicircularDistribution`

Class for Semicircular Distribution.

**Status:** Tested method

**Since:** version 0.4

### 29.17.1 Methods

<b>__init__</b> ( <i>self</i> , <i>location</i> =0.0, <i>scale</i> =1.0)
--

Constructor method. The parameters are used to construct the probability distribution.
--

**Parameters**

**location:** mean of the distribution, default = 0.0

**scale:** spread of the distribution, default = 1.0

Overrides: <code>copads.statisticsdistribution.Distribution.__init__</code>
---

<b>CDF</b> ( <i>self</i> , <i>x</i> )
---------------------------------------

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.
--

Overrides: <code>copads.statisticsdistribution.Distribution.CDF</code>
--

**PDF**(*self*, *x*)

Partial Distribution Function, which gives the probability for the particular value of *x*, or the area under probability distribution from *x-h* to *x+h* for continuous distribution.

Overrides: `copads.statisticsdistribution.Distribution.PDF`

**inverseCDF**(*self*, *probability*, *start*=-10.0, *step*=0.01)

It does the reverse of `CDF()` method, it takes a probability value and returns the corresponding value on the *x*-axis.

Overrides: `copads.statisticsdistribution.Distribution.inverseCDF`

**mean**(*self*)

Gives the arithmetic mean of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mean`

**mode**(*self*)

Gives the mode of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mode`

**kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: `copads.statisticsdistribution.Distribution.kurtosis`

**skew**(*self*)

Gives the skew of the sample.

Overrides: `copads.statisticsdistribution.Distribution.skew`

**variance**(*self*)

Gives the variance of the sample.

Overrides: `copads.statisticsdistribution.Distribution.variance`

**quantile1**(*self*)

Gives the 1st quantile of the sample.

**quantile3**(*self*)

Gives the 3rd quantile of the sample.

**qmean**(*self*)

Gives the quantile of the arithmetic mean of the sample.

**qmode**(*self*)

Gives the quantile of the mode of the sample.

## 29.18 Class **TDistribution**

`copads.statisticsdistribution.Distribution``copads.statisticsdistribution.TDistribution`

Class for Student's t-distribution.

**See Also:** Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square, F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

**Status:** Tested method**Since:** version 0.2

### 29.18.1 Methods

**\_\_init\_\_**(*self*, *location*=0.0, *scale*=1.0, *shape*=2)

Constructor method. The parameters are used to construct the probability distribution.

**Parameters****location:** default = 0.0**scale:** default = 1.0**shape:** degrees of freedom; default = 2Overrides: `copads.statisticsdistribution.Distribution.__init__`

**CDF**(*self*, *x*)

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: copads.statisticsdistribution.Distribution.CDF

**PDF**(*self*, *x*)

Calculates the density (probability) at x with n-th degrees of freedom as  $f(x) = \Gamma((n+1)/2) / (\sqrt{n} * \pi) \Gamma(n/2) (1 + x^2/n)^{-((n+1)/2)}$

for all real x. It has mean 0 (for  $n > 1$ ) and variance  $n/(n-2)$  (for  $n > 2$ ).

Overrides: copads.statisticsdistribution.Distribution.PDF

**inverseCDF**(*self*, *probability*, *start*=-10.0, *end*=10.0, *error*=1e-07)

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis, together with the cumulative probability.

**Parameters**

**probability:** probability under the curve from -infinity  
**start:** lower boundary of calculation (default = -10)  
**end:** upper boundary of calculation (default = 10)  
**error:** error between the given and calculated probabilities (default = 10e-8)

**Return Value**

Returns a tuple (start, cprob) where 'start' is the standard deviation for the area under the curve from -infinity to the given 'probability' (+/- step). 'cprob' is the calculated area under the curve from -infinity to the returned 'start'.

Overrides: copads.statisticsdistribution.Distribution.inverseCDF

**mean**(*self*)

Gives the arithmetic mean of the sample.

Overrides: copads.statisticsdistribution.Distribution.mean

**mode**(*self*)

Gives the mode of the sample.

Overrides: copads.statisticsdistribution.Distribution.mode

**kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: `copads.statisticsdistribution.Distribution.kurtosis`**skew**(*self*)

Gives the skew of the sample.

Overrides: `copads.statisticsdistribution.Distribution.skew`**variance**(*self*)

Gives the variance of the sample.

Overrides: `copads.statisticsdistribution.Distribution.variance`

## 29.19 Class **TriangularDistribution**

`copads.statisticsdistribution.Distribution``copads.statisticsdistribution.TriangularDistribution`

Class for Triangular Distribution.

**Status:** Tested method**Since:** version 0.4

**29.19.1 Methods**


---

**`__init__(self, upper_limit, peak, lower_limit=0)`**


---

Constructor method. The parameters are used to construct the probability distribution.

**Parameters**

**upper\_limit:** upper limit of the distribution

(*type=float*)

**peak:** peak of the distribution, which has to be between the lower and upper limits of the distribution

(*type=float*)

**lower\_limit:** lower limit of the distribution, default = 0

(*type=float*)

Overrides: *copads.statisticsdistribution.Distribution.\_\_init\_\_*

---



---

**`CDF(self, x)`**


---

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: *copads.statisticsdistribution.Distribution.CDF*

---



---

**`PDF(self, x)`**


---

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: *copads.statisticsdistribution.Distribution.PDF*

---



---

**`inverseCDF(self, probability, start=0, step=0.01)`**


---

It does the reverse of `CDF()` method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: *copads.statisticsdistribution.Distribution.inverseCDF*

---



---

**`mean(self)`**


---

Gives the arithmetic mean of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mean*

---



**mode**(*self*)

Gives the mode of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mode***kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: *copads.statisticsdistribution.Distribution.kurtosis***skew**(*self*)

Gives the skew of the sample.

Overrides: *copads.statisticsdistribution.Distribution.skew***variance**(*self*)

Gives the variance of the sample.

Overrides: *copads.statisticsdistribution.Distribution.variance***quantile1**(*self*)

Gives the 1st quantile of the sample.

**quantile3**(*self*)

Gives the 3rd quantile of the sample.

**qmean**(*self*)

Gives the quantile of the arithmetic mean of the sample.

**qmode**(*self*)

Gives the quantile of the mode of the sample.

## 29.20 Class **UniformDistribution**

*copads.statisticsdistribution.Distribution****copads.statisticsdistribution.UniformDistribution***

Class for Uniform distribution.

**See Also:** Ling, MHT. 2009. Compendium of Distributions, I: Beta, Binomial, Chi- Square,

F, Gamma, Geometric, Poisson, Student's t, and Uniform. The Python Papers Source Codes 1:4

**Status:** Tested method

**Since:** version 0.2

### 29.20.1 Methods

**`__init__(self, location, scale)`**

Constructor method. The parameters are used to construct the probability distribution.

**Parameters**

**location:**

**scale:**

Overrides: *copads.statisticsdistribution.Distribution.\_\_init\_\_*

**`CDF(self, x)`**

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: *copads.statisticsdistribution.Distribution.CDF*

**`PDF(self)`**

Partial Distribution Function, which gives the probability for particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: *copads.statisticsdistribution.Distribution.PDF*

**`inverseCDF(self, probability, start=0.0, step=0.01)`**

It does the reverse of `CDF()` method, it takes a probability value and the corresponding value on the x-axis.

Overrides: *copads.statisticsdistribution.Distribution.inverseCDF*

**`mean(self)`**

Gives the arithmetic mean of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mean*

**median**(*self*)

Gives the median of the sample.

**kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: *copads.statisticsdistribution.Distribution.kurtosis***skew**(*self*)

Gives the skew of the sample.

Overrides: *copads.statisticsdistribution.Distribution.skew***variance**(*self*)

Gives the variance of the sample.

Overrides: *copads.statisticsdistribution.Distribution.variance***quantile1**(*self*)

Gives the 1st quantile of the sample.

**quantile3**(*self*)

Gives the 3rd quantile of the sample.

**qmean**(*self*)

Gives the quantile of the arithmetic mean of the sample.

**random**(*self*, *lower*, *upper*)

Gives a random number based on the distribution.

*Inherited from copads.statisticsdistribution.Distribution(Section 29.3)**mode()***29.21 Class WeiBullDistribution***copads.statisticsdistribution.Distribution**copads.statisticsdistribution.WeiBullDistribution*

Class for Weibull distribution.

**Status:** Tested method

**Since:** version 0.4

### 29.21.1 Methods

**`__init__(self, location=1.0, scale=1.0)`**

Constructor method. The parameters are used to construct the probability distribution.

**Parameters**

**location:** position of the distribution, default = 1.0

**scale:** shape of the distribution, default = 1.0

Overrides: *copads.statisticsdistribution.Distribution.\_\_init\_\_*

**`CDF(self, x)`**

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: *copads.statisticsdistribution.Distribution.CDF*

**`PDF(self, x)`**

Partial Distribution Function, which gives the probability for the particular value of x, or the area under trobability distribution from x-h to x+h for continuous distribution.

Overrides: *copads.statisticsdistribution.Distribution.PDF*

**`inverseCDF(self, probability, start=0.0, step=0.01)`**

It does the reverse of `CDF()` method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: *copads.statisticsdistribution.Distribution.inverseCDF*

**`median(self)`**

Gives the median of the sample.

**`mode(self)`**

Gives the mode of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mode*

<b>random</b> ( <i>self</i> )
-------------------------------

Gives a random number based on the distribution.
--

*Inherited from copads.statisticsdistribution.Distribution(Section 29.3)*

kurtosis(), mean(), skew(), variance()

## 29.22 Class BernoulliDistribution

copads.statisticsdistribution.Distribution

└─ copads.statisticsdistribution.BernoulliDistribution

Bernoulli distribution is a special case of Binomial distribution where where number of trials = 1

### 29.22.1 Methods

<b>__init__</b> ( <i>self</i> , <i>success</i> )
--

Constructor method. The parameters are used to construct the probability distribution.
--

**Parameters**

**success:** probability of success; 0 <= success <= 1

Overrides: copads.statisticsdistribution.Distribution.\_\_init\_\_

<b>CDF</b> ( <i>self</i> , <i>x</i> )
---------------------------------------

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.
--

Overrides: copads.statisticsdistribution.Distribution.CDF

<b>PDF</b> ( <i>self</i> , <i>x</i> )
---------------------------------------

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.
---

Overrides: copads.statisticsdistribution.Distribution.PDF

**inverseCDF**(*self*, *probability*, *start*=0, *step*=1)

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: `copads.statisticsdistribution.Distribution.inverseCDF`

**mean**(*self*)

Gives the arithmetic mean of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mean`

**mode**(*self*)

Gives the mode of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mode`

**kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: `copads.statisticsdistribution.Distribution.kurtosis`

**skew**(*self*)

Gives the skew of the sample.

Overrides: `copads.statisticsdistribution.Distribution.skew`

**variance**(*self*)

Gives the variance of the sample.

Overrides: `copads.statisticsdistribution.Distribution.variance`

## 29.23 Class **BradfordDistribution**

`copads.statisticsdistribution.Distribution`

`copads.statisticsdistribution.BradfordDistribution`

Class for Bradford distribution.

**29.23.1 Methods**


---

**`__init__(self, location, scale, shape)`**


---

Constructor method. The parameters are used to construct the probability distribution.

**Parameters**

**location:**

**scale:**      upper bound

**shape:**

Overrides: *copads.statisticsdistribution.Distribution.\_\_init\_\_*

---



---

**`CDF(self, x)`**


---

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: *copads.statisticsdistribution.Distribution.CDF*

---



---

**`PDF(self, x)`**


---

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: *copads.statisticsdistribution.Distribution.PDF*

---



---

**`inverseCDF(self, probability, start=0.0, step=0.01)`**


---

It does the reverse of `CDF()` method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: *copads.statisticsdistribution.Distribution.inverseCDF*

---



---

**`mean(self)`**


---

Gives the arithmetic mean of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mean*

---



---

**`mode(self)`**


---

Gives the mode of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mode*

---

**kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: *copads.statisticsdistribution.Distribution.kurtosis***skew**(*self*)

Gives the skew of the sample.

Overrides: *copads.statisticsdistribution.Distribution.skew***variance**(*self*)

Gives the variance of the sample.

Overrides: *copads.statisticsdistribution.Distribution.variance***quantile1**(*self*)

Gives the 1st quantile of the sample.

**quantile3**(*self*)

Gives the 3rd quantile of the sample.

**qmean**(*self*)

Gives the quantile of the arithmetic mean of the sample.

**qmode**(*self*)

Gives the quantile of the mode of the sample.

**random**(*self*, *seed*)

Gives a random number based on the distribution.

## 29.24 Class **BurrDistribution**

*copads.statisticsdistribution.Distribution****copads.statisticsdistribution.BurrDistribution***

Burr distribution is the generalization of Fisk distribution. Burr distribution with  $D = 1$  becomes Fisk distribution.



**29.24.1 Methods**


---

**`__init__(self, location, scale, C, D)`**


---

Constructor method. The parameters are used to construct the probability distribution.

**Parameters**

**location:**

**scale:**

**C:**            shape

**D:**            shape

Overrides: *copads.statisticsdistribution.Distribution.\_\_init\_\_*

---



---

**`CDF(self, x)`**


---

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: *copads.statisticsdistribution.Distribution.CDF*

---



---

**`PDF(self, x)`**


---

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: *copads.statisticsdistribution.Distribution.PDF*

---



---

**`inverseCDF(self, probability, start=0.0, step=0.01)`**


---

It does the reverse of `CDF()` method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: *copads.statisticsdistribution.Distribution.inverseCDF*

---



---

**`mean(self)`**


---

Gives the arithmetic mean of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mean*

---



---

**`mode(self)`**


---

Gives the mode of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mode*

---

<b>variance</b> ( <i>self</i> )
---------------------------------

Gives the variance of the sample.
-----------------------------------

Overrides: <code>copads.statisticsdistribution.Distribution.variance</code>
---

<b>qmode</b> ( <i>self</i> )
------------------------------

Gives the quantile of the mode of the sample.
---

<b>random</b> ( <i>self</i> , <i>seed</i> )
---

Gives a random number based on the distribution.
--

*Inherited from copads.statisticsdistribution.Distribution(Section 29.3)*

kurtosis(), skew()

## 29.25 Class *ChiDistribution*

`copads.statisticsdistribution.Distribution`

`copads.statisticsdistribution.ChiDistribution`

Class for Chi distribution.

### 29.25.1 Methods

<b>inverseCDF</b> ( <i>self</i> , <i>probability</i> , <i>start</i> =0.0, <i>step</i> =0.01)
--

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.
--

Overrides: <code>copads.statisticsdistribution.Distribution.inverseCDF</code>
---

*Inherited from copads.statisticsdistribution.Distribution(Section 29.3)*

CDF(), PDF(), `--init--`(), kurtosis(), mean(), mode(), skew(), variance()

## 29.26 Class *DoubleGammaDistribution*

`copads.statisticsdistribution.Distribution`

`copads.statisticsdistribution.DoubleGammaDistribution`

Double Gamma distribution is the signed version of Gamma distribution.

**29.26.1 Methods**


---

**`__init__(self, location, scale, shape)`**


---

Constructor method. The parameters are used to construct the probability distribution.

**Parameters****location:****scale:****shape:**

Overrides: `copads.statisticsdistribution.Distribution.__init__`

---



---

**`CDF(self, x)`**


---

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: `copads.statisticsdistribution.Distribution.CDF`

---



---

**`PDF(self, x)`**


---

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: `copads.statisticsdistribution.Distribution.PDF`

---



---

**`inverseCDF(self, probability, start=0.0, step=0.01)`**


---

It does the reverse of `CDF()` method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: `copads.statisticsdistribution.Distribution.inverseCDF`

---



---

**`mean(self)`**


---

Gives the arithmetic mean of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mean`

---



---

**`skew(self)`**


---

Gives the skew of the sample.

Overrides: `copads.statisticsdistribution.Distribution.skew`

---

<b>variance</b> ( <i>self</i> )
---------------------------------

Gives the variance of the sample.
-----------------------------------

Overrides: copads.statisticsdistribution.Distribution.variance
--

<b>qmean</b> ( <i>self</i> )
------------------------------

Gives the quantile of the arithmetic mean of the sample.
--

*Inherited from copads.statisticsdistribution.Distribution(Section 29.3)*

kurtosis(), mode()

## 29.27 Class DoubleWeibullDistribution

copads.statisticsdistribution.Distribution

copads.statisticsdistribution.DoubleWeibullDistribution

Double Weibull distribution is the signed version of Weibull distribution.

### 29.27.1 Methods

<b>inverseCDF</b> ( <i>self</i> , <i>probability</i> , <i>start</i> =0.0, <i>step</i> =0.01)
--

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.
--

Overrides: copads.statisticsdistribution.Distribution.inverseCDF
--

*Inherited from copads.statisticsdistribution.Distribution(Section 29.3)*

CDF(), PDF(), \_\_init\_\_(), kurtosis(), mean(), mode(), skew(), variance()

## 29.28 Class ExtremeLBDistribution

copads.statisticsdistribution.Distribution

copads.statisticsdistribution.ExtremeLBDistribution

Class for Extreme LB distribution.

**29.28.1 Methods**

<b>inverseCDF</b> ( <i>self</i> , <i>probability</i> , <i>start</i> =0.0, <i>step</i> =0.01)
--

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.
--

Overrides: <i>copads.statisticsdistribution.Distribution.inverseCDF</i>
---

***Inherited from copads.statisticsdistribution.Distribution(Section 29.3)***

CDF(), PDF(), \_\_init\_\_(), kurtosis(), mean(), mode(), skew(), variance()

**29.29 Class FiskDistribution***copads.statisticsdistribution.Distribution**copads.statisticsdistribution.FiskDistribution*

Class for Fisk distribution.

**29.29.1 Methods**

<b>inverseCDF</b> ( <i>self</i> , <i>probability</i> , <i>start</i> =0.0, <i>step</i> =0.01)
--

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.
--

Overrides: <i>copads.statisticsdistribution.Distribution.inverseCDF</i>
---

***Inherited from copads.statisticsdistribution.Distribution(Section 29.3)***

CDF(), PDF(), \_\_init\_\_(), kurtosis(), mean(), mode(), skew(), variance()

**29.30 Class FoldedNormalDistribution***copads.statisticsdistribution.Distribution**copads.statisticsdistribution.FoldedNormalDistribution*

Class for Folded Normal distribution.

**29.30.1 Methods**

<b>inverseCDF</b> ( <i>self</i> , <i>probability</i> , <i>start</i> =0.0, <i>step</i> =0.01)
--

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.
--

Overrides: <i>copads.statisticsdistribution.Distribution.inverseCDF</i>
---

***Inherited from copads.statisticsdistribution.Distribution(Section 29.3)***

CDF(), PDF(), \_\_init\_\_(), kurtosis(), mean(), mode(), skew(), variance()

**29.31 Class GenLogisticDistribution***copads.statisticsdistribution.Distribution****copads.statisticsdistribution.GenLogisticDistribution***

Generalized Logistic distribution is a generalization of Logistic distribution. It becomes Logistic distribution when *shape* = 1

**29.31.1 Methods**

<b>inverseCDF</b> ( <i>self</i> , <i>probability</i> , <i>start</i> =0.0, <i>step</i> =0.01)
--

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.
--

Overrides: <i>copads.statisticsdistribution.Distribution.inverseCDF</i>
---

***Inherited from copads.statisticsdistribution.Distribution(Section 29.3)***

CDF(), PDF(), \_\_init\_\_(), kurtosis(), mean(), mode(), skew(), variance()

**29.32 Class GumbelDistribution***copads.statisticsdistribution.Distribution****copads.statisticsdistribution.GumbelDistribution***

Class for Gumbel Distribution.

**29.32.1 Methods**


---

**`__init__(self, location, scale)`**


---

Constructor method. The parameters are used to construct the probability distribution.

**Parameters**

**location:**  $\eta$

**scale:**  $\theta$

Overrides: *copads.statisticsdistribution.Distribution.\_\_init\_\_*

---



---

**`CDF(self, x)`**


---

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: *copads.statisticsdistribution.Distribution.CDF*

---



---

**`PDF(self, x)`**


---

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: *copads.statisticsdistribution.Distribution.PDF*

---



---

**`inverseCDF(self, probability, start=0.0, step=0.01)`**


---

It does the reverse of `CDF()` method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: *copads.statisticsdistribution.Distribution.inverseCDF*

---



---

**`mean(self)`**


---

Gives the arithmetic mean of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mean*

---



---

**`mode(self)`**


---

Gives the mode of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mode*

---

**median**(*self*)

Gives the median of the sample.

**kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: *copads.statisticsdistribution.Distribution.kurtosis***skew**(*self*)

Gives the skew of the sample.

Overrides: *copads.statisticsdistribution.Distribution.skew***variance**(*self*)

Gives the variance of the sample.

Overrides: *copads.statisticsdistribution.Distribution.variance***quantile1**(*self*)

Gives the 1st quantile of the sample.

**quantile3**(*self*)

Gives the 3rd quantile of the sample.

**qmean**(*self*)

Gives the quantile of the arithmetic mean of the sample.

**qmode**(*self*)

Gives the quantile of the mode of the sample.

**random**(*self*, *seed*)

Gives a random number based on the distribution.

### 29.33 Class **HalfNormalDistribution**

*copads.statisticsdistribution.Distribution****copads.statisticsdistribution.HalfNormalDistribution***

Half Normal distribution is a special case of Chi distribution where shape (also degrees of



freedom) = 1, and Folded Normal distribution where location = 0

### 29.33.1 Methods

**`__init__(self, **parameters)`**

Constructor method. The parameters are used to construct the probability distribution.

Overrides: `copads.statisticsdistribution.Distribution.__init__`

**`CDF(self, x)`**

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: `copads.statisticsdistribution.Distribution.CDF`

**`PDF(self, x)`**

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: `copads.statisticsdistribution.Distribution.PDF`

**`inverseCDF(self, probability, start=0.0, step=0.01)`**

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: `copads.statisticsdistribution.Distribution.inverseCDF`

**`mean(self)`**

Gives the arithmetic mean of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mean`

**`mode(self)`**

Gives the mode of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mode`

**kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: `copads.statisticsdistribution.Distribution.kurtosis`**skew**(*self*)

Gives the skew of the sample.

Overrides: `copads.statisticsdistribution.Distribution.skew`**variance**(*self*)

Gives the variance of the sample.

Overrides: `copads.statisticsdistribution.Distribution.variance`

## 29.34 Class **HyperbolicSecantDistribution**

`copads.statisticsdistribution.Distribution``copads.statisticsdistribution.HyperbolicSecantDistribution`

Class for Hyperbolic Secant Distribution.

### 29.34.1 Methods

**\_\_init\_\_**(*self*, *location*, *scale*)

Constructor method. The parameters are used to construct the probability distribution.

**Parameters****location:****scale:**Overrides: `copads.statisticsdistribution.Distribution.__init__`**CDF**(*self*, *x*)

Cumulative Distribution Function, which gives the cumulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: `copads.statisticsdistribution.Distribution.CDF`

**PDF**(*self*, *x*)

Partial Distribution Function, which gives the probability for the particular value of *x*, or the area under probability distribution from *x-h* to *x+h* for continuous distribution.

Overrides: `copads.statisticsdistribution.Distribution.PDF`

**inverseCDF**(*self*, *probability*, *start*=0.0, *step*=0.01)

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: `copads.statisticsdistribution.Distribution.inverseCDF`

**mean**(*self*)

Gives the arithmetic mean of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mean`

**mode**(*self*)

Gives the mode of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mode`

**kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: `copads.statisticsdistribution.Distribution.kurtosis`

**skew**(*self*)

Gives the skew of the sample.

Overrides: `copads.statisticsdistribution.Distribution.skew`

**variance**(*self*)

Gives the variance of the sample.

Overrides: `copads.statisticsdistribution.Distribution.variance`

**qmean**(*self*)

Gives the quantile of the arithmetic mean of the sample.

<b>qmode</b> ( <i>self</i> )
------------------------------

Gives the quantile of the mode of the sample.
---

## 29.35 Class LaplaceDistribution



### 29.35.1 Methods

<b>inverseCDF</b> ( <i>self</i> , <i>probability</i> , <i>start</i> =0.0, <i>step</i> =0.01)
--

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.
--

Overrides: copads.statisticsdistribution.Distribution.inverseCDF
--

***Inherited from copads.statisticsdistribution.Distribution(Section 29.3)***

CDF(), PDF(), \_\_init\_\_(), kurtosis(), mean(), mode(), skew(), variance()

## 29.36 Class LogisticDistribution



### 29.36.1 Methods

<b>inverseCDF</b> ( <i>self</i> , <i>probability</i> , <i>start</i> =0.0, <i>step</i> =0.01)
--

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.
--

Overrides: copads.statisticsdistribution.Distribution.inverseCDF
--

***Inherited from copads.statisticsdistribution.Distribution(Section 29.3)***

CDF(), PDF(), \_\_init\_\_(), kurtosis(), mean(), mode(), skew(), variance()

**29.37 Class LogNormalDistribution****29.37.1 Methods**

<b><code>__init__(self, a, b)</code></b>
--

Constructor method. The parameters are used to construct the probability distribution.
--

Overrides: <code>copads.statisticsdistribution.Distribution.__init__</code>
---

<b><code>inverseCDF(self, probability, start=0.0, step=0.01)</code></b>
---

It does the reverse of <code>CDF()</code> method, it takes a probability value and returns the corresponding value on the x-axis.
---

Overrides: <code>copads.statisticsdistribution.Distribution.inverseCDF</code>
---

<b><code>mean(self)</code></b>
--------------------------------

Gives the arithmetic mean of the sample.
--

Overrides: <code>copads.statisticsdistribution.Distribution.mean</code>
---

<b><code>random(self)</code></b>
----------------------------------

Gives a random number based on the distribution.
--

*Inherited from `copads.statisticsdistribution.Distribution` (Section 29.3)*

`CDF()`, `PDF()`, `kurtosis()`, `mode()`, `skew()`, `variance()`

**29.38 Class MaxwellDistribution**

Maxwell distribution is a special case of Chi distribution where location = 0 and shape (degrees of freedom) = 3

**29.38.1 Methods****`__init__(self, scale)`**

Constructor method.

**Parameters****scale:**Overrides: *copads.statisticsdistribution.Distribution.\_\_init\_\_***`CDF(self, x)`**

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: *copads.statisticsdistribution.Distribution.CDF***`PDF(self, x)`**

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: *copads.statisticsdistribution.Distribution.PDF***`inverseCDF(self, probability, start=0.0, step=0.01)`**It does the reverse of `CDF()` method, it takes a probability value and the corresponding value on the x-axis.Overrides: *copads.statisticsdistribution.Distribution.inverseCDF***`mean(self)`**

Gives the arithmetic mean of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mean***`mode(self)`**

Gives the mode of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mode***`kurtosis(self)`**

Gives the kurtosis of the sample.

Overrides: *copads.statisticsdistribution.Distribution.kurtosis*



**29.40.1 Methods****`__init__(self, success, target)`**

Constructor method. The parameters are used to construct the probability distribution.

**Parameters**

**success:** probability of success;  $0 \leq \text{success} \leq 1$

**target:** a constant, target number of successes

Overrides: *copads.statisticsdistribution.Distribution.\_\_init\_\_*

**`CDF(self, x)`**

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: *copads.statisticsdistribution.Distribution.CDF*

**`PDF(self, x)`**

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: *copads.statisticsdistribution.Distribution.PDF*

**`inverseCDF(self, probability, start=0, step=1)`**

It does the reverse of `CDF()` method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: *copads.statisticsdistribution.Distribution.inverseCDF*

**`mean(self)`**

Gives the arithmetic mean of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mean*

**`mode(self)`**

Gives the mode of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mode*

*Inherited from copads.statisticsdistribution.Distribution(Section 29.3)*

`kurtosis()`, `skew()`, `variance()`



## 29.41 Class *ParetoDistribution*

`copads.statisticsdistribution.Distribution`

`copads.statisticsdistribution.ParetoDistribution`

Class for Pareto Distribution.

### 29.41.1 Methods

**`__init__(self, location=1.0, scale=1.0)`**

Constructor method. The parameters are used to construct the probability distribution.

**Parameters**

**location:** also the scale; default = 1.0

**scale:**  $\lambda$ ; default = 1.0

Overrides: `copads.statisticsdistribution.Distribution.__init__`

**`CDF(self, x)`**

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: `copads.statisticsdistribution.Distribution.CDF`

**`PDF(self, x)`**

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: `copads.statisticsdistribution.Distribution.PDF`

**`inverseCDF(self, probability, start=0.0, step=0.01)`**

It does the reverse of `CDF()` method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: `copads.statisticsdistribution.Distribution.inverseCDF`

**`mean(self)`**

Gives the arithmetic mean of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mean`

**mode**(*self*)

Gives the mode of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mode***median**(*self*)

Gives the median of the sample.

**kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: *copads.statisticsdistribution.Distribution.kurtosis***skew**(*self*)

Gives the skew of the sample.

Overrides: *copads.statisticsdistribution.Distribution.skew***variance**(*self*)

Gives the variance of the sample.

Overrides: *copads.statisticsdistribution.Distribution.variance***quantile1**(*self*)

Gives the 1st quantile of the sample.

**quantile3**(*self*)

Gives the 3rd quantile of the sample.

**qmean**(*self*)

Gives the quantile of the arithmetic mean of the sample.

**qmode**(*self*)

Gives the quantile of the mode of the sample.

**random**(*self*)

Gives a random number based on the distribution.

**29.42 Class PascalDistribution**

copads.statisticsdistribution.Distribution

└─ **copads.statisticsdistribution.PascalDistribution**

Class for Pascal Distribution. Pascal Distribution is a form of Negative Binomial Distribution where the 'target' is an integer

**29.42.1 Methods****\_\_init\_\_**(*self*, *success*, *target*)

Constructor method.

**Parameters****success**: probability of success;  $0 \leq \text{success} \leq 1$ **target**: a constant, target number of successes

Overrides: copads.statisticsdistribution.Distribution.\_\_init\_\_

**CDF**(*self*, *x*)

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: copads.statisticsdistribution.Distribution.CDF

**PDF**(*self*, *x*)

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: copads.statisticsdistribution.Distribution.PDF

**inverseCDF**(*self*, *probability*, *start*=0.0, *step*=0.01)

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: copads.statisticsdistribution.Distribution.inverseCDF

**mean**(*self*)

Gives the arithmetic mean of the sample.

Overrides: copads.statisticsdistribution.Distribution.mean

**mode**(*self*)

Gives the mode of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mode`**kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: `copads.statisticsdistribution.Distribution.kurtosis`**skew**(*self*)

Gives the skew of the sample.

Overrides: `copads.statisticsdistribution.Distribution.skew`**variance**(*self*)

Gives the variance of the sample.

Overrides: `copads.statisticsdistribution.Distribution.variance`

### 29.43 Class **PowerFunctionDistribution**

`copads.statisticsdistribution.Distribution``copads.statisticsdistribution.PowerFunctionDistribution`

Class for Power Function Distribution. It is a form of Beta Distribution.

#### 29.43.1 Methods

**\_\_init\_\_**(*self*, *shape*)

Constructor method.

**Parameters****shape:**Overrides: `copads.statisticsdistribution.Distribution.__init__`

**CDF**(*self*, *x*)

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: *copads.statisticsdistribution.Distribution.CDF*

**PDF**(*self*, *x*)

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: *copads.statisticsdistribution.Distribution.PDF*

**inverseCDF**(*self*, *probability*, *start*=0.0, *step*=0.01)

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: *copads.statisticsdistribution.Distribution.inverseCDF*

**mean**(*self*)

Gives the arithmetic mean of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mean*

**mode**(*self*)

Gives the mode of the sample.

Overrides: *copads.statisticsdistribution.Distribution.mode*

**kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: *copads.statisticsdistribution.Distribution.kurtosis*

**skew**(*self*)

Gives the skew of the sample.

Overrides: *copads.statisticsdistribution.Distribution.skew*

**variance**(*self*)

Gives the variance of the sample.

Overrides: *copads.statisticsdistribution.Distribution.variance*

## 29.44 Class RademacherDistribution

copads.statisticsdistribution.Distribution

copads.statisticsdistribution.RademacherDistribution

Class for Rademacher Distribution.

### 29.44.1 Methods

**`__init__(self)`**

Constructor method.

Overrides: copads.statisticsdistribution.Distribution.\_\_init\_\_

**`CDF(self, x)`**

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.

Overrides: copads.statisticsdistribution.Distribution.CDF

**`PDF(self, x)`**

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.

Overrides: copads.statisticsdistribution.Distribution.PDF

**`inverseCDF(self, probability, start=0.0, step=0.01)`**

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: copads.statisticsdistribution.Distribution.inverseCDF

**`mean(self)`**

Gives the arithmetic mean of the sample.

Overrides: copads.statisticsdistribution.Distribution.mean

**`skew(self)`**

Gives the skew of the sample.

Overrides: copads.statisticsdistribution.Distribution.skew

<b>variance</b> ( <i>self</i> )
---------------------------------

Gives the variance of the sample.
-----------------------------------

Overrides: <code>copads.statisticsdistribution.Distribution.variance</code>
---

*Inherited from copads.statisticsdistribution.Distribution(Section 29.3)*

`kurtosis()`, `mode()`

## 29.45 Class *RayleighDistribution*

`copads.statisticsdistribution.Distribution`

`copads.statisticsdistribution.RayleighDistribution`

Rayleigh distribution is a special case of Chi distribution where location = 0 and shape (degrees of freedom) = 2

### 29.45.1 Methods

<b>__init__</b> ( <i>self</i> , <i>scale</i> )
--

Constructor method.
---------------------

<b>Parameters</b>
-------------------

<b>scale:</b>
---------------

Overrides: <code>copads.statisticsdistribution.Distribution.__init__</code>
---

<b>CDF</b> ( <i>self</i> , <i>x</i> )
---------------------------------------

Cummulative Distribution Function, which gives the cummulative probability (area under the probability curve) from -infinity or 0 to a give x-value on the x-axis where y-axis is the probability.
--

Overrides: <code>copads.statisticsdistribution.Distribution.CDF</code>
--

<b>PDF</b> ( <i>self</i> , <i>x</i> )
---------------------------------------

Partial Distribution Function, which gives the probability for the particular value of x, or the area under probability distribution from x-h to x+h for continuous distribution.
---

Overrides: <code>copads.statisticsdistribution.Distribution.PDF</code>
--

**inverseCDF**(*self*, *probability*, *start*=0.0, *step*=0.01)

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.

Overrides: `copads.statisticsdistribution.Distribution.inverseCDF`

**mean**(*self*)

Gives the arithmetic mean of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mean`

**mode**(*self*)

Gives the mode of the sample.

Overrides: `copads.statisticsdistribution.Distribution.mode`

**kurtosis**(*self*)

Gives the kurtosis of the sample.

Overrides: `copads.statisticsdistribution.Distribution.kurtosis`

**skew**(*self*)

Gives the skew of the sample.

Overrides: `copads.statisticsdistribution.Distribution.skew`

**variance**(*self*)

Gives the variance of the sample.

Overrides: `copads.statisticsdistribution.Distribution.variance`

## 29.46 Class *ReciprocalDistribution*

`copads.statisticsdistribution.Distribution`

`copads.statisticsdistribution.ReciprocalDistribution`



**29.46.1 Methods**

<b>inverseCDF</b> ( <i>self</i> , <i>probability</i> , <i>start</i> =0.0, <i>step</i> =0.01)
--

It does the reverse of CDF() method, it takes a probability value and returns the corresponding value on the x-axis.
--

Overrides: <i>copads.statisticsdistribution.Distribution.inverseCDF</i>
---

***Inherited from copads.statisticsdistribution.Distribution(Section 29.3)***

CDF(), PDF(), \_\_init\_\_(), kurtosis(), mean(), mode(), skew(), variance()

## 30 Module *copads.tree*

Tree Data Structures and Algorithms.

Copyright (c) Maurice H.T. Ling <mauriceling@acm.org>

Date created: 19th March 2008

### 30.1 Variables

Name	Description
BLACK	<b>Value:</b> 0
RED	<b>Value:</b> 1
<code>--package--</code>	<b>Value:</b> 'copads'

### 30.2 Class *RBTreeIter*



#### 30.2.1 Methods

<b><code>--init--</code></b> ( <i>self</i> , <i>tree</i> ) x. <b><code>--init--</code></b> (...) initializes x; see help(type(x)) for signature Overrides: object. <b><code>--init--</code></b> extit(inherited documentation)
<b><code>--iter--</code></b> ( <i>self</i> ) Return the current item in the container
<b><code>next</code></b> ( <i>self</i> ) Return the next item in the container Once we go off the list we stay off even if the list changes

#### *Inherited from object*

`--delattr--`(), `--format--`(), `--getattribute--`(), `--hash--`(), `--new--`(), `--reduce--`(), `--reduce_ex--`(), `--repr--`(), `--setattr--`(), `--sizeof--`(), `--str--`(), `--subclasshook--`()

### 30.2.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 30.3 Class `OrderedBinaryTree`

Adapted from <http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/286239> Original author: Lawrence Oluyede

### 30.3.1 Methods

<code>__init__(self)</code>
Initializes the root member
<code>addNode(self, data)</code>
<code>insert(self, root, data)</code>
<code>lookup(self, root, target)</code>
<code>minValue(self, root)</code>
<code>maxDepth(self, root)</code>
<code>size(self, root)</code>
<code>printTree(self, root)</code>
<code>printRevTree(self, root)</code>

## 30.4 Class `RBTree`



**Known Subclasses:** `copads.tree.RBDict`, `copads.tree.RBList`

### 30.4.1 Methods

**`__init__(self, cmpfn=<built-in function cmp>, unique=True)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit`(inherited documentation)

**`__len__(self)`**

**`__del__(self)`**

**`__str__(self)`**

`str(x)`

Overrides: `object.__str__` `exitit`(inherited documentation)

**`__repr__(self)`**

`repr(x)`

Overrides: `object.__repr__` `exitit`(inherited documentation)

**`__iter__(self)`**

**`rotateLeft(self, x)`**

**`rotateRight(self, x)`**

**`insertFixup(self, x)`**

**`insertNode(self, key, value)`**

**`deleteFixup(self, x)`**

**`deleteNode(self, z, all=True)`**

**`findNode(self, key)`**

**`traverseTree(self, f)`**

<b>nodesByTraversal</b> ( <i>self</i> )
---

return all nodes as a list
----------------------------

<b>nodes</b> ( <i>self</i> )
------------------------------

return all nodes as a list
----------------------------

<b>firstNode</b> ( <i>self</i> )
----------------------------------

<b>lastNode</b> ( <i>self</i> )
---------------------------------

<b>nextNode</b> ( <i>self</i> , <i>prev</i> )
---

returns None if there isn't one
---------------------------------

<b>prevNode</b> ( <i>self</i> , <i>next</i> )
---

returns None if there isn't one
---------------------------------

### *Inherited from object*

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`,  
`--setattr--()`, `--sizeof--()`, `--subclasshook--()`

### 30.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

## 30.5 Class RBList



List class uses same object for key and value Assumes you are putting sortable items into the list.

### 30.5.1 Methods

**`__init__(self, list=[], cmpfn=<built-in function cmp>, unique=True)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

**`__getitem__(self, index)`**

**`__delitem__(self, index)`**

**`__contains__(self, item)`**

**`__str__(self)`**

`str(x)`

Overrides: `object.__str__` `extit`(inherited documentation)

**`findNodeByIndex(self, index)`**

**`insert(self, item)`**

**`append(self, item)`**

**`count(self, item)`**

**`index(self, item)`**

**`extend(self, otherList)`**

**`pop(self, index=None)`**

**`remove(self, item, all=True)`**

**`reverse(self)`**

**`sort(self)`**

**clear**(*self*)

delete all entries

**values**(*self*)**reverseValues**(*self*)*Inherited from copads.tree.RBTree(Section 30.4)*

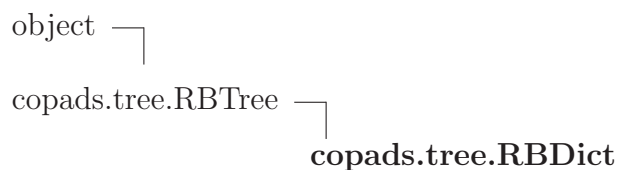
`__del__()`, `__iter__()`, `__len__()`, `__repr__()`, `deleteFixup()`, `deleteNode()`, `findNode()`, `firstNode()`, `insertFixup()`, `insertNode()`, `lastNode()`, `nextNode()`, `nodes()`, `nodes-ByTraversal()`, `prevNode()`, `rotateLeft()`, `rotateRight()`, `traverseTree()`

*Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__sizeof__()`, `__subclasshook__()`

**30.5.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**30.6 Class RBDict****30.6.1 Methods****\_\_init\_\_**(*self*, *dict*=`{}`, *cmpfn*=<built-in function cmp>)*x*.`__init__`(...) initializes *x*; see `help(type(x))` for signatureOverrides: `object.__init__` `extit`(inherited documentation)

**`--str--(self)`**

`str(x)`

Overrides: `object.__str__` `exitit`(inherited documentation)

**`--repr--(self)`**

`repr(x)`

Overrides: `object.__repr__` `exitit`(inherited documentation)

**`--getitem--(self, key)`**

**`--setitem--(self, key, value)`**

**`--delitem--(self, key)`**

**`get(self, key, default=None)`**

**`keys(self)`**

**`values(self)`**

**`items(self)`**

**`has_key(self, key)`**

**`clear(self)`**

delete all entries

**`copy(self)`**

return shallow copy

**`update(self, other)`**

Add all items from the supplied mapping to this one.

Will overwrite old entries with new ones.

**`setdefault(self, key, value=None)`**

*Inherited from `copads.tree.RBTree`(Section 30.4)*



`__del__()`, `__iter__()`, `__len__()`, `deleteFixup()`, `deleteNode()`, `findNode()`, `firstNode()`,  
`insertFixup()`, `insertNode()`, `lastNode()`, `nextNode()`, `nodes()`, `nodesByTraversal()`,  
`prevNode()`, `rotateLeft()`, `rotateRight()`, `traverseTree()`

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__setattr__()`, `__sizeof__()`, `__subclasshook__()`

**30.6.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 31 Module *copads.treenodes*

Node Classes for Tree Data Structures.

Copyright (c) Maurice H.T. Ling <mauriceling@acm.org>

Date created: 19th March 2008

### 31.1 Variables

Name	Description
BLACK	<b>Value:</b> 0
RED	<b>Value:</b> 1
<code>--package--</code>	<b>Value:</b> None

### 31.2 Class *RBNode*



#### 31.2.1 Methods

```
--init--(self, key=None, value=None, color=1)
```

`x.--init--(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.--init--` extit(inherited documentation)

```
--str--(self)
```

`str(x)`

Overrides: `object.--str--` extit(inherited documentation)

```
--nonzero--(self)
```

```
--len--(self)
```

---

imitate sequence

<code>--getitem--(<i>self</i>, <i>index</i>)</code>
---

imitate sequence
------------------

***Inherited from object***

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`,  
`--repr--()`, `--setattr--()`, `--sizeof--()`, `--subclasshook--()`

**31.2.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

**31.3 Class *BinaryNode*****31.3.1 Methods**

<code>--init--(<i>self</i>, <i>data</i>)</code>
---

**31.4 Class *TreeNode*****31.4.1 Methods**

<code>--init--(<i>self</i>, <i>data</i>)</code>
---

**31.5 Class *FourNode*****31.5.1 Methods**

<code>--init--(<i>self</i>, <i>data</i>)</code>
---

## Index

- copads (*package*), 2–4
  - copads.array (*module*), 5–6
    - copads.array.ParallelArray (*class*), 5–6
  - copads.bag (*module*), 7–9
    - copads.bag.Bag (*class*), 7–9
  - copads.colormap (*module*), 10
    - copads.colormap.floatRgb (*function*), 10
    - copads.colormap.htmlRgb (*function*), 10
    - copads.colormap.rgb (*function*), 10
    - copads.colormap.strRgb (*function*), 10
  - copads.constants (*module*), 11–12
  - copads.copadsexceptions (*module*), 13–46
    - copads.copadsexceptions.ArrayError (*class*), 44–45
    - copads.copadsexceptions.CopadsError (*class*), 14–15
    - copads.copadsexceptions.DistanceError (*class*), 38–39
    - copads.copadsexceptions.DistanceInputSizeError (*class*), 39–40
    - copads.copadsexceptions.DistributionError (*class*), 33–34
    - copads.copadsexceptions.DistributionFunctionError (*class*), 37–38
    - copads.copadsexceptions.DistributionParameterError (*class*), 35–37
    - copads.copadsexceptions.EdgeNotFoundError (*class*), 25–26
    - copads.copadsexceptions.FunctionParameterTypeError (*class*), 42–43
    - copads.copadsexceptions.FunctionParameterValueError (*class*), 43–44
    - copads.copadsexceptions.GraphEdgeSizeMismatchError (*class*), 30–31
    - copads.copadsexceptions.GraphError (*class*), 24–25
    - copads.copadsexceptions.GraphParameterError (*class*), 31–32
    - copads.copadsexceptions.MatrixAdditionError (*class*), 18–19
    - copads.copadsexceptions.MatrixArithmeticError (*function*), 59 (*class*), 15–17
    - copads.copadsexceptions.MatrixDeterminantError (*class*), 23–24
    - copads.copadsexceptions.MatrixError (*class*), 15
    - copads.copadsexceptions.MatrixMinorError (*class*), 22–23
    - copads.copadsexceptions.MatrixMultiplicationError (*class*), 17–18
    - copads.copadsexceptions.MatrixSquareError (*class*), 19–20
    - copads.copadsexceptions.MatrixTraceError (*class*), 20–22
    - copads.copadsexceptions.MaxIterationsException (*class*), 45–46
    - copads.copadsexceptions.NormalDistributionTypeError (*class*), 34–35
    - copads.copadsexceptions.NotAdjacencyGraphMatrixError (*class*), 28–30
    - copads.copadsexceptions.StatisticsError (*class*), 32–33
    - copads.copadsexceptions.TreeError (*class*), 40–41
    - copads.copadsexceptions.TreeNodeTypeError (*class*), 41–42
    - copads.copadsexceptions.UnknownGraphMatrixError (*class*), 27–28
    - copads.copadsexceptions.VertexNotFoundError (*class*), 26–27
  - copads.dataframe (*module*), 47–49
    - copads.dataframe.dataframe (*class*), 47–49
  - copads.dose\_entities (*module*), 50–59
    - copads.dose\_entities.Organism (*class*), 50–53
    - copads.dose\_entities.Population (*class*), 53–55
    - copads.dose\_entities.World (*class*), 55–59
  - copads.dose\_executor (*module*), 59–60
    - copads.dose\_executor.set\_instruction\_version
    - copads.dose\_executor.simulate (*function*),

- 59
- copads.dose\_executor.write\_parameters (*function*), 59
- copads.dose\_parameters (*module*), 61–63
- copads.dose\_world (*module*), 64–68
  - copads.dose\_world.World (*class*), 64–68
- copads.genetic (*module*), 69–84
  - copads.genetic.Chromosome (*class*), 71–75
  - copads.genetic.crossover (*function*), 69
  - copads.genetic.Organism (*class*), 75–80
  - copads.genetic.Population (*class*), 80–84
  - copads.genetic.population\_constructor (*function*), 69
  - copads.genetic.population\_simulate (*function*), 70
- copads.graph (*module*), 85–88
  - copads.graph.Graph (*class*), 85–88
- copads.hash (*module*), 89–91
  - copads.hash.backward\_file\_hash (*function*), 89
  - copads.hash.cfh (*function*), 90
  - copads.hash.forward\_file\_hash (*function*), 89
- copads.hypothesis (*module*), 92–115
  - copads.hypothesis.Chisq2Variance (*function*), 107
  - copads.hypothesis.Chisq2x2 (*function*), 111
  - copads.hypothesis.ChisqFit (*function*), 109
  - copads.hypothesis.Chisquare2xKtable (*function*), 112
  - copads.hypothesis.ChisquareKx2table (*function*), 111
  - copads.hypothesis.ChiSquarePopVar (*function*), 104
  - copads.hypothesis.F2CorrelatedObs (*function*), 105
  - copads.hypothesis.F2Count (*function*), 108
  - copads.hypothesis.FVarianceRatio (*function*), 105
  - copads.hypothesis.MedianTestfor2Pop (*function*), 113
  - copads.hypothesis.SpearmanCorrelation (*function*), 113
  - copads.hypothesis.t1Mean (*function*), 98
  - copads.hypothesis.t2Mean2EqualVariance (*function*), 99
  - copads.hypothesis.t2Mean2UnequalVariance (*function*), 100
  - copads.hypothesis.test (*function*), 93
  - copads.hypothesis.tPaired (*function*), 101
  - copads.hypothesis.tPearsonCorrelation (*function*), 102
  - copads.hypothesis.tRegressionCoefficient (*function*), 102
  - copads.hypothesis.tx2testofKcounts (*function*), 110
  - copads.hypothesis.Z1Mean1Variance (*function*), 93
  - copads.hypothesis.Z1Proportion (*function*), 96
  - copads.hypothesis.Z2Count (*function*), 97
  - copads.hypothesis.Z2Mean1Variance (*function*), 94
  - copads.hypothesis.Z2Mean2Variance (*function*), 95
  - copads.hypothesis.Z2PearsonCorrelation (*function*), 104
  - copads.hypothesis.Z2Proportion (*function*), 97
  - copads.hypothesis.ZCorrProportion (*function*), 106
  - copads.hypothesis.ZPearsonCorrelation (*function*), 103
  - copads.hypothesis.ZtestLogOddsRatio (*function*), 114
- copads.lc\_bf (*module*), 116–117
  - copads.lc\_bf.accept\_predefined (*function*), 116
  - copads.lc\_bf.backward (*function*), 116
  - copads.lc\_bf.call\_out (*function*), 116
  - copads.lc\_bf.cbf\_end\_loop (*function*), 117
  - copads.lc\_bf.cbf\_start\_loop (*function*), 117

- copads.lc.bf.decrement (*function*), 116
- copads.lc.bf.forward (*function*), 116
- copads.lc.bf.increment (*function*), 116
- copads.matrix (*module*), 118–129
  - copads.matrix.isSparse (*function*), 120
  - copads.matrix.isVector (*function*), 118
  - copads.matrix.Matrix (*class*), 123–127
  - copads.matrix.smDiag (*function*), 120
  - copads.matrix.smDot (*function*), 120
  - copads.matrix.smDotDot (*function*), 120
  - copads.matrix.smIdentity (*function*), 120
  - copads.matrix.smTranspose (*function*), 120
  - copads.matrix.SparseMatrix (*class*), 127–129
  - copads.matrix.vAcos (*function*), 119
  - copads.matrix.vAsin (*function*), 119
  - copads.matrix.vAtan (*function*), 119
  - copads.matrix.vAtan2 (*function*), 119
  - copads.matrix.vCos (*function*), 119
  - copads.matrix.vCosh (*function*), 119
  - copads.matrix.vDot (*function*), 118
  - copads.matrix.Vector (*class*), 120–123
  - copads.matrix.vExp (*function*), 118
  - copads.matrix.vLog (*function*), 118
  - copads.matrix.vLog10 (*function*), 118
  - copads.matrix.vNorm (*function*), 118
  - copads.matrix.vOnes (*function*), 118
  - copads.matrix.vPow (*function*), 119
  - copads.matrix.vRandom (*function*), 118
  - copads.matrix.vSin (*function*), 119
  - copads.matrix.vSinh (*function*), 119
  - copads.matrix.vSqrt (*function*), 119
  - copads.matrix.vSum (*function*), 118
  - copads.matrix.vTan (*function*), 119
  - copads.matrix.vTanh (*function*), 119
  - copads.matrix.vZeros (*function*), 118
- copads.n.bf (*module*), 130
  - copads.n.bf.random\_op (*function*), 130
- copads.neural (*module*), 131–137
  - copads.neural.Brain (*class*), 134–137
  - copads.neural.dtf\_linear (*function*), 131
  - copads.neural.itf\_linear (*function*), 131
  - copads.neural.Neuron (*class*), 131–134
  - copads.neural.tf\_linear (*function*), 131
- copads.nrpy (*module*), 138–149
  - copads.nrpy.bessi (*function*), 139
  - copads.nrpy.bessi0 (*function*), 139
  - copads.nrpy.bessi1 (*function*), 139
  - copads.nrpy.bessj (*function*), 139
  - copads.nrpy.bessj0 (*function*), 140
  - copads.nrpy.bessj1 (*function*), 140
  - copads.nrpy.bessk (*function*), 140
  - copads.nrpy.bessk0 (*function*), 141
  - copads.nrpy.bessk1 (*function*), 141
  - copads.nrpy.bessy (*function*), 141
  - copads.nrpy.bessy0 (*function*), 141
  - copads.nrpy.bessy1 (*function*), 142
  - copads.nrpy.beta (*function*), 142
  - copads.nrpy.betacf (*function*), 142
  - copads.nrpy.betai (*function*), 143
  - copads.nrpy.bico (*function*), 143
  - copads.nrpy.cdf\_binomial (*function*), 148
  - copads.nrpy.cdf\_poisson (*function*), 148
  - copads.nrpy.chebev (*function*), 143
  - copads.nrpy.erf (*function*), 144
  - copads.nrpy.erfc (*function*), 144
  - copads.nrpy.erfcc (*function*), 144
  - copads.nrpy.expdev (*function*), 145
  - copads.nrpy.factln (*function*), 145
  - copads.nrpy.gammln (*function*), 145
  - copads.nrpy.gammp (*function*), 146
  - copads.nrpy.gammq (*function*), 146
  - copads.nrpy.gcf (*function*), 147
  - copads.nrpy.gser (*function*), 147
  - copads.nrpy.mdian1 (*function*), 147
  - copads.nrpy.moment (*function*), 147
  - copads.nrpy.qgaus (*function*), 148
- copads.objectdistance (*module*), 150–168
  - copads.objectdistance.Anderberg (*function*), 153
  - copads.objectdistance.binarize (*function*), 150
  - copads.objectdistance.Bray\_Curtis (*function*), 166
  - copads.objectdistance.Buser (*function*), 159
  - copads.objectdistance.Canberra (*function*),

- 166
- copads.objectdistance.compare (*function*), 150
- copads.objectdistance.Cosine (*function*), 167
- copads.objectdistance.Dennis (*function*), 162
- copads.objectdistance.Dice (*function*), 152
- copads.objectdistance.Euclidean (*function*), 164
- copads.objectdistance.Forbes (*function*), 155
- copads.objectdistance.Fossum (*function*), 159
- copads.objectdistance.Gower\_Legendre (*function*), 163
- copads.objectdistance.Hamann (*function*), 155
- copads.objectdistance.Hamming (*function*), 164
- copads.objectdistance.Jaccard (*function*), 150
- copads.objectdistance.Kulczynski (*function*), 154
- copads.objectdistance.Kulczynski2 (*function*), 154
- copads.objectdistance.Manhattan (*function*), 165
- copads.objectdistance.Matching (*function*), 151
- copads.objectdistance.Mcconnaughey (*function*), 161
- copads.objectdistance.Minkowski (*function*), 165
- copads.objectdistance.Ochiai (*function*), 152
- copads.objectdistance.Ochiai2 (*function*), 153
- copads.objectdistance.Pearson (*function*), 162
- copads.objectdistance.Roger\_Tanimoto (*function*), 157
- copads.objectdistance.Russel\_Rao (*function*), 156
- copads.objectdistance.Simpson (*function*), 156
- copads.objectdistance.Sokal\_Michener (*function*), 151
- copads.objectdistance.Sokal\_Sneath (*function*), 157
- copads.objectdistance.Sokal\_Sneath2 (*function*), 158
- copads.objectdistance.Sokal\_Sneath3 (*function*), 158
- copads.objectdistance.Stiles (*function*), 161
- copads.objectdistance.Tanimoto (*function*), 167
- copads.objectdistance.Tulloss (*function*), 163
- copads.objectdistance.YuleQ (*function*), 160
- copads.objectdistance.YuleY (*function*), 160
- copads.operations (*module*), 169–173
  - copads.operations.asdict (*function*), 169
  - copads.operations.Boolean (*class*), 172–173
  - copads.operations.combination (*function*), 171
  - copads.operations.contents (*function*), 169
  - copads.operations.count (*function*), 169
  - copads.operations.factorial (*function*), 171
  - copads.operations.fcmp (*function*), 170
  - copads.operations.fibonacci (*function*), 171
  - copads.operations.indexesof (*function*), 170
  - copads.operations.intd (*function*), 170
  - copads.operations.itemindex (*function*), 169
  - copads.operations.items (*function*), 169
  - copads.operations.Modulus2 (*class*), 172
  - copads.operations.permutation (*function*), 171
  - copads.operations.product (*function*), 172
  - copads.operations.safe\_exp (*function*), 171
  - copads.operations.safe\_log (*function*), 170



- copads.operations.safe\_log2 (*function*), 170
- copads.operations.sample (*function*), 171
- copads.operations.sample\_wr (*function*), 171
- copads.operations.summation (*function*), 172
- copads.operations.take (*function*), 170
- copads.operations.take\_byfn (*function*), 170
- copads.prioritydictionary (*module*), 174–175
  - copads.prioritydictionary.PriorityDictionary (*class*), 174–175
- copads.ragaraja (*module*), 176–188
  - copads.ragaraja.accumulations (*function*), 178
  - copads.ragaraja.activate\_version (*function*), 187
  - copads.ragaraja.flipping (*function*), 184
  - copads.ragaraja.input\_IO (*function*), 184
  - copads.ragaraja.instruction\_padding (*function*), 176
  - copads.ragaraja.jump\_identifier (*function*), 186
  - copads.ragaraja.logic (*function*), 183
  - copads.ragaraja.loop\_end (*function*), 177
  - copads.ragaraja.loop\_start (*function*), 176
  - copads.ragaraja.mathematics (*function*), 181
  - copads.ragaraja.nBF\_random\_op (*function*), 179
  - copads.ragaraja.nBF\_to\_Ragaraja (*function*), 187
  - copads.ragaraja.not\_used (*function*), 187
  - copads.ragaraja.output\_IO (*function*), 182
  - copads.ragaraja.register\_IO (*function*), 185
  - copads.ragaraja.set\_tape\_value (*function*), 180
  - copads.ragaraja.source\_filter (*function*), 187
  - copads.ragaraja.source\_move (*function*), 180
  - copads.ragaraja.tape\_manipulate (*function*), 185
  - copads.ragaraja.tape\_move (*function*), 177
  - copads.ragaraja.tape\_size (*function*), 179
- copads.register\_machine (*module*), 189–190
  - copads.register\_machine.interpret (*function*), 189
- copads.ring (*module*), 191–192
  - copads.ring.RingList (*class*), 191–192
- copads.samplestatistics (*module*), 193–196
  - copads.samplestatistics.SampleDistribution (*class*), 195
  - copads.samplestatistics.SingleSample (*class*), 193–195
  - copads.samplestatistics.TwoSample (*class*), 195–196
- copads.set (*module*), 197–198
  - copads.set.Set (*class*), 197–198
- copads.statisticsdistribution (*module*), 199–266
  - copads.statisticsdistribution.AntiLogNormalDistribution (*function*), 201
  - copads.statisticsdistribution.BernoulliDistribution (*class*), 238–239
  - copads.statisticsdistribution.BetaDistribution (*class*), 205–207
  - copads.statisticsdistribution.BilateralExponentialDistribution (*function*), 201
  - copads.statisticsdistribution.BinomialDistribution (*class*), 207–209
  - copads.statisticsdistribution.BradfordDistribution (*class*), 239–241
  - copads.statisticsdistribution.BurrDistribution (*class*), 241–243
  - copads.statisticsdistribution.CauchyDistribution (*class*), 209–211
  - copads.statisticsdistribution.ChiDistribution (*class*), 243
  - copads.statisticsdistribution.ChiSquareDistribution (*class*), 218–219
  - copads.statisticsdistribution.CobbDouglasDistribution (*function*), 201
  - copads.statisticsdistribution.CosineDistribution (*class*), 211–213



- copads.statisticsdistribution.Distribution  
     (class), 203–205  
 copads.statisticsdistribution.DoubleExponentialDistribution  
     (function), 201  
 copads.statisticsdistribution.DoubleGammaDistribution  
     (class), 243–245  
 copads.statisticsdistribution.DoubleWeibullDistribution  
     (class), 245  
 copads.statisticsdistribution.ErlangDistribution  
     (function), 201  
 copads.statisticsdistribution.ExponentialDistribution  
     (class), 213–215  
 copads.statisticsdistribution.ExtremeLBDistribution  
     (class), 245–246  
 copads.statisticsdistribution.FDistribution  
     (class), 215–216  
 copads.statisticsdistribution.FisherTippettDistribution  
     (function), 202  
 copads.statisticsdistribution.FiskDistribution  
     (class), 246  
 copads.statisticsdistribution.FoldedNormalDistribution  
     (class), 246–247  
 copads.statisticsdistribution.FrechetDistribution  
     (function), 201  
 copads.statisticsdistribution.FurryDistribution  
     (function), 201  
 copads.statisticsdistribution.GammaDistribution  
     (class), 216–218  
 copads.statisticsdistribution.GenLogisticDistribution  
     (class), 247  
 copads.statisticsdistribution.GeometricDistribution  
     (class), 219–221  
 copads.statisticsdistribution.GompertzDistribution  
     (function), 202  
 copads.statisticsdistribution.GumbelDistribution  
     (class), 247–249  
 copads.statisticsdistribution.HalfNormalDistribution  
     (class), 249–251  
 copads.statisticsdistribution.HyperbolicSecantDistribution  
     (class), 251–253  
 copads.statisticsdistribution.HypergeometricDistribution  
     (class), 221–222  
 copads.statisticsdistribution.LaplaceDistribution  
     (class), 253  
 copads.statisticsdistribution.LogarithmicDistribution  
     (class), 222–224  
 copads.statisticsdistribution.LogisticDistribution  
     (class), 253  
 copads.statisticsdistribution.LogLogisticDistribution  
     (function), 202  
 copads.statisticsdistribution.LogNormalDistribution  
     (class), 253–254  
 copads.statisticsdistribution.LogWeibullDistribution  
     (function), 202  
 copads.statisticsdistribution.LorentzDistribution  
     (function), 202  
 copads.statisticsdistribution.MaxwellDistribution  
     (class), 254–256  
 copads.statisticsdistribution.NakagamiDistribution  
     (class), 256  
 copads.statisticsdistribution.NegativeBinomialDistribution  
     (class), 256–258  
 copads.statisticsdistribution.NegativeExponentialDistribution  
     (function), 202  
 copads.statisticsdistribution.NormalDistribution  
     (class), 224–226  
 copads.statisticsdistribution.ParetoDistribution  
     (class), 258–259  
 copads.statisticsdistribution.PascalDistribution  
     (class), 259–261  
 copads.statisticsdistribution.PoissonDistribution  
     (class), 226–228  
 copads.statisticsdistribution.PolyaDistribution  
     (function), 202  
 copads.statisticsdistribution.PowerFunctionDistribution  
     (class), 261–262  
 copads.statisticsdistribution.RademacherDistribution  
     (class), 262–264  
 copads.statisticsdistribution.RayleighDistribution  
     (class), 264–265  
 copads.statisticsdistribution.ReciprocalDistribution  
     (class), 265–266  
 copads.statisticsdistribution.RectangularDistribution  
     (function), 202  
 copads.statisticsdistribution.SechSquaredDistribution  
     (function), 203  
 copads.statisticsdistribution.SemicircularDistribution  
     (class), 228–230

- copads.statisticsdistribution.TDistribution  
(*class*), 230–232
- copads.statisticsdistribution.TriangularDistribution  
(*class*), 232–234
- copads.statisticsdistribution.UniformDistribution  
(*class*), 234–236
- copads.statisticsdistribution.WaldDistribution  
(*function*), 203
- copads.statisticsdistribution.WeiBullDistribution  
(*class*), 236–238
- copads.tree (*module*), 267–274
  - copads.tree.OrderedBinaryTree (*class*),  
268
  - copads.tree.RBDict (*class*), 272–274
  - copads.tree.RBList (*class*), 270–272
  - copads.tree.RBTree (*class*), 268–270
  - copads.tree.RBTreeIter (*class*), 267–268
- copads.treenodes (*module*), 275–276
  - copads.treenodes.BinaryNode (*class*), 276
  - copads.treenodes.FourNode (*class*), 276
  - copads.treenodes.RBNode (*class*), 275–  
276
  - copads.treenodes.ThreeNode (*class*), 276