

# Library lifeactuary

Pedro Corte Real & Gracinda Rita Guerreiro

July 28, 2022

**Type:** Python Package

**Version:** 0.1

**Authors:** Pedro Corte Real & Gracinda R. Guerreiro

**Contact:** parcr@fct.unl.pt & grg@fct.unl.pt

**License:** MIT License

**Repository:** <https://github.com/parcr/lifeactuary>

## Abstract

*lifeactuary* is a Python library to perform actuarial mathematics on life contingencies and classical financial mathematics computations. Versatile, simple and easy to use. The main functions are implemented using the usual actuarial approach, making it a natural choice for the life actuary.

This document is produced as a descriptive tool on how to use the package and as a user guide for the developed actuarial functions. For each actuarial function, an illustrative example is provided.

The package uses Python version 3.7 or higher.

This package and functions herein are provided as is, without any guarantee regarding the accuracy of calculations. It's distributed using the [MIT License](#) and the authors disclaim any liability arising by any losses due to direct or indirect use of this package.

This package is still under development and further useful and interesting functions will be available any time soon.

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Mortality Tables</b>	<b>6</b>
2.1	Class MortalityTable . . . . .	6
2.2	Importing Mortality Tables . . . . .	6
2.2.1	Reading from lifeactuary Package . . . . .	6
2.2.2	Importing from File . . . . .	7
2.3	Demographic Functions . . . . .	7
2.3.1	$lx[x]$ . . . . .	7
2.3.2	$dx[x]$ . . . . .	7
2.3.3	$qx[x]$ . . . . .	7
2.3.4	$px[x]$ . . . . .	8
2.3.5	$ex[x]$ . . . . .	8
2.3.6	$w$ . . . . .	8
2.4	Survival Probabilities Functions . . . . .	8
2.4.1	$nqx$ . . . . .	9
2.4.2	$npx$ . . . . .	9
2.4.3	$t\_nqx$ . . . . .	10
2.5	Life Expectancy Function . . . . .	10
2.6	Actuarial Tables . . . . .	11
2.6.1	Class CommutationTable . . . . .	11
2.6.2	$v$ . . . . .	12
2.6.3	$Dx$ . . . . .	12
2.6.4	$Nx$ . . . . .	12
2.6.5	$Sx$ . . . . .	12
2.6.6	$Cx$ . . . . .	13
2.6.7	$Mx$ . . . . .	13
2.6.8	$Rx$ . . . . .	13
<b>3</b>	<b>Some lifeactuary Functions and Syntax</b>	<b>14</b>
3.1	Life Annuities . . . . .	14
3.2	Life Insurances . . . . .	15
3.3	Financial Annuities . . . . .	16
<b>4</b>	<b>Life Annuities</b>	<b>17</b>
4.1	Whole Life Annuities . . . . .	17
4.1.1	$ax$ . . . . .	17
4.1.2	$aax$ . . . . .	17
4.1.3	$t\_ax$ . . . . .	18
4.1.4	$t\_aax$ . . . . .	18
4.2	Temporary Life Annuities . . . . .	19
4.2.1	$nax$ . . . . .	19
4.2.2	$naax$ . . . . .	19
4.2.3	$t\_nax$ . . . . .	20

4.2.4	$t_{naax}$	20
4.3	Life Annuities with variable terms	21
4.3.1	$t_{nIax}$	21
4.3.2	$t_{nIaax}$	21
4.3.3	Geometric Life Annuities	22
4.3.4	Present_Value Function	23
<b>5</b>	<b>Pricing Life Insurance</b>	<b>24</b>
5.1	Pure Endowment / Deferred Capital / Expected Present Value	24
5.2	Whole Life Insurance	24
5.2.1	$Ax$	24
5.2.2	$Ax_{\cdot}$	25
5.2.3	$t_{Ax}$	25
5.2.4	$t_{Ax_{\cdot}}$	25
5.3	Temporary Life Insurance	26
5.3.1	$nAx$	26
5.3.2	$nAx_{\cdot}$	26
5.3.3	$t_{nAx}$	27
5.3.4	$t_{nAx_{\cdot}}$	27
5.4	Endowment Insurance	28
5.4.1	$nAEx$	28
5.4.2	$nAEx_{\cdot}$	28
5.4.3	$t_{nAEx}$	28
5.4.4	$t_{AEx_{\cdot}}$	29
5.5	Temporary Life Insurance with variable Capitals	29
5.5.1	$IAx$	29
5.5.2	$nIAx$	30
5.5.3	$nIArx$	30
5.5.4	$nIArx_{\cdot}$	31
<b>6</b>	<b>Financial Annuities</b>	<b>32</b>
6.1	Class Annuities_Certain	32
6.1.1	$im$	32
6.1.2	$vm$	32
6.1.3	$dm$	33
6.2	Constant Terms Financial Annuities	33
6.2.1	$an$	33
6.2.2	$aan$	33
6.3	Variable Terms Financial Annuities	34
6.3.1	$Ian$	34
6.3.2	$Iaan$	35
6.3.3	$Iman$	35
6.3.4	$Gan$	36
6.3.5	$Gaan$	37
6.3.6	$Gman$	37
6.3.7	$Gmaan$	38

<b>7</b>	<b>Examples</b>	<b>39</b>
7.1	Survival Probabilities . . . . .	39
7.2	Life Tables and Life Annuities . . . . .	41
7.3	Life Insurance . . . . .	45

# 1 Introduction

The *lifeactuary* library for Python aims to provide a wide range of actuarial functions for life contingencies.

The names of the functions follow the International Actuarial Notation and are intuitive ( $qx$ ,  $p_x$ ,  $lx$ ,  $an$ ,  $axn$ ,  $nEx$ ,  $Ax$ , ...) and the common parameters are set as usual ( $x$  for actuarial age,  $n$  for term of the contract,  $p$  for term of payments, ...). This aims to provide with an easy to use and “guessable” list of functions.

Using mortality tables, the library provides functions for computing (a) survival probabilities for integer and non-integer ages and terms, (b) life expectancy for both integer and non integer ages and terms, (c) expected present value of life annuities and (d) expected present value of traditional life insurances. All these features allow for the development of simple or more complicated actuarial evaluations and product developments.

This library can be used for academic or professional purposes. The library contains the functions of main traditional products in Life Insurance, allows for an easy computation of actuarial tables but also provides the tools for designing new products, building tariffs, computing reserves.

It incorporates a very generic and simple to use function to compute the Expected Present Value (aka, actuarial present value) what is becoming a very relevant tool in the solvency analysis.

A set of examples is presented in the end of this manual, showing some potential uses of this library in life contingencies products and evaluations.

This package is still under development and further useful functions will be available any time soon. In the next release, the package will include functions that allow the computation of: (1) variance of life annuities, (2) variance of classical life insurances and (3) expected present value of annuities for multiple lives.

## 2 Mortality Tables

The functions developed on this section are related to common actuarial biometric functions, computed upon a mortality table.

### 2.1 Class MortalityTable

---

class *MortalityTable*

This class instantiates a life table. Data can be provided in the form of the  $l_x$ ,  $q_x$  or  $p_x$ . Note that the first value is the first age considered in the table. The life table will be complete, that is, from age 0 to age  $\omega$  (the last age where  $l_x > 0$ ). It includes the computation of common biometric functions:  $l_x$ ,  $p_x$ ,  $q_x$ ,  $d_x$ ,  $e_x$  for integer ages and for non-integers ages, using methods as Uniform Distribution of Death (udd), Constant Force of Mortality (cfm) and Balducci approximation (bal).

---

#### Usage

```
1 MortalityTable(data_type='q', mt=None, perc=100, last_q=1)
```

#### Description

Initializes the MortalityTable class so that we can construct a mortality table with the usual fields.

#### Parameters

<b>data_type</b>	Use 'l' for $l_x$ , 'p' for $p_x$ and 'q' for $q_x$ .
<b>mt</b>	The mortality table, in array format, according to the data_type defined
<b>perc</b>	The percentage of $q_x$ to use, e.g., use 50 for 50%.
<b>last_q</b>	The value for $q_\omega$ .

### 2.2 Importing Mortality Tables

The package includes a wide number of mortality tables and allows for the inclusion of any other mortality tables extracted from [SOA](#), in xml format, or by importing other ones in usual formats, such as xlsx, csv, txt. For instance, in the manual, one of the tables that we will be using is the [TV7377](#) in the xml format supported by the SOA.

#### 2.2.1 Reading from lifeactuary Package

##### Example

```
1 from lifeactuary import mortality_table as mt, read_soa_table_xml as rst
2
3 # reads TV7377 mortality table from SOA table
4 soa = rst.SoaTable('soa_tables/' + 'TV7377' + '.xml')
5
6 # creates mortality table from qx of SOA table
7 tv7377 = mt.MortalityTable(data_type='q', mt=soa.table_qx, perc=100, last_q=1)
```

### 2.2.2 Importing from File

When building a new mortality table to import from a file, please note that the first value of the table corresponds to the first age considered in the table. For instance, if the first value of the table is 20, it means that  $l_x = 0$ , for  $x = 0, \dots, 19$ .

#### Usage

```
1 from lifeactuary import mortality_table as mt
2 import pandas as pd
3
4 # reads manually imported mortality table
5 table_manual_qx = pd.read_excel('soa_tables/' + 'tables_manual' + '.xlsx', sheet_name='qx')
6 table_manual_lx = pd.read_excel('soa_tables/' + 'tables_manual' + '.xlsx', sheet_name='lx')
7
8 # creates mortality table from lx of a xlsx file
9 grf95 = mt.MortalityTable(data_type='q', mt=list(table_manual_qx['GRF95']), perc=80)
10 grm95 = mt.MortalityTable(data_type='l', mt=list(table_manual_lx['GRM95']), perc=80)
```

## 2.3 Demographic Functions

After the mortality table is instantiated, the common demographic functions are available in the package, such as  $l_x$  (expected number of subjects alive at age  $x$ ),  $d_x$  (expected number of subjects that die with age  $x$ ),  $q_x$  (mortality rate at age  $x$ ),  $e_x$  (complete life expectancy at age  $x$ ),  $\omega$  (terminal age of the mortality table):

### 2.3.1 $lx[x]$

Actuarial Notation	$l_x$
Usage	mt.lx[x]
Args	x: age as an integer number
Example	tv7377.lx[50]
Result	94055.99997478718

### 2.3.2 $dx[x]$

Actuarial Notation	$d_x$
Usage	mt.dx[x]
Args	x: age as an integer number
Example	tv7377.dx[50]
Result	353.99999675630613

### 2.3.3 $qx[x]$

Actuarial Notation	$q_x$
Usage	mt.qx[x]
Args	x: age as an integer number
Example	tv7377.qx[50]
Result	0.0037637152

### 2.3.4 px[x]

Actuarial Notation	$p_x$
Usage	mt.qx[x]
Args	x: age as an integer number
Example	tv7377.px[50]
Result	0.9962362848

### 2.3.5 ex[x]

Actuarial Notation	$e_x$
Usage	mt.ex[x]
Args	x: age as an integer number
Example	tv7377.ex[50]
Result	30.07981415164423

### 2.3.6 w

Actuarial Notation	$\omega$
Usage	mt.w
Example	tv7377.w
Result	106

### Other Examples

```
1 ## Consulting information from an object
2
3 # Outputs the information necessary to clone the object
4 tv7377
5
6 # consults the lx of TV7377
7 tv7377.lx
8
9 # Consults the ex of GRF95
10 grf95.ex
11
12 # extracts all methods from the object grm95
13 grm95.__dict__
```

## 2.4 Survival Probabilities Functions

The package also allows for the direct computation of survival probabilities for an aged  $x$  individual. Focusing on the common actuarial probabilities, some functions are available for the computations of the following probabilities:  ${}_nq_x$ ,  ${}_np_x$ ,  ${}_t|{}_nq_x$  for integer and non-integer ages and periods. In fact, in this library, the non-integer ages and periods are just a particular case when using any method.



### 2.4.1 nqx

Actuarial Notation:  ${}_nq_x$

#### Usage

```
1 nqx(x, n=1, method='udd')
```

#### Parameters

<b>x</b>	age at the beginning
<b>n</b>	period
<b>method</b>	For non-integer ages and periods, use 'udd' for <i>Uniform Distribution of Death</i> , 'cfm' for <i>Constant Force of Mortality</i> and 'bal' for <i>Balducci approximation</i>
<b>return</b>	probability of (x) dying before age $x + n$

#### Examples

```
1 # probability that (50) dies before age 52.
2 tv7377.nqx(50, 2) # 0.0078038614928698236
3
4 # probability that an aged 50.5 individual dies before age 53.
5 tv7377.nqx(50.5, 2.5, method='udd') # 0.010321797187509807
6 tv7377.nqx(50.5, 2.5, method='cfm') # 0.010320038151286903
7 tv7377.nqx(50.5, 2.5, method='bal') # 0.010318279111937612
```

### 2.4.2 npx

Actuarial Notation:  ${}_np_x$

#### Usage

```
1 npx(x, n=1, method='udd')
```

#### Parameters

<b>x</b>	age at the beginning
<b>n</b>	period
<b>method</b>	For non-integer ages and periods, use 'udd' for <i>Uniform Distribution of Death</i> , 'cfm' for <i>Constant Force of Mortality</i> and 'bal' for <i>Balducci approximation</i>
<b>return</b>	probability of (x) surviving beyond age $x + n$

#### Examples

```
1 # probability that (80) reaches age 82.
2 tv7377.npx(80, 2) # 0.8563257446904969
3
4 # probability that an aged 80.5 individual reaches age 85.
5 tv7377.npx(80.5, 4.5, method='udd') # 0.3512032870461814
6 tv7377.npx(80.5, 4.5, method='cfm') # 0.3507713780990377
7 tv7377.npx(80.5, 4.5, method='bal') # 0.35033918162679567
```

### 2.4.3 t\_nqx

**Actuarial Notation:**  ${}_t|_np_x$

#### Usage

```
1 t_nqx(x, t=1, n=1, method='udd')
```

#### Parameters

<b>x</b>	age at the beginning
<b>t</b>	deferment period
<b>n</b>	period
<b>method</b>	For non-integer ages and periods, use 'udd' for <i>Uniform Distribution of Death</i> , 'cfm' for <i>Constant Force of Mortality</i> and 'bal' for <i>Balducci approximation</i>
<b>return</b>	probability of ( $x$ ) surviving beyond age $x + t$ and die before age $x + t + n$

#### Examples

```
1 # probability that (80) reaches age 82.
2 tv7377.npx(80, 2) # 0.8563257446904969
3
4 # probability that an aged 80.5 individual reaches age 85.
5 tv7377.npx(80.5, 4.5, 'udd') # 0.3512032870461814
6 tv7377.npx(80.5, 4.5, 'cfm') # 0.3507713780990377
7 tv7377.npx(80.5, 4.5, 'bal') # 0.35033918162679567
```

## 2.5 Life Expectancy Function

The library allows for the computation of Complete Life Expectancy for integer and non-integer ages and periods.

#### Usage

```
1 exn(x, n, method='udd')
```

#### Parameters

<b>x</b>	age at the beginning
<b>n</b>	period
<b>method</b>	For non-integer ages, use 'udd' for <i>Uniform Distribution of Death</i> , 'cfm' for <i>Constant Force of Mortality</i> and 'bal' for <i>Balducci approximation</i>
<b>return</b>	life expectancy for ( $x$ ) over the next $n$ years

#### Examples

```
1 # complete life expectancy for (60) over the next 10 years
2 tv7377.exn(60, 10) # 9.498277332706456
3 tv7377.exn(60, 10, 'cfm') # 9.498146560076156
4 tv7377.exn(60, 10, 'bal') # 9.498015788406414
5
6 # complete life expectancy for (60.1) over the next 10.2 years
7 tv7377.exn(60.1, 10.2) # 9.673511678284852
8 tv7377.exn(60.1, 10.2, 'cfm') # 9.67338786347054
9 tv7377.exn(60.1, 10.2, 'bal') # 9.673264041773342
```

## 2.6 Actuarial Tables

From a given Mortality Table, the library allows for the construction of an Actuarial Table (or Commutation Table) providing and allowing to access the values of the common commutation symbols.

This functions are useful for academic purposes, or to implement “old” life contingency products where commutation symbols were commonly used.

At the moment, actuarial evaluation should use cashflow projections, for which interest rate curves should be considered instead of a fixed one, as in the case of actuarial tables.

Despite this, this library allows for the computation of traditional methods in life insurance.

### 2.6.1 Class CommutationTable

---

class *CommutationTable*

This class instantiates, for a specific mortality table and interest rate, all the usual commutation functions:  $D_x$ ,  $N_x$ ,  $S_x$ ,  $C_x$ ,  $M_x$  and  $R_x$ .

---

#### Usage

```
1 CommutationFunctions(i=None, g=0, data_type='q', mt=None, perc=100, app_cont=False)
```

#### Description

Initializes the CommutationTable class so that we can construct an actuarial table with the usual fields.

#### Parameters

<b>i</b>	Interest Rate, in percentage. For instance, use 5 for 5%.
<b>g</b>	Rate of growing (in percentage), for capitals evolving geometrically. $g > 0$ for increasing capitals and $g < 0$ for decreasing capitals
<b>data_type</b>	Use 'l' for $l_x$ , 'p' for $p_x$ and 'q' for $q_x$ .
<b>mt</b>	The mortality table, in array format, according to the data_type defined
<b>perc</b>	The percentage of $q_x$ to use, e.g., use 50 for 50%.
<b>app_cont</b>	Use True for continuous approach (deaths occur, in average, in the middle of the year) or False for considering death payments are considered in the end of the year.

#### Examples

```
1 from lifeactuary import mortality_table as mt, commutation_table as ct, read_soa_table_xml as  
  rst  
2 import pandas as pd  
3  
4 # reads SOA table  
5 soa = rst.SoaTable('soa_tables/' + 'TV7377' + '.xml')  
6  
7 # creates an actuarial table from qx of SOA table  
8 tv7377_ct = ct.CommutationFunctions(i=2, g=0, data_type='q', mt=soa.table_qx, perc=100,  
  app_cont=False)  
9  
10 # creates an actuarial table from the lx of an Excel file, with death payments to be processed  
    in the moment of death
```

```

11 grm95_ct = ct.CommutationFunctions(i=1.5, g=0, data_type='l', mt=list(table_manual_lx['GRM95',
    ]), perc=100, app_cont=False)

```

With the construction of the commutation table (ct), the class computes methods that are useful when computing actuarial evaluations.

### 2.6.2 v

<b>Actuarial Notation</b>	$v$
<b>Definition</b>	$\frac{1}{1+i}$
<b>Usage</b>	ct.v
<b>Example</b>	tv7377_ct.v
<b>Result</b>	0.9803921568627451

### 2.6.3 Dx

<b>Actuarial Notation</b>	$D_x$
<b>Usage</b>	ct.Dx[x]
<b>Args</b>	x: age as an integer number
<b>Example</b>	tv7377_ct.Dx[50]
<b>Result</b>	34944.42647196618

### 2.6.4 Nx

<b>Actuarial Notation</b>	$N_x$
<b>Usage</b>	ct.Nx[x]
<b>Args</b>	x: age as an integer number
<b>Example</b>	tv7377_ct.Nx[50]
<b>Result</b>	788151.7176774722

### 2.6.5 Sx

<b>Actuarial Notation</b>	$S_x$
<b>Usage</b>	ct.Sx[x]
<b>Args</b>	x: age as an integer number
<b>Example</b>	tv7377_ct.Sx[50]
<b>Result</b>	12024274.4751688

In the following methods, the choice between the payments are made in the “end of the year” or in the “moment of death” must be performed when constructing the commutation table (False or True in the app\_cont parameter, respectively). In that sense, the actuarial notation in each of the following methods is given for both scenarios.

### 2.6.6 Cx

Actuarial Notation	$C_x$ or $\overline{C}_x$
Usage	ct.Cx[x]
Args	x: age as an integer number
Example	tv7377_ct.Cx[50]
Result	128.94202849786421

### 2.6.7 Mx

Actuarial Notation	$M_x$ or $\overline{M}_x$
Usage	ct.Mx[x]
Args	x: age as an integer number
Example	tv7377_ct.Mx[50]
Result	19490.471223388264

### 2.6.8 Rx

Actuarial Notation	$R_x$ or $\overline{R}_x$
Usage	ct.Rx[x]
Args	x: age as an integer number
Example	tv7377_ct.Rx[50]
Result	552381.6299290637

### Examples

```
1 # actuarial present value of a unitary due whole life annuity for (50) paid annually
2 tv7377_ct.Nx[50]/tv7377_ct.Dx[50]      # 22.55443277
3
4 # actuarial present value of a whole life insurance for a (50) with 100.000 m.u. capital.
   Payment is made in the moment of death
5 tv7377_ct_md = ct.CommutationFunctions(2, 0, 'q', soa.table_qx, 100, True)
6 tv7377_ct_md.Mx[50]/tv7377_ct_md.Dx[50]      # 56330.616995
```

### 3 Some lifeactuary Functions and Syntax

#### 3.1 Life Annuities

Table 1: Actuarial Notation and Syntax Formula for Life Annuities

Notation	Description	Syntax
$a_x$	whole life annuity	ax(x,1)
$\ddot{a}_x$	whole life annuity due	aax(x,1)
${}_t a_x$	$t$ years deferred whole life annuity	t_ax(x,1,t)
${}_t \ddot{a}_x$	$t$ years deferred whole life annuity due	t_aax(x,1,t)
$a_x^{(m)}$	whole life annuity payable $m$ times per year	ax(x,m)
$\ddot{a}_x^{(m)}$	whole life annuity due payable $m$ times per year	aax(x,m)
${}_t a_x^{(m)}$	$t$ years deferred whole life annuity payable $m$ times per year	t_ax(x,m,t)
${}_t \ddot{a}_x^{(m)}$	$t$ years deferred whole life annuity due payable $m$ times per year	t_aax(x,m,t)
$a_{x:\overline{n} }$	$n$ year temporary life annuity	nax(x,n,1)
$\ddot{a}_{x:\overline{n} }$	$n$ year temporary life annuity due	naax(x,n,1)
${}_t a_{x:\overline{n} }$	$t$ year deferred $n$ year temporary life annuity	t_nax(x,n,1,t)
${}_t \ddot{a}_{x:\overline{n} }$	$t$ year deferred $n$ year temporary life annuity due	t_naax(x,n,1,t)
$a_{x:\overline{n} }^{(m)}$	$n$ year temporary life annuity payable $m$ times per year	nax(x,n,m)
$\ddot{a}_{x:\overline{n} }^{(m)}$	$n$ year temporary life annuity due payable $m$ times per year	naax(x,n,m)
${}_t a_{x:\overline{n} }^{(m)}$	$t$ year deferred $n$ year temporary life annuity payable $m$ times per year	t_nax(x,n,m,t)
${}_t \ddot{a}_{x:\overline{n} }^{(m)}$	$t$ year deferred $n$ year temporary life annuity due payable $m$ times per year	t_naax(x,n,m,t)

Table 2: Actuarial Notation and Syntax Formula for Increasing Life Annuities

Notation	Description	Syntax
${}_t Ia_{x:\overline{n} }^{(m)r}$	$t$ -years deferred $n$ -year temporary increasing life annuity, payable $m$ times per year. First payment $C$ and increasing/decreasing amount $r$	t_nIax(x,n,m,t,C,r)
${}_t I\ddot{a}_{x:\overline{n} }^{(m)r}$	$t$ -years deferred $n$ -year temporary increasing life annuity, payable $m$ times per year. First payment $C$ and increasing/decreasing amount $r$	t_nIaax(x,n,m,t,C,r)

For life annuities with terms varying geometrically, the Actuarial Table must be built with an increasing rate  $g$  and the functions from Table 1 are applied.

### 3.2 Life Insurances

The following table resumes the available function for life insurances. As usual in the actuarial notation, the capital letters with bar refer to payments due in the moment of death and the absense of bar refers to payments due in the end of the year in which the death occurs.

Table 3: Actuarial Notation and Syntax Formula for Life Insurances - fixed capitals

Notation	Description	Syntax
${}_nE_x$	pure endowment	nEx(x,n)
$A_x$	whole life insurance (end of the year)	Ax(x)
$\bar{A}_x$	whole life insurance (moment of death)	Ax_(x)
${}_t A_x$	$t$ years deferred whole life insurance (end of the year)	t_Ax(x,t)
${}_t \bar{A}_x$	$t$ years deferred whole life insurance (moment of death)	t_Ax_(x,t)
$A^1_{x:\overline{n} }$	term life insurance (end of the year)	nAx(x,n)
$\bar{A}^1_{x:\overline{n} }$	term life insurance (moment of death)	nAx_(x,n)
${}_t A^1_{x:\overline{n} }$	$t$ years deferred term life insurance (end of the year)	t_nAx(x,n,t)
${}_t \bar{A}^1_{x:\overline{n} }$	$t$ years deferred term life insurance (moment of death)	t_nAx_(x,n,t)
$A_{x:\overline{n} }$	endowment insurance (end of the year)	nAEx(x,n)
$\bar{A}_{x:\overline{n} }$	endowment insurance (moment of death)	nAEx_(x,n)
${}_t A_{x:\overline{n} }$	$t$ -years deferred endowment insurance (end of the year)	t_nAEx(x,n,t)
${}_t \bar{A}_{x:\overline{n} }$	$t$ -years deferred endowment insurance (moment of death)	t_nAEx_(x,n,t)

Table 4: Actuarial Notation and Syntax Formula for Life Insurances - variable capitals

Notation	Description	Syntax
$(IA)_x$	whole life insurance with arithmetically increasing capitals (end of the year)	IAx(x)
$(I\bar{A})_x$	whole life insurance with arithmetically increasing capitals (moment of death)	IAx_(x)
$(IA)_{x:\overline{n} }$	term life insurance with arithmetically increasing capitals (end of the year)	nIAx(x,n)
$(I\bar{A})_{x:\overline{n} }$	term life insurance with arithmetically increasing capitals (moment of death)	nIAx_(x,n)
${}_t (IA)^r_{x:\overline{n} }$	$t$ -years deferred term life insurance with capitals evolving arithmetically (increasing or decreasing). First Capital $C$ and increase amount $r$ (end of the year)	nIArx(x,n,t,C,r)
${}_t (I\bar{A})^r_{x:\overline{n} }$	$t$ -years deferred term life insurance with capitals evolving arithmetically (increasing or decreasing). First Capital $C$ and increase amount $r$ (moment of death)	nIArx_(x,n,t,C,r)

### 3.3 Financial Annuities

Table 5: Actuarial Notation and Syntax Formula for Financial Annuities

Notation	Description	Syntax
$a_{\overline{n} }$	$n$ -year immediate financial annuity	an(n)
$\ddot{a}_{\overline{n} }$	$n$ -year due financial annuity	aan(n)
$a_{\infty}$	perpetual immediate financial annuity	a(None)
$\ddot{a}_{\infty}$	perpetual due financial annuity	aa(None)
${}_r(Ia)_{\overline{n} }^{(m)}$	$n$ -year immediate financial annuity with first payment $C$ and evolving arithmetically (increasing [ $r > 0$ ] or decreasing [ $r < 0$ ]). Payment increases in each period of the interest rate.	Ian(n,C,r)
${}_r(I\ddot{a})_{\overline{n} }^{(m)}$	$n$ -year due financial annuity with first payment $C$ and evolving arithmetically (increasing [ $r > 0$ ] or decreasing [ $r < 0$ ]). Payment increases in each period of the interest rate.	Iaan(n,C,r)
${}_r(I^{(m)}a)_{\overline{n} }^{(m)}$	$n$ -year immediate financial annuity with first payment $C$ and evolving arithmetically (increasing [ $r > 0$ ] or decreasing [ $r < 0$ ]). Payments increase in each payment period.	Iman(n,C,r)
${}_r(I^{(m)}\ddot{a})_{\overline{n} }^{(m)}$	$n$ -year due financial annuity with first payment $C$ and evolving arithmetically (increasing [ $r > 0$ ] or decreasing [ $r < 0$ ]). Payments increase in each payment period.	Imaan(n,C,r)
${}_g(Ga)_{\overline{n} }^{(m)}$	$n$ -year immediate financial annuity with first payment $C$ and evolving geometrically with rate $g$ . Payments change in each period of the interest rate.	Gan(n,C,g)
${}_g(G\ddot{a})_{\overline{n} }^{(m)}$	$n$ -year due financial annuity with first payment $C$ and evolving geometrically with rate $g$ . Payments change in each period of the interest rate.	Gaan(n,C,g)
${}_g(G^{(m)}a)_{\overline{n} }^{(m)}$	$n$ -year immediate financial annuity with first payment $C$ and evolving geometrically with rate $g$ . Payments change in each payment period.	Gman(n,C,g)
${}_g(G^{(m)}\ddot{a})_{\overline{n} }^{(m)}$	$n$ -year due financial annuity with first payment $C$ and evolving geometrically with rate $g$ . Payments change in each payment period.	Gmaan(n,C,g)

For annuities paid  $m$  times per year, the class Annuities Certain must be initiated with the correspondent frequency  $m$ . Examples are presented in section 6.



## 4 Life Annuities

A life annuity corresponds to a series of payments paid as long as an individual is alive on the payment date. The life annuity can be temporary or payable for whole life, the payments are due in the beginning (annuity due) or at the end of the periods (annuity immediate) and starts immediately in the next period or after some delay (deferred annuities). The payments are constant through the all term of contract or are variable (with or without a mathematical regularity). The number of payments in each period of the interest rate may also be defined.

The computation of the present value of all these life annuities is available in the library, and are presented in this chapter, for integer ages and terms.

For a more general approach, the library includes a function that computes the present value of a given series of cash-flows, with a given set of interest rates and a defined set of probabilities.

### 4.1 Whole Life Annuities

#### 4.1.1 ax

**Actuarial Notation:**  $a_x$  and  $a_x^{(m)}$

**Usage**

```
1 ax(x, m=1)
```

**Description:** Returns the actuarial present value of a whole life annuity of 1 per time period. Payments of  $1/m$  are made  $m$  times per year at the end of the periods.

**Parameters**

<b>x</b>	age at the beginning of the contract
<b>m</b>	number of payments in each period of the interest rate

**Examples**

```
1 tv7377_ct.ax(50, 1) # 21.554432773700235
2 tv7377_ct.ax(50, 4) # 21.929432773700235
```

#### 4.1.2 aax

**Actuarial Notation:**  $\ddot{a}_x$  and  $\ddot{a}_x^{(m)}$

**Usage**

```
1 aax(x, m=1)
```

**Description:** Returns the actuarial present value of a whole life annuity due of 1 per time period. The payments of  $1/m$  are made  $m$  times per year at the beginning of the periods.

### Parameters

<b>x</b>	age at the beginning of the contract
<b>m</b>	number of payments in each period of the interest rate

### Examples

```
1 tv7377_ct.ax(50, 1) # 22.55443277370024
2 tv7377_ct.ax(50, 4) # 22.17943277370024
```

#### 4.1.3 t\_ax

**Actuarial Notation:**  ${}_t|a_x$  and  ${}_t|a_x^{(m)}$

### Usage

```
1 t_ax(x, m=1, defer=0)
```

**Description:** Returns the actuarial present value of a immediate whole life annuity of 1 per time period, deferred  $t$  periods. The payments of  $1/m$  are made  $m$  times per year at the end of the periods.

### Parameters

<b>x</b>	age at the beginning of the contract
<b>m</b>	number of payments in each period of the interest rate
<b>defer</b>	number of deferment years

**Observation:**  $t\_ax(x, m, defer=0) = ax(x, m)$

### Examples

```
1 tv7377_ct.t_ax(50, 1, 5) # 16.899196591768252
2 tv7377_ct.t_ax(50, 4, 5) # 17.231374204075433
```

#### 4.1.4 t\_aax

**Actuarial Notation:**  ${}_t|\ddot{a}_x$  and  ${}_t|\ddot{a}_x^{(m)}$

### Usage

```
1 t_aax(x, m=1, defer=0)
```

**Description:** Returns the actuarial present value of a whole life annuity due of 1 per time period, deferred  $t$  periods. The payments of  $1/m$  are made  $m$  times per year at the beginning of the periods.

### Parameters

<b>x</b>	age at the beginning of the contract
<b>m</b>	number of payments in each period of the interest rate
<b>defer</b>	number of deferment years

**Observation:**  $t\_aax(x, m, defer=0) = aax(x, m)$

## Examples

```
1 tv7377_ct.t_aax(50, 1, 5) # 17.78500355792074
2 tv7377_ct.t_aax(50, 4, 5) # 17.45282594561355
```

## 4.2 Temporary Life Annuities

### 4.2.1 nax

**Actuarial Notation:**  $a_{x:\overline{n}|}$  and  $a_{x:\overline{n}|}^{(m)}$

#### Usage

```
1 nax(x, n, m=1)
```

**Description:** Returns the actuarial present value of an immediate  $n$  term life annuity of 1 per time period. The payments of  $1/m$  are made  $m$  times per year at the end of the periods.

#### Parameters

<b>x</b>	age at the beginning of the contract
<b>n</b>	number of periods until the end of the contract (measured in periods of the interest rate)
<b>m</b>	number of payments in each period of the interest rate

## Examples

```
1 tv7377_ct.nax(50, 10, 1) # 8.756215803256639
2 tv7377_ct.nax(50, 10, 4) # 8.839775242816884
```

### 4.2.2 naax

**Actuarial Notation:**  $\ddot{a}_{x:\overline{n}|}$  and  $\ddot{a}_{x:\overline{n}|}^{(m)}$

#### Usage

```
1 naax(x, n, m=1)
```

**Description:** Returns the actuarial present value of a  $n$  term life annuity due of 1 per time period. The payments of  $1/m$  are made  $m$  times per year at the beginning of the periods.

#### Parameters

<b>x</b>	age at the beginning of the contract
<b>n</b>	number of periods until the end of the contract (measured in periods of the interest rate)
<b>m</b>	number of payments in each period of the interest rate

## Examples

```
1 tv7377_ct.naax(50, 10, 1) # 8.979040975417291
2 tv7377_ct.naax(50, 10, 4) # 8.895481535857046
```

### 4.2.3 t\_nax

**Actuarial Notation:**  ${}_t|a_{x:\overline{n}}|$  and  ${}_t|a_{x:\overline{n}}^{(m)}$

#### Usage

```
1 t_nax(x, n, m=1, defer=0)
```

**Description:** Returns the actuarial present value of a immediate  $n$  term life annuity of 1 per time period, deferred  $t$  periods. The payments of  $1/m$  are made  $m$  times per year at the end of the periods.

#### Parameters

<b>x</b>	age at the beginning of the contract
<b>n</b>	number of periods until the end of the contract (measured in periods of the interest rate)
<b>m</b>	number of payments in each period of the interest rate
<b>defer</b>	number of deferment years

**Observation:**  $t\_nax(x, m, defer=0) = nax(x, m)$

#### Examples

```
1 tv7377_ct.t_nax(50, 10, 1, 5) # 7.670292001795834
2 tv7377_ct.t_nax(50, 10, 4, 5) # 7.750622055449898
```

### 4.2.4 t\_naax

**Actuarial Notation:**  ${}_t|\ddot{a}_{x:\overline{n}}|$  and  ${}_t|\ddot{a}_{x:\overline{n}}^{(m)}$

#### Usage

```
1 t_naax(x, m=1, defer=0)
```

**Description:** Returns the actuarial present value of a  $n$  term life annuity due of 1 per time period, deferred  $t$  periods. The payments of  $1/m$  are made  $m$  times per year at the beginning of the periods.

#### Parameters

<b>x</b>	age at the beginning of the contract
<b>n</b>	number of periods until the end of the contract (measured in periods of the interest rate)
<b>m</b>	number of payments in each period of the interest rate
<b>defer</b>	number of deferment years

**Observation:**  $t\_naax(x, m, defer=0) = naax(x, m)$

#### Examples

```
1 tv7377_ct.t_naax(50, 10, 1, 5) # 7.8845054782066715
2 tv7377_ct.t_naax(50, 10, 4, 5) # 7.804175424552608
```

### 4.3 Life Annuities with variable terms

In this section, are presented functions that compute the actuarial present value of life annuities whose payments are not constant overtime.

As special regularities, life annuities with terms evolving in arithmetic or geometric progression (increasing or decreasing) are well known and easily computed and this library presents easy to use solutions and functions. As a general case, the library allows for the computation of the actuarial present value for a given set of cash-flows, interest rates and , that can vary without any regularity. The set of parameters should be provided in vector formats.

#### 4.3.1 `t_nIax`

**Actuarial Notation:**  ${}_t|(Ia)_{x:\overline{n}|}$  and  ${}_t|(Ia)_{x:\overline{n}|}^{(m)}$

**Usage**

```
1 t_nIax(x, n, m=1, defer=0, first_amount=1, increase_amount=1)
```

**Description:** Returns the actuarial present value of an immediate  $n$  term life annuity, deferred  $t$  periods, with payments evolving in arithmetic progression. Payments of  $1/m$  are made  $m$  times per year at the end of the periods. First amount and Increase amount may be different. For decreasing life annuities, the Increase Amount should be negative.

**Parameters**

<b>x</b>	age at the beginning of the contract
<b>n</b>	number of periods of the contract (measured in periods of the interest rate)
<b>m</b>	number of payments in each period of the interest rate
<b>defer</b>	number of deferment years
<b>first_amount</b>	amount of the first payment
<b>increase_amount</b>	amount of the increase amount
	increasing life annuities: $\text{increase\_amount} > 0$
	decreasing life annuities: $\text{increase\_amount} < 0$

**Observation:**  $\text{t\_nIax}(x, n, m, \text{defer}, 1, \text{increase\_amount}=0) = \text{t\_nax}(x, n, m, \text{defer})$

**Examples**

```
1 tv7377_ct.t_nIax(x=50, n=10, m=1, defer=0, first_amount=1, increase_amount=1) # 46.33017
2 tv7377_ct.t_nIax(x=50, n=10, m=1, defer=0, first_amount=1, increase_amount=5) # 196.62599
3 tv7377_ct.t_nIax(x=50, n=10, m=1, defer=0, first_amount=100, increase_amount=-5) # 687.75180
```

#### 4.3.2 `t_nIaax`

**Actuarial Notation:**  ${}_t|(I\ddot{a})_{x:\overline{n}|}$  and  ${}_t|(I\ddot{a})_{x:\overline{n}|}^{(m)}$

**Usage**

```
1 t_nIaax(x, n, m=1, defer=0, first_amount=1, increase_amount=1)
```

**Description:** Returns the actuarial present value of a  $n$  term life annuity due, deferred  $t$  periods, with payments evolving in arithmetic progression. The payments are made  $m$  times per year at the beginning of the periods and the payment of each period is divided into  $m$  equal payments. First amount and Increase amount may be different. For decreasing life annuities, the *Increase Amount* should be negative.

#### Parameters

<b>x</b>	age at the beginning of the contract
<b>n</b>	number of periods until the end of the contract (measured in periods of the interest rate)
<b>m</b>	number of payments in each period of the interest rate
<b>defer</b>	number of deferment years
<b>first_amount</b>	amount of the first payment
<b>increase_amount</b>	amount of the increase amount
	increasing life annuities: $\text{increase\_amount} > 0$
	decreasing life annuities: $\text{increase\_amount} < 0$

**Observation:**  $\text{t\_nIaax}(x, n, m, \text{defer}, 1, \text{increase\_amount}=0) = \text{t\_naax}(x, n, m, \text{defer})$

#### Examples

```
1 tv7377_ct.t_nIaax(x=50, n=10, m=1, defer=0, first_amount=1, increase_amount=1) # 47.5374643
2 tv7377_ct.t_nIaax(x=50, n=10, m=1, defer=0, first_amount=1, increase_amount=5) # 201.771158
3 tv7377_ct.t_nIaax(x=50, n=10, m=1, defer=0, first_amount=100, increase_amount=-5) # 705.111980
```

### 4.3.3 Geometric Life Annuities

For life annuities with payments evolving in geometric progression (increasing or decreasing) the growth rate ( $g$ ) should be included when computing the actuarial table (see section 2.6.1) and then use the life annuities functions presented in the previous sections.

#### Examples

```
1 from lifeactuary import commutation_table as ct
2
3 # reads SOA table
4 soa = rst.SoaTable('/soa_tables/' + 'TV7377' + '.xml')
5
6 # creates an actuarial table from qx of SOA table with geometric increase of 5% on payments
7 tv7377_ctg_inc = ct.CommutationFunctions(i=2, g=5, data_type='q', mt=soa.table_qx, perc=100,
8     app_cont=False)
9
10 # creates an actuarial table from qx of SOA table with geometric decrease of 5% on payments
11 tv7377_ctg_dec = ct.CommutationFunctions(i=2, g=-5, data_type='q', mt=soa.table_qx, perc=100,
12     app_cont=False)
13
14 # actuarial present value of a geometrically evolving life annuity for a 50 years old
15     individual, for 10 years period
16 tv7377_ctg_inc.naax(50,10,1) # 11.18091822195998
17 tv7377_ctg_dec.naax(50,10,1) # 7.281682932595854
```

#### 4.3.4 Present\_Value Function

This function generalizes any of the above, returning the present value of a series of cash-flows (introduced in vector mode), where the interest rate for each period may differ as well as the probability assigned to each payment/benefit. According to the defined probabilities, the function returns the present value (for probabilities equal to 1) or the actuarial present value of the series of cash-flows.

##### Usage

```
1 present_value(probs, age, spot_rates, capital)
```

**Description:** This function computes the expected present value of a cash-flow, that can be contingent on some probabilities. The payments are considered at the end of the period.

##### Parameters

<b>probs</b>	vector of probabilities. For using the instantiated actuarial table, introduce probs=None.
<b>age</b>	age at the beginning of the contract
<b>spot_rates</b>	vector of interest rates for the considered time periods
<b>capital</b>	vector of cash-flow amounts

##### Examples

```
1 pv1 = tv7377_ct.present_value(probs=1, age=None,
2     spot_rates=[1.2, 1.4, 1.8, 1.6, 1.9], capital=[100, -25, 120, 300, -50])
3 print('Present Value:', pv1)
4 # 425.750701233034
5
6 pv2 = tv7377_ct.present_value(probs=None, age=35,
7     spot_rates=[1.2, 1.4, 1.8, 1.6, 1.9], capital=[100, -25, 120, 300, -50])
8 print('Present Value:', pv2)
9 # 424.2408517830521
```

## 5 Pricing Life Insurance

When pricing life insurance, there is a need to compute the present value of the benefit payment, for a given mortality table and an interest rate.

This library includes functions that allow for pricing the most common contracts in life insurance, such as Pure Endowment, Whole Life and Temporary Insurances, Endowment Insurance as well as the traditional life insurance with increasing (or decreasing) capitals.

The available functions allow for the pricing of any other type of life insurance, whose actuarial evaluation makes use of the functions available in this library.

### 5.1 Pure Endowment / Deferred Capital / Expected Present Value

**Actuarial Notation:**  ${}_nE_x$

**Usage**

```
1 nEx(x, n)
```

**Description:** Returns the present value of a Pure Endowment of 1 for an aged  $x$  individual, paid at age  $x + n$ .

**Parameters**

<b>x</b>	age at the beginning of the contract
<b>m</b>	years until payment, if ( $x$ ) is alive

**Examples**

```
1 tv7377_ct.nEx(50, 5) # 0.8858069661524854
2 tv7377_ct.nEx(50, 10) # 0.7771748278393479
3 tv7377_ct.nEx(80, 10) # 0.2283081320230277
```

### 5.2 Whole Life Insurance

#### 5.2.1 Ax

**Actuarial Notation:**  $A_x$

**Usage**

```
1 Ax(x)
```

**Description:** Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a whole life insurance (i.e. net single premium), that pays 1 at the end of the year of death.

**Parameters**

<b>x</b>	age at the beginning of the contract
----------	--------------------------------------

**Examples**

```
1 tv7377_ct.Ax(50) # 0.5577562201235239
```



### 5.2.2 Ax\_

**Actuarial Notation:**  $\bar{A}_x$

**Usage**

```
1 Ax_(x)
```

**Description:** Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a whole life insurance (i.e. net single premium), that pays 1 at the moment of death.

**Parameters**

**x** | age at the beginning of the contract

**Examples**

```
1 tv7377_ct.Ax_(50) # 0.5633061699539695
```

### 5.2.3 t\_Ax

**Actuarial Notation:**  ${}_t|A_x$

**Usage**

```
1 t_Ax(x, defer=0)
```

**Description:** Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a whole life insurance (i.e. net single premium), that pays 1 at the end of year of death. The contract is deferred  $t$  years.

**Parameters**

**x** | age at the beginning of the contract  
**t** | deferment period (in years)

**Observation:**  $t\_Ax(x, defer=0) = Ax(x)$

**Examples**

```
1 tv7377_ct.t_Ax(50,2) # 0.550183040772438
```

### 5.2.4 t\_Ax\_

**Actuarial Notation:**  ${}_t|\bar{A}_x$

**Usage**

```
1 t_Ax_(x, defer=0)
```

**Description:** Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a whole life insurance (i.e. net single premium), that pays 1 at the moment of death. The contract is deferred  $t$  years.

### Parameters

<b>x</b>	age at the beginning of the contract
<b>defer</b>	deferment period (in years)

**Observation:**  $t_{Ax\_}(x, \text{defer}=0) = Ax\_ (x)$

### Examples

```
1 tv7377_ct.t_Ax(50,2) # 0.5556576337284301
```

## 5.3 Temporary Life Insurance

### 5.3.1 nAx

**Actuarial Notation:**  $A^1_{x:\overline{n}|}$

### Usage

```
1 nAx(x, n)
```

**Description:** Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a term (temporary) life insurance (i.e. net single premium), that pays 1, at the end of the year of death.

### Parameters

<b>x</b>	age at the beginning of the contract
<b>n</b>	number of years of the contract

### Examples

```
1 tv7377_ct.nAx(50,10) # 0.046765545191685375
```

### 5.3.2 nAx\_

**Actuarial Notation:**  $\bar{A}^1_{x:\overline{n}|}$

### Usage

```
1 nAx_(x, n)
```

**Description:** Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a term (temporary) life insurance (i.e. net single premium), that pays 1, at the moment of death.

### Parameters

<b>x</b>	age at the beginning of the contract
<b>n</b>	number of years of the contract

### Examples

```
1 tv7377_ct.t_Ax_(50,10) # 0.047230885460862126
```

### 5.3.3 t\_nAx

**Actuarial Notation:**  ${}_t|A_{x:\overline{n}}^1$

#### Usage

```
1 t_nAx(x, n, defer=0)
```

**Description:** Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a term (temporary) deferred life insurance (i.e. net single premium), that pays 1, at the end of the year of death.

#### Parameters

<b>x</b>	age at the beginning of the contract
<b>n</b>	number of years of the contract
<b>defer</b>	number of years of deferment

**Observation:**  $t\_nAx(x, n, defer=0) = nAx(x, n)$

#### Examples

```
1 tv7377_ct.t_nAx(50,10,5) # 0.059615329779334834
```

### 5.3.4 t\_nAx\_

**Actuarial Notation:**  ${}_t|\bar{A}_{x:\overline{n}}^1$

#### Usage

```
1 t_nAx_(x, n, defer=0)
```

**Description:** Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a term (temporary) deferred life insurance (i.e. net single premium), that pays 1, at the moment of death.

#### Parameters

<b>x</b>	age at the beginning of the contract
<b>n</b>	number of years of the contract
<b>defer</b>	number of years of deferment

**Observation:**  $t\_nAx\_ (x, n, defer=0) = nAx\_ (x, n)$

#### Examples

```
1 tv7377_ct.t_nAx_(50,10,5) # 0.060208531750847685
```

## 5.4 Endowment Insurance

### 5.4.1 nAEx

**Actuarial Notation:**  $A_{x:\overline{n}|}$

**Usage**

```
1 nAEx(x, n)
```

**Description:** Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of an Endowment life insurance (i.e. net single premium), that pays 1, at the end of year of death or 1 if  $(x)$  survives to age  $x + n$ .

**Parameters**

<b>x</b>	age at the beginning of the contract
<b>n</b>	number of years of the contract

**Examples**

```
1 tv7377_ct.nAEx(50,10) # 0.8239403730310333
```

### 5.4.2 nAEx\_

**Actuarial Notation:**  $\bar{A}_{x:\overline{n}|}$

**Usage**

```
1 nAEx_(x, n)
```

**Description:** Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of an Endowment life insurance (i.e. net single premium), that pays 1, at the moment of death or 1 if  $(x)$  survives to age  $x + n$ .

**Parameters**

<b>x</b>	age at the beginning of the contract
<b>n</b>	number of years of the contract

**Examples**

```
1 tv7377_ct.nAEx_(50,10) # 0.8244057133002101
```

### 5.4.3 t\_nAEx

**Actuarial Notation:**  ${}_t|A_{x:\overline{n}|}$

**Usage**

```
1 t_nAEx(x, n, defer=0)
```

**Description:** Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a deferred Endowment life insurance (i.e. net single premium) that pays 1, at the end of year of death or 1 if ( $x$ ) survives to age  $x + t + n$ .

#### Parameters

<b>x</b>	age at the beginning of the contract
<b>n</b>	number of years of the contract
<b>defer</b>	number of years of deferment

#### Examples

```
1 tv7377_ct.t_nAEx(50,10,2) # 0.7863040688470341
```

#### 5.4.4 t\_AEx\_

**Actuarial Notation:**  ${}_t|\bar{A}_{x:\overline{n}|}$

#### Usage

```
1 t_nAEx_(x, n, defer=0)
```

**Description:** Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a deferred Endowment life insurance (i.e. net single premium) that pays 1, at the moment of death or 1 if ( $x$ ) survives to age  $x + t + n$ .

#### Parameters

<b>x</b>	age at the beginning of the contract
<b>n</b>	number of years of the contract
<b>defer</b>	number of years of deferment

#### Examples

```
1 tv7377_ct.t_nAEx_(50,10,2) # 0.7868146552887256
```

### 5.5 Temporary Life Insurance with variable Capitals

#### 5.5.1 IAx

**Actuarial Notation:**  $(IA)_x$

#### Usage

```
1 IAx(x)
```

**Description:** Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a Term Life Insurance (i.e. net single premium), that pays  $1 + k$ , at the end of year of death, if death occurs between ages  $x + k$  and  $x + k + 1$ , for  $k=0, 1, \dots$ . The capital of the first year is equal to the rate of the progression.

## Parameters

**x** | age at the beginning of the contract

## Examples

```
1 tv7377_ct.IAx(50) # 15.807431562003355
```

### 5.5.2 nIAx

**Actuarial Notation:**  $(IA)_{x:\overline{n}|}$

## Usage

```
1 nIAx(x,n)
```

**Description:** Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of an arithmetically increasing life insurance (i.e. net single premium), that pays  $1 + k$ , at the end of the year if death happens between age  $x + k$  and  $x + k + 1$ ,  $k=0, \dots, n - 1$ .

## Parameters

**x** | age at the beginning of the contract  
**n** | number of years of the contract

## Examples

```
1 tv7377_ct.nIAx(50,10) # 0.2751855520152587
```

### 5.5.3 nIArx

This function computes the actuarial present value of a term insurance whose capitals increase arithmetically, allowing for different amounts of initial capital and increase amount. It corresponds to a generalization of function `nIAx`, where the first amount is equal to increase amount.

**Actuarial Notation:**  ${}_r(IA)_{x:\overline{n}|}$

## Usage

```
1 nIArx_ct(x,n, defer=0, first_amount=1, increase_amount=1)
```

**Description:** Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a term life insurance (i.e. net single premium), that pays  $(\text{first\_amount} + k \times \text{increase\_amount})$ , at the end of the year if death occurs between ages  $x + k$  and  $x + k + 1$ , for  $k = 0, \dots, n - 1$ . Allows the computation for decreasing capitals. The first capital may differ from the increasing/decreasing amount.

## Parameters

<b>x</b>	age at the beginning of the contract
<b>n</b>	number of years of the contract
<b>defer</b>	number of deferment years
<b>first_amount</b>	insured amount in the first year of the contract
<b>increase_amount</b>	rate of increasing (if $> 0$ ) or decreasing (if $< 0$ )

## Examples

```

1 tv7377_ct.nIArx(50,10,0,1000,50) # 58.18654553286392
2 tv7377_ct.nIArx(50,10,10,1000,50) # 133.3402470815534
3 tv7377_ct.nIArx(50,10,0,1000,-50) # 35.344544850506836
4 tv7377_ct.nIArx(50,10,10,1000,-50) # 28.027981923486493

```

**Observation:**  $nIAx(x, n) = nIArx(x, n, \text{defer}=0, \text{first\_amount}=1, \text{increase\_amount}=1)$

### 5.5.4 nIArx\_

This function computes the actuarial present value of a term insurance whose capitals increase arithmetically, allowing for different amounts of initial capital and increase amount. It corresponds to a generalization of function `nIAx_`, where the first amount is equal to increase amount.

**Actuarial Notation:**  $(I\bar{A})_{x:\overline{n}|}^r$

## Usage

```

1 nIArx_(x,n, defer=0, first_amount=1, increase_amount=1)

```

**Description:** Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a term life insurance (i.e. net single premium), that pays  $(\text{first\_amount} + k \times \text{increase\_amount})$ , at the moment of death if death happens between age  $x + k$  and  $x + k + 1$ , for  $k = 0, \dots, n - 1$ .

## Parameters

<b>x</b>	age at the beginning of the contract
<b>n</b>	number of years of the contract
<b>defer</b>	number of deferment years
<b>first_amount</b>	insured amount in the first year of the contract
<b>increase_amount</b>	rate of increasing (if $> 0$ ) or decreasing (if $< 0$ )

## Examples

```

1 tv7377_ct.nIArx_(50,10,0,1000,50) # 58.765530395538875
2 tv7377_ct.nIArx_(50,10,10,1000,50) # 134.66704838825683
3 tv7377_ct.nIArx_(50,10,0,1000,-50) # 35.69624052618538
4 tv7377_ct.nIArx_(50,10,10,1000,-50) # 28.306874184857506

```

## 6 Financial Annuities

### 6.1 Class Annuities\_Certain

---

class *Annuities\_Certain*

This class instantiates the methods for the computation of financial annuities, for a given interest rate and a chosen frequency of payments  $m$  (in each period of the interest rate).

---

#### Usage

```
1 AnnuitiesCertain(interest_rate, m=1)
```

#### Description

Initializes the AnnuitiesCertain class so that we can compute the present value of financial annuities.

#### Parameters

<b>interest_rate</b>	interest rate, in percentage (e.g. use 5 for 5%)
<b>m</b>	frequency of payments, in each period of the interest rate

#### Examples

```
1 from lifeactuary import annuities_certain as ac
2
3 # instantiates a methods for computing financial annuities with a 5% annual interest rate,
  # with annual payments
4 i1=ac.Annuities_Certain(5,1)
5
6 # instantiates a methods for computing financial annuities with a 5% annual interest rate,
  # with quarterly payments
7 i4=ac.Annuities_Certain(5,4)
```

The following methods are available after instantiating the class:

#### 6.1.1 im

<b>Actuarial Notation</b>	$i_m$
<b>Definition</b>	$m [(1 + i)^{1/m} - 1]$
<b>Usage</b>	irate.im
<b>Example</b>	i4.im
<b>Result</b>	0.04908893771615741

#### 6.1.2 vm

<b>Actuarial Notation</b>	$v_m$
<b>Definition</b>	$\frac{1}{1 + \frac{i_m}{m}}$
<b>Usage</b>	irate.vm
<b>Example</b>	i4.vm
<b>Result</b>	0.9878765474230741



### 6.1.3 dm

Actuarial Notation	$d_m$
Definition	$\left(1 - \frac{i}{1+i}\right)^{1/m} - 1$
Usage	irate.dm
Example	i4.dm
Result	0.0484938103077039

## 6.2 Constant Terms Financial Annuities

### 6.2.1 an

**Actuarial Notation:**  $a_{\overline{n}|}$  and  $a_{\overline{n}|}^{(m)}$  or  $a_{\infty|}$  and  $a_{\infty|}^{(m)}$

**Usage**

```
1 an(terms)
```

**Description:** Returns the present value of an immediate  $n$  term financial annuity with payments equal to 1. Payments are made in the end of the periods. In fractional annuities, payments of  $1/m$  are made  $m$  times per year at the end of the periods.

**Parameters**

<b>terms</b>	number of periods (measured in periods of the interest rate) (terms=None) or (terms=0) returns the present value of the perpetual annuity
--------------	--

**Examples**

```
1 i1.an(10) # 7.721734929184813
2 i4.an(10) # 7.86504586209782
3
4 i1.an(None) # 19.999999999999982
5 i4.an(0) # 20.371188429095998
```

### 6.2.2 aan

**Actuarial Notation:**  $\ddot{a}_{\overline{n}|}$  and  $\ddot{a}_{\overline{n}|}^{(m)}$  or  $\ddot{a}_{\infty|}$  and  $\ddot{a}_{\infty|}^{(m)}$

**Usage**

```
1 aan(terms)
```

**Description:** Returns the present value of a due  $n$  term financial annuity with payments equal to 1. Payments are made in the beginning of periods. In fractional annuities, payments of  $1/m$  are made  $m$  times per year at the end of the periods.

**Parameters**

<b>terms</b>	number of periods (measured in periods of the interest rate) (terms=None) or (terms=0) returns the present value of the perpetual annuity
--------------	--

## Examples

```
1 i1.aan(10) # 8.107821675644054
2 i4.aan(10) # 7.9615675487126305
3
4 i1.aan(None) # 20.999999999999982
5 i4.aan(0) # 20.621188429095998
```

## 6.3 Variable Terms Financial Annuities

In this section we present functions that allow the computation of the present value of financial annuities with variable terms, for the particular cases where the terms evolve in arithmetic or geometric progressions.

For both cases, functions are defined in a general way such that the first term may differ from the rate of growth and the same function allows for increasing and decreasing terms.

For annuities with payments more frequent than the interest rate period, two approaches are considered and corresponding functions are developed: (1) payments level within each interest period and increase/decrease from one interest period to the next and (2) payments increase in each payment period.

### Annuities with terms evolving Arithmetically

#### 6.3.1 Ian

**Actuarial Notation:**  $(Ia)_{\overline{n}|}$  and  $(Ia)_{\overline{n}|}^{(m)}$  or  $(Da)_{\overline{n}|}$  and  $(Da)_{\overline{n}|}^{(m)}$

#### Usage

```
1 Ian(terms, payment=1, increase=1)
```

**Description:** Returns the present value of an immediate  $n$  term financial annuity with payments increasing/decreasing arithmetically. Payments are made in the end of the periods. First payment and increase amount may differ. In fractional annuities, payments level within each interest period and increase/decrease from one interest period to the next.

#### Parameters

<b>terms</b>	number of periods (measured in periods of the interest rate)
<b>payment</b>	amount of the first payment
<b>increase</b>	increase amount of payments ( $> 0$ for increasing annuities and $< 0$ for decreasing annuities)

#### Actuarial Formula

For  $C$  - first payment,  $r$  - rate of increasing/decreasing,  $i$ -annual interest rate,  $n$  - number of terms,  $m$ -frequency of payments

$${}_{(C,r)}(Ia)_{\overline{n}|}^{(m)} = C a_{\overline{n}|}^{(m)} + r \frac{a_{\overline{n}|}i - nv^n}{i^{(m)}}$$

## Examples

```
1 a4 = ac.Annuities_Certain(interest_rate=5, m=12)
2 r4 = a4.Ian(terms=20, payment=2000 * 12, increase=400 * 12)
3 print(r4)
4 #789369.5624059099
```

### 6.3.2 Iaan

**Actuarial Notation:**  $I\ddot{a}_{\overline{n}|}$  and  $I\ddot{a}_{\overline{n}|}^{(m)}$  or  $D\ddot{a}_{\overline{n}|}$  and  $D\ddot{a}_{\overline{n}|}^{(m)}$

#### Usage

```
1 Iaan(terms, payment=1, increase=1)
```

**Description:** Returns the present value of a due  $n$  term financial annuity with payments increasing/decreasing arithmetically. Payments are made in the beginning of the periods. First payment and increase amount may differ. In fractional annuities, payments level within each interest period and increase/decrease from one interest period to the next.

#### Parameters

<b>terms</b>	number of periods (measured in periods of the interest rate)
<b>payment</b>	amount of the first payment
<b>increase</b>	increase amount of payments ( $> 0$ for increasing annuities and $< 0$ for decreasing annuities)

## Examples

```
1 a5 = ac.Annuities_Certain(interest_rate=2, m=2)
2 r5 = a5.Iaan(terms=2, payment=1, increase=1)
3 print(r5)
4 # 2.946198813622495
```

### 6.3.3 Iman

**Actuarial Notation:**  $(I^{(m)}a)_{\overline{n}|}$  and  $(I^{(m)}a)_{\overline{n}|}^{(m)}$  or  $(D^{(m)}a)_{\overline{n}|}$  and  $(D^{(m)}a)_{\overline{n}|}^{(m)}$

#### Usage

```
1 Iman(terms, payment=1, increase=1)
```

**Description:** Returns the present value of an immediate  $n$ -term financial annuity with payments increasing/decreasing arithmetically. Payments are made in the end of the periods. First payment and increase amount may differ. In fractional annuities, payments increase in each payment period.

## Parameters

<b>terms</b>	number of periods (measured in periods of the interest rate)
<b>payment</b>	amount of the first payment
<b>increase</b>	increase amount of payments ( $> 0$ for increasing annuities and $< 0$ for decreasing annuities)

## Actuarial Formula

For  $C$  - first payment,  $r$  - rate of increasing/decreasing,  $i$ -annual interest rate,  $n$  - number of terms,  $m$ -frequency of payments

$${}_{(C,r)}(I^{(m)}a)_{\overline{n}|i}^{(m)} = C a_{\overline{n}|i}^{(m)} + rm \frac{a_{\overline{n}|i}^{(m)} - nv^n}{i^{(m)}}$$

## Examples

```
1 a3 = ac.Anuities_Certain(interest_rate=3.3, m=12)
2 r3 = a3.Iman(terms=8, payment=25 * 12, increase=2 * 12)
3 print(r3)
4 # 9781.284321297218
```

## Annuities with terms evolving Geometrically

For computing the present value of geometric financial annuities, let us consider the interest rate  $i_g$  which reflects the annual interest rate of the annuity,  $i$ , and the growth rate  $g$ . For these cases, let us define

$$i_g = \frac{1-g}{1+i} \quad \text{and} \quad v_g = \frac{1+g}{1+i}.$$

### 6.3.4 Gan

**Actuarial Notation:**  ${}_g(Ga)_{\overline{n}|}$  and  ${}_g(Ga)_{\overline{n}|}^{(m)}$

## Usage

```
1 Gan(terms, payment=1, grow=0)
```

**Description:** Returns the present value of an immediate  $n$  term financial annuity with payments increasing/decreasing geometrically. Payments are made in the end of the periods. In fractional annuities, payments level within each interest period and increase/decrease from one interest period to the next.

## Parameters

<b>terms</b>	number of periods (measured in periods of the interest rate)
<b>payment</b>	amount of the first payment
<b>grow</b>	rate of growing of payments, in percentage

### Actuarial Formula

For  $C$  - first payment ,  $g$  - rate of increasing/decreasing,  $i$ -annual interest rate,  $n$  - number of terms,  $m$ -frequency of payments

$${}_{(C,g)}(Ga)_{\overline{n}|}^{(m)} = \begin{cases} \frac{C}{m} \frac{1}{(1+g)^{1/m}} a_{\overline{n}|i_g}^{(m)} & , \quad i \neq g \\ Cnv^{1/m} & , \quad i = g \end{cases}$$

### Examples

```

1 g1=ac.Annuities_Certain(interest_rate=5, m=2)
2 g1.Gan(terms=5,payment=10,grow=10)
3 # 108.60048398210647

```

### 6.3.5 Gaan

**Actuarial Notation:**  ${}_g(G\ddot{a})_{\overline{n}|}^{(m)}$

#### Usage

```

1 Gaan(terms,payment=1, grow=0)

```

**Description:** Returns the present value of an immediate  $n$  term financial annuity with payments increasing/decreasing geometrically. Payments are made in the end of the periods. In fractional annuities, payments level within each interest period and increase/decrease from one interest period to the next.

#### Parameters

<b>terms</b>	number of periods (measured in periods of the interest rate)
<b>payment</b>	amount of the first payment
<b>grow</b>	rate of growing of payments, in percentage

### Examples

```

1 g2=ac.Annuities_Certain(interest_rate=5, m=4)
2 g2.Gaan(terms=5,payment=100,grow=10)
3 # 2185.9539086346845

```

### 6.3.6 Gman

**Actuarial Notation:**  ${}_g(G^{(m)}a)_{\overline{n}|}$  or  ${}_g(G^{(m)}a)_{\overline{n}|}^{(m)}$

#### Usage

```

1 Gman(terms,payment=1, grow=0)

```

**Description:** Returns the present value of an immediate  $n$  term financial annuity with payments increasing/decreasing geometrically. Payments are made in the end of the periods. In fractional annuities, payments increase in each payment period.

#### Parameters

<b>terms</b>	number of periods (measured in periods of the interest rate)
<b>payment</b>	amount of the first payment
<b>grow</b>	rate of growing of payments, in percentage

#### Actuarial Formula

For  $C$  - first payment,  $g$  - rate of increasing/decreasing,  $i$ -annual interest rate,  $n$  - number of terms,  $m$ -frequency of payments

$${}_{(C,g)}(Ga)_{\overline{n}|i}^{(m)} = \begin{cases} Ca_{\overline{n}|i}^{(m)} \times \ddot{a}_{\overline{n}|i_g} & , \quad i \neq g \\ C n a_{\overline{n}|i}^{(m)} & , \quad i = g \end{cases}$$

#### Examples

```
1 g3=ac.Annuities_Certain(interest_rate=5, m=2)
2 g3.Gman(terms=5,payment=10,grow=10)
3 # 53.022051853437205
```

### 6.3.7 Gmaan

**Actuarial Notation:**  ${}_g(G^{(m)}\ddot{a})_{\overline{n}|}$  or  ${}_g(G^{(m)}\ddot{a})_{\overline{n}|}$

#### Usage

```
1 Gmaan(terms,payment=1, grow=0)
```

**Description:** Returns the present value of an immediate  $n$  term financial annuity with payments increasing/decreasing geometrically. Payments are made in the beginning of the periods. In fractional annuities, payments increase in each payment period.

#### Parameters

<b>terms</b>	number of periods (measured in periods of the interest rate)
<b>payment</b>	amount of the first payment
<b>grow</b>	rate of growing of payments, in percentage

#### Examples

```
1 g4=ac.Annuities_Certain(interest_rate=5, m=2)
2 g4.Gmaan(terms=5,payment=10,grow=10)
3 # 51.80299115517991
```

## 7 Examples

In this section we present some interesting and more complete examples of application, for which the use of lifeactuary package is helpful and provides “easy to use” solutions and functions.

### 7.1 Survival Probabilities

---

#### Example 1

Consider a 50 year old individual and the TV7377 mortality table.

1. Determine the probabilities of (50) being alive in the end of each month of the following ten years, considering the Uniform Distribution of Death approximation.
2. Determine the probabilities of (50) not surviving up to the end of each month of the following ten years, considering the Uniform Distribution of Death approximation.
3. Build a dataframe with ages and estimated probabilities.
4. Export data to an Excel file.
5. Plot the estimated probabilities in a scatterplot.

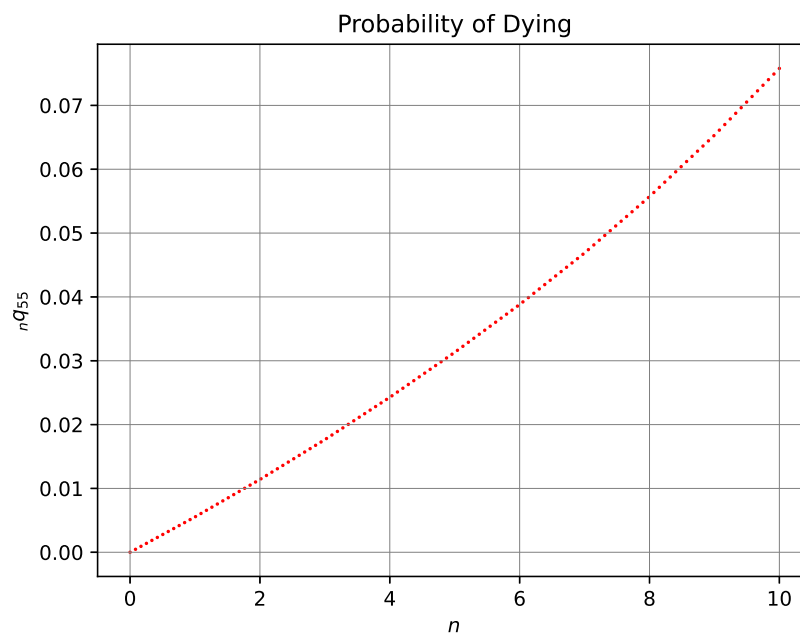
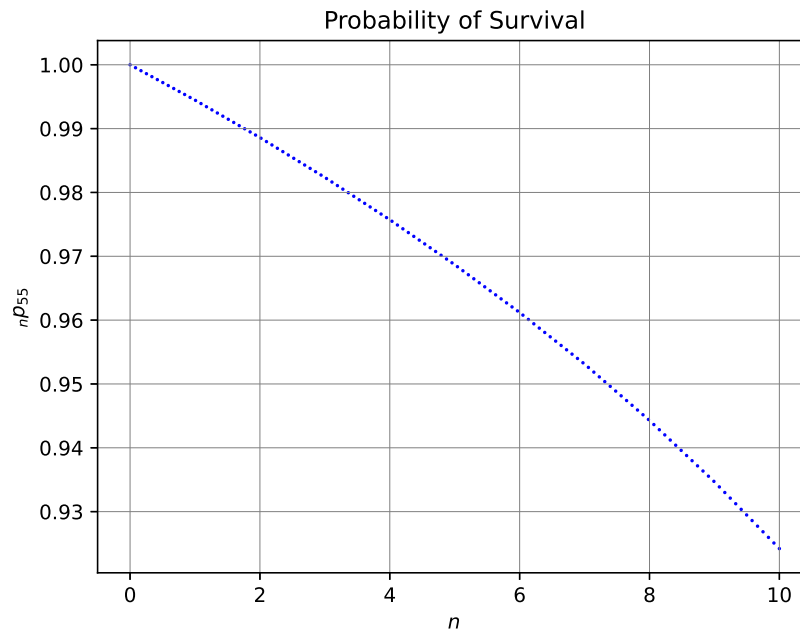
---

```
1 from lifeactuary import mortality_table as mtable, read_soa_table_xml as rst
2
3 import numpy as np
4 import pandas as pd
5 import matplotlib.pyplot as plt
6
7 mt = rst.SoaTable('/soa_tables/TV7377' + '.xml')
8 lt = mtable.MortalityTable(mt=mt.table_qx)
9
10 # Question 1
11 x = 55
12 n = np.linspace(0, 10, 10*12)
13 sprobs = [lt.npx(x=x, n=i, method='udd') for i in n]
14 dprobs = [lt.nqx(x=x, n=i, method='udd') for i in n]
15 ages = x + n
16 df = pd.DataFrame.from_dict({'n': n, 'x': ages, 'npx': sprobs, 'nqx': dprobs})
17
18 # Question 2
19 df.to_excel(excel_writer='example1.xlsx', sheet_name='example1', index=False, freeze_panes=(1,
20 1))
21
22 # Question 3
23 plt.scatter(n, sprobs, s=.5, color='blue')
24
25 plt.xlabel(r'$n$')
26 plt.ylabel(r'${}_n p_{55}$')
27 plt.title('Probability of Survival')
28 plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='--', linewidth=.1)
29 plt.savefig('example1s' + '.eps', format='eps', dpi=3600)
30 plt.show()
```

```

31 plt.scatter(n, dprobs, s=.5, color='red')
32
33 plt.xlabel(r'$n$')
34 plt.ylabel(r'$\{ \}_{n}q_{55}$')
35 plt.title('Probability of Dying')
36 plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='-', linewidth=.1)
37 plt.savefig('example1d' + '.eps', format='eps', dpi=3600)
38 plt.show()

```





## 7.2 Life Tables and Life Annuities

---

### Example 2

In this example, let us develop the Python code for answering the following questions:

1. Import mortality tables TV7377, GRF95 and GRM95, from SOA Mortality Tables, with the columns  $x$ ,  $l_x$ ,  $p_x$ ,  $q_x$ ,  $e_x$ .
2. Construct an actuarial table considering a technical rate of interest of 4% *per annum*, and append the columns  $D_x$  and  $N_x$  to the tables produced in the previous question.
3. Plot the  $l_x$ ,  $q_x$ ,  $p_x$ ,  $e_x$  and  $\ln(D_x)$ , comparing, in the same graph, the values of each mortality table.
4. Determine the net single premium (risk single premium) of a whole life annuity immediate, if someone 55 years old today, wants to receive 1000€ per year considering that:
  - (a) The contract is paid at single premium.
  - (b) The contract is paid at level premiums during 5 years.
5. Determine the net single premium (risk single premium) of a 10 years temporary due life annuity, if someone 55 years old today, wants to receive 1000 m.u. per year.

---

```
1 from lifeactuary import mortality_table as mtable, commutation_table as ct, read_soa_table_xml
  as rst
2
3 import numpy as np
4 import os
5 import sys
6 import matplotlib.pyplot as plt
7
8 this_py = os.path.split(sys.argv[0])[-1][:-3]
9
10 def parse_table_name(name):
11     return name.replace(' ', '').replace('/', '')
12
13 # Question 1
14 table_names = ['TV7377', 'GRF95', 'GRM95']
15 mt_lst = [rst.SoaTable('.././../soa_tables/' + name + '.xml') for name in table_names]
16 lt_lst = [mtable.MortalityTable(mt=mt.table_qx) for mt in mt_lst]
17
18
19 # Question 2
20 interest_rate = 4
21 ct_lst = [ct.CommutationFunctions(i=interest_rate, g=0, mt=mt.table_qx) for mt in mt_lst]
22
23 for idx, lt in enumerate(lt_lst):
24     name = parse_table_name(mt_lst[idx].name)
25     lt.df_life_table().to_excel(excel_writer=name + '.xlsx', sheet_name=name, index=False,
26                               freeze_panes=(1, 1))
27     ct_lst[idx].df_commutation_table().to_excel(excel_writer=name + '_comm' + '.xlsx',
28                                                sheet_name=name, index=False, freeze_panes=(1, 1))
29
```

```

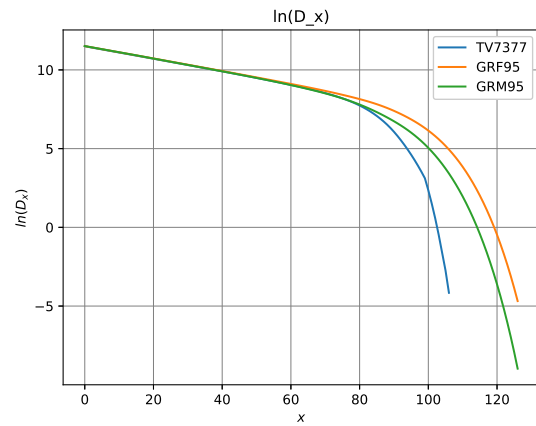
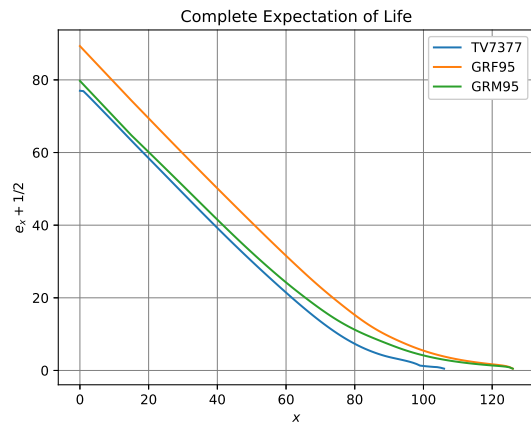
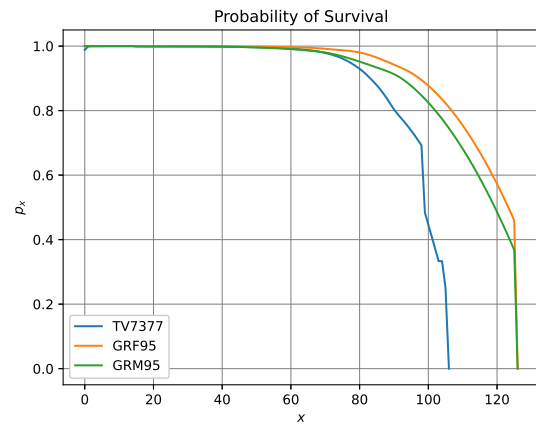
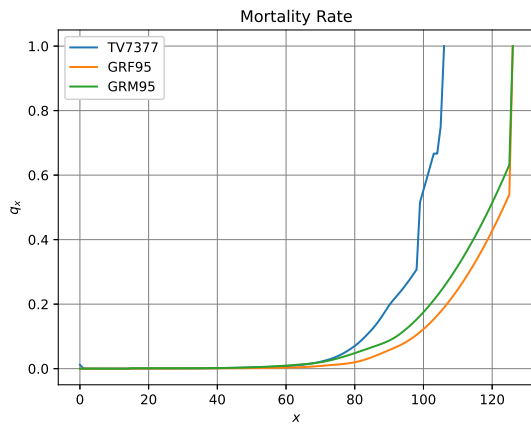
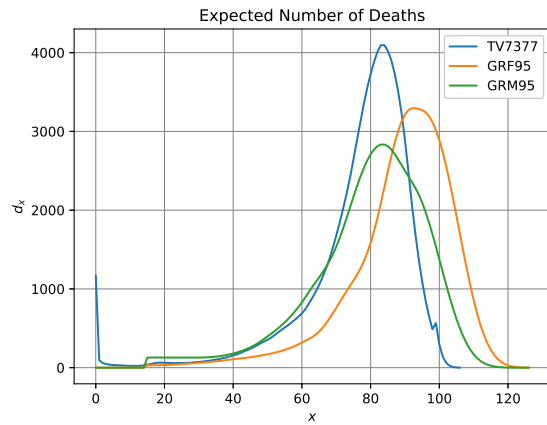
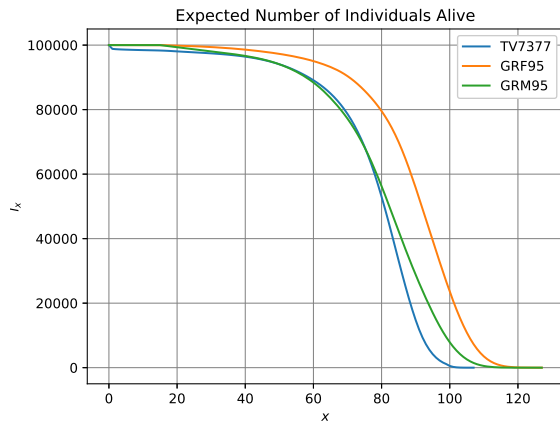
28 # Question 3
29 '''
30 Plot lx
31 '''
32 fig, axes = plt.subplots()
33 for idx, lt in enumerate(lt_lst):
34     ages = np.arange(0, lt.w + 2)
35     plt.plot(ages, lt.lx, label=table_names[idx])
36 plt.xlabel(r'$x$')
37 plt.ylabel(r'$l_x$')
38 plt.title('Expected Number of Individuals Alive')
39 plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='--', linewidth=.1)
40 plt.legend()
41 plt.savefig(this_py + 'lx' + '.eps', format='eps', dpi=3600)
42 plt.show()
43
44 '''
45 Plot dx
46 '''
47 fig, axes = plt.subplots()
48 for idx, lt in enumerate(lt_lst):
49     ages = np.arange(0, lt.w + 1)
50     plt.plot(ages, lt.dx, label=table_names[idx])
51 plt.xlabel(r'$x$')
52 plt.ylabel(r'$d_x$')
53 plt.title('Expected Number of Deaths')
54 plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='--', linewidth=.1)
55 plt.legend()
56 plt.savefig(this_py + 'dx' + '.eps', format='eps', dpi=3600)
57 plt.show()
58
59 '''
60 Plot qx
61 '''
62 fig, axes = plt.subplots()
63 for idx, lt in enumerate(lt_lst):
64     ages = np.arange(0, lt.w + 1)
65     plt.plot(ages, lt.qx, label=table_names[idx])
66 plt.xlabel(r'$x$')
67 plt.ylabel(r'$q_x$')
68 plt.title('Mortality Rate')
69 plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='--', linewidth=.1)
70 plt.legend()
71 plt.savefig(this_py + 'qx' + '.eps', format='eps', dpi=3600)
72 plt.show()
73
74 '''
75 Plot px
76 '''
77 fig, axes = plt.subplots()
78 for idx, lt in enumerate(lt_lst):
79     ages = np.arange(0, lt.w + 1)
80     plt.plot(ages, lt.px, label=table_names[idx])
81 plt.xlabel(r'$x$')
82 plt.ylabel(r'$p_x$')
83 plt.title('Probability of Survival')
84 plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='--', linewidth=.1)

```

```

85 plt.legend()
86 plt.savefig(this_py + 'px' + '.eps', format='eps', dpi=3600)
87 plt.show()
88
89 '''
90 Plot ex
91 '''
92 fig, axes = plt.subplots()
93 for idx, lt in enumerate(lt_lst):
94     ages = np.arange(0, lt.w + 1)
95     plt.plot(ages, lt.ex, label=table_names[idx])
96 plt.xlabel(r'$x$')
97 plt.ylabel(r'$e_{\{x\}+1/2}$')
98 plt.title('Complete Expectation of Life')
99 plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='--', linewidth=.1)
100 plt.legend()
101 plt.savefig(this_py + 'ex' + '.eps', format='eps', dpi=3600)
102 plt.show()
103
104 '''
105 Plot ln(Dx)
106 '''
107 fig, axes = plt.subplots()
108 for idx, lt in enumerate(ct_lst):
109     ages = np.arange(0, lt.w + 1)
110     plt.plot(ages, np.log(lt.Dx), label=table_names[idx])
111 plt.xlabel(r'$x$')
112 plt.ylabel(r'$\ln(D_x)$')
113 plt.title(r'$\ln(D_x)$')
114 plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='--', linewidth=.1)
115 plt.legend()
116 plt.savefig(this_py + 'lnDx' + '.eps', format='eps', dpi=3600)
117 plt.show()
118
119 # Question 4
120 for idx, ct in enumerate(ct_lst):
121     print(table_names[idx] + ": " + f'{round(1000 * ct.ax(x=55, m=1), 2):,}')
122 print()
123
124 # Question 5
125 for idx, ct in enumerate(ct_lst):
126     print(table_names[idx] + ":" + f'{round(1000 * ct.ax(x=55, m=1) / ct.naax(x=55, n=5, m=1), 2):,}')
127
128 # Consult the values used each computation (Nx, Dx)
129 for idx, ct in enumerate(ct_lst):
130     print(ct.msn)
131     print()
132
133 # Consult the values used in the computation, only for TV7377
134 ct_lst[0].msn

```



**Question 4 a):**  $1000a_{55} = 1000 \frac{N_{56}}{D_{55}}$

**TV7377:** 14,979.28 **GRF95:** 18,019.96 **GRM95:** 15,531.28

**Question 4 b):**  $P \ddot{a}_{55:\overline{5}|} = 1000 \frac{N_{56}}{D_{55}}$

**TV7377:** 3,272.3 **GRF95:** 3,910.88 **GRM95:** 3,398.66

**Question 5:**  $1000a_{55:\overline{10}|} = 1000 \frac{N_{56} - N_{66}}{D_{55}}$

**TV7377:** 3,272.3 **GRF95:** 3,910.88 **GRM95:** 3,398.66

## 7.3 Life Insurance

### Example 3

Consider a Pure Endowment insurance with duration 10 years, if someone 55 years old today, subscribes 1000€, considering  $i = 4\%$ /year. Considering the mortality tables TV7377, GRF95 and GRM95:

1. Determine the net single premium (risk single premium).
2. Determine the annual premium paid during 5 years if the insured is alive.
3. Evaluate the price of contract, including the refund of all premiums paid at the end of the year of death and at the end of the term.

```
1 from lifeactuary import mortality_table as mtable, commutation_table as ct, read_soa_table_xml
   as rst
2 import numpy as np
3
4 table_names = ['TV7377', 'GRF95', 'GRM95']
5 interest_rate = 4
6 mt_lst = [rst.SoaTable('soa_tables/' + name + '.xml') for name in table_names]
7 lt_lst = [mtable.MortalityTable(mt=mt.table_qx) for mt in mt_lst]
8 ct_lst = [ct.CommutationFunctions(i=interest_rate, g=0, mt=mt.table_qx) for mt in mt_lst]
9
10 # General Information
11 x = 55
12 capital = 1000
13 term = 10
14 term_annuity = 5
15 # pure endowment
16 pureEndow = [ct.nEx(x=x, n=term) for ct in ct_lst]
17 # temporary annuity due
18 tad = [ct.naax(x=x, n=term_annuity, m=1) for ct in ct_lst]
19
20 ### Question (a)
21 print('\nnet single premium')
22 for idx, ct in enumerate(ct_lst):
23     print(table_names[idx] + ": " + f'{round(capital * pureEndow[idx], 5):,}')
24
25 ### Question (b)
26 print('\nlevel premium')
27 for idx, ct in enumerate(ct_lst):
28     print(table_names[idx] + ": " + f'{round(capital * pureEndow[idx] / tad[idx], 5):,}')
29
30 # show the annuities
31 print('\nannuities')
32 for idx, ct in enumerate(ct_lst):
33     print(table_names[idx] + ": " + f'{round(tad[idx], 5):,}')
34
35 ### Question (c)
36
37 ## Refund of Net Single Premium
38 print('\nSingle Net Risk Premium Refund at End of the Year of Death')
39 termLifeInsurance = [ct.nAx(x=x, n=term) for ct in ct_lst]
40 pureEndow_refund = [ct.nEx(x=x, n=term) / (1 - ct.nAx(x=x, n=term)) for ct in ct_lst]
```

```

41
42 print('\nTerm Life Insurance')
43 for idx, ct in enumerate(ct_lst):
44     print(table_names[idx] + ":" + f'{round(termLifeInsurance[idx], 5):,}')
45
46 print('\nSingle Net Premium Refund Cost at End of the Year of Death')
47 for idx, ct in enumerate(ct_lst):
48     print(table_names[idx] + ":" + f'{round(capital * pureEndow[idx] / (1 - termLifeInsurance[
49         idx]), 5):,}')
50
51 print('Refund Cost at End of the Year of Death')
52 for idx, ct in enumerate(ct_lst):
53     print(table_names[idx] + ":" + f'{round(capital * (pureEndow_refund[idx] - pureEndow[idx]
54         , 5):,}')
55
56 print('\nSingle Net Risk Premium Refund at End of the Term')
57 pureEndow_refund_eot = [ct.nEx(x=x, n=term) / (1 - (1 + interest_rate / 100) ** (-term) + ct.
58     nEx(x=x, n=term))
59     for ct in ct_lst]
60
61 for idx, ct in enumerate(ct_lst):
62     print(table_names[idx] + ":" + f'{round(capital * pureEndow_refund_eot[idx], 5):,}')
63
64 print('Refund Cost at End of the the Term')
65 for idx, ct in enumerate(ct_lst):
66     print(table_names[idx] + ":" + f'{round(capital * (pureEndow_refund_eot[idx] - pureEndow[
67         idx]), 5):,}')
68
69 ## Refund of Net Level Premiums
70
71 print('\nLeveled Net Risk Premium Refund at End of the Year of Death')
72 tli_increasing = [ct.nIAx(x=x, n=term_annuity) for ct in ct_lst]
73 tli_deferred = [ct.t_nAx(x=x, n=term - term_annuity, defer=term_annuity) for ct in ct_lst]
74 pureEndow_leveled_refund = [
75     pureEndow[idx_ct] / (tad[idx_ct] - tli_increasing[idx_ct] - term_annuity * tli_deferred[
76         idx_ct])
77     for idx_ct, ct in enumerate(ct_lst)]
78
79 for idx, ct in enumerate(ct_lst):
80     print(table_names[idx] + ":" + f'{round(capital * pureEndow_leveled_refund[idx], 5):,}')

```

## Solutions:

### Question 1:

$$PP = 1000 {}_{10}E_{55}$$

**TV7377:** 624.36092 **GRF95:** 653.67485 **GRM95:** 615.65987

**Question 2:**

$$P\ddot{a}_{55} = 1000 \frac{{}_{10}E_{55}}{\ddot{a}_{55:\overline{10}|}}$$

**TV7377:** 136.39495 **GRF95:** 141.86738 **GRM95:** 134.72276

**Question 3:**

**Single Premium with Refund paid at the end of the Year of Death**

$$P = 1000 \frac{{}_{10}E_{55}}{1 - A_{55:\overline{10}|}^1}$$

**TV7377:** 664.2747 **GRF95:** 670.91998 **GRM95:** 662.20316

**Single Premium with Refund paid at the end of the contract**

$$P = 1000 \frac{{}_{10}E_{55}}{1 - v^{10} {}_{10}q_{55}}$$

**TV7377:** 658.0555 **GRF95:** 668.30356 **GRM95:** 654.89063

**Level Premium with Refund paid at the end of the year of death**

$$P = 1000 \frac{{}_{10}E_{55}}{\ddot{a}_{55:\overline{10}|} - (IA)_{55:\overline{10}|}}$$

**TV7377:** 144.14453 **GRF95:** 145.18891 **GRM95:** 143.8195

---

#### Example 4

For the contract in Example 3.2, estimate the net premium reserves until the end of the contract, export the estimates to an EXcel file and plot the evolution of the reserves in a graph.

---

```
1
2 ## Net premium reserves path
3
4 10 = 1000
5 reserves_dict = {'table': [], 'x': [], 'insurer': [], 'insured': [], 'reserve': []}
6 fund_dict = {'lx': [], 'claim': [], 'premium': [], 'fund': []}
7
8 # Expected reserves value, that is, considering the survivorship of the group
9 expected_reserve_dict = {'insurer_exp': [], 'insured_exp': [], 'reserve_exp': []}
10 ages = range(x, x + term + 1)
11 print('\n\n Net Premium reserves \n\n')
12 for idx_clt, clt in enumerate(ct_lst):
13     premium_unit = pureEndow[idx_clt]
14     premium_capital = capital * premium_unit
15     premium_unit_levelled = premium_unit / tad[idx_clt]
16     premium_levelled = premium_unit_levelled * capital
17     for age in ages:
18         # reserves
19         reserves_dict['table'].append(table_names[idx_clt])
20         reserves_dict['x'].append(age)
21         insurer_liability = clt.nEx(x=age, n=term - (age - x)) * \
22             capital
23         reserves_dict['insurer'].append(insurer_liability)
24         tad2 = clt.naax(x=age, n=term - annuity - (age - x))
25         insured_liability = premium_levelled * tad2
26         reserves_dict['insured'].append(insured_liability)
27         reserve = insurer_liability - insured_liability
28         reserves_dict['reserve'].append(reserve)
29
30     prob_survival = clt.npx(x=x, n=age - x)
31     lx = 10 * prob_survival
32     expected_reserve_dict['insurer_exp'].append(insurer_liability*lx)
33     expected_reserve_dict['insured_exp'].append(insured_liability*lx)
34     expected_reserve_dict['reserve_exp'].append(reserve*prob_survival*lx)
35
36     # fund
37     fund_dict['lx'].append(lx)
38     qx_1 = clt.nqx(x=age, n=1)
39     claim = 0
40     if age == x + term:
41         claim = capital * lx
42     fund_dict['claim'].append(claim)
43     premium = 0
44     if tad2 > 0:
45         premium = premium_levelled*lx
46     fund_dict['premium'].append(premium)
47     if age == x:
48         fund = lx * premium_levelled
49     else:
50         fund = fund_dict['fund'][-1] * (1 + interest_rate / 100) - claim + premium
51     fund_dict['fund'].append(fund)
```



```

52
53 reserves_df = pd.DataFrame(reserves_dict)
54 expected_reserve_df = pd.DataFrame(expected_reserve_dict)
55 fund_df = pd.DataFrame(fund_dict)
56 name = 'pureEndowment_55_1'
57 reserves_df.to_excel(excel_writer=name + '_netReserves' + '.xlsx', sheet_name=name, index=
    False, freeze_panes=(1, 1))
58
59 '''
60 plot the reserves
61 '''
62 for idx_clt, clt in enumerate(ct_lst):
63     plt.plot(ages, reserves_df.loc[reserves_df['table'] == table_names[idx_clt]]['reserve'],
64             label=table_names[idx_clt])
65
66 plt.xlabel(r'$x$')
67 plt.ylabel('Reserves')
68 plt.title('Net Premium Reserves Pure Endowment')
69 plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='-', linewidth=.1)
70 plt.legend()
71 plt.show()
72
73 # save the graph
74 plt.savefig(this_py + '.eps', format='eps', dpi=3600)

```

