PPL Assignment - Question11

API Documentation

April 19, 2017

Contents

1 Package error-3-nameError

1.1 Modules

- **boy** *(Section* **??**, *p.* **??***)*
- **boy'** *(Section 2, p. 3)*
- **couple** *(Section* **??**, *p.* **??***)*
- **couple'** *(Section 3, p. 5)*
- **gift** *(Section* **??**, *p.* **??***)*
- **gift'** *(Section 4, p. 7)*
- **girl** *(Section 5, p. 9)*
- **q11** *(Section 6, p. 12)*
- **utility** *(Section* **??**, *p.* **??***)*
- **utility'** *(Section 7, p. 13)*

1.2 Variables

Name	Description
__package__	**Value:** `None`

2 Module error-3-nameError.boy’

2.1 Variables

Name	Description
__package__	**Value:** `None`

2.2 Class Boy

Known Subclasses: error-3-nameError.boy.Geek, error-3-nameError.boy.Generous, error-3-nameError.boy.Miser

:param boy: a boy object :param name: name attribute of the boy object :param attractiveness: attractiveness of the boy :param budget: Budget the boy has :param intelligence_level: Intelligence level of the boy :param min_attraction_requirement: Attraction Requirement of the boy :param committed: Commitment status of the boy 0-single 1-committed :param happiness: Happiness of the boy

2.2.1 Methods

__init__(*self*, *name*, *attractiveness*, *budget*, *intelligence_level*, *min_attraction_requirement*, *committed*, *happiness*)

commit(*self*)

isCommitted(*self*)

2.3 Class Geek

error-3-nameError.boy.Boy

error-3-nameError.boy.Geek

2.3.1 Methods

__init__(*self*, *name*, *attractiveness*, *budget*, *intelligence_level*, *min_attraction_requirement*, *committed*, *happiness*)

Overrides: error-3-nameError.boy.Boy.__init__

getHappiness(*self*, *girl*, *total_gifts*)

Inherited from error-3-nameError.boy.Boy(Section 7.3)

commit(), isCommitted()

2.4 Class Generous

error-3-nameError.boy.Boy

error-3-nameError.boy.Generous

2.4.1 Methods

__init__(*self*, *name*, *attractiveness*, *budget*, *intelligence_level*, *min_attraction_requirement*, *committed*, *happiness*)

Overrides: error-3-nameError.boy.Boy.__init__

getHappiness(*self*, *girl*, *total_gifts*)

Inherited from error-3-nameError.boy.Boy(Section 7.3)

commit(), isCommitted()

2.5 Class Miser

error-3-nameError.boy.Boy

error-3-nameError.boy.Miser

2.5.1 Methods

__init__(*self*, *name*, *attractiveness*, *budget*, *intelligence_level*, *min_attraction_requirement*, *committed*, *happiness*)

Overrides: error-3-nameError.boy.Boy.__init__

getHappiness(*self*, *girl*, *total_gifts*)

Inherited from error-3-nameError.boy.Boy(Section 7.3)

commit(), isCommitted()

3 Module error-3-nameError.couple'

3.1 Variables

Name	Description
__package__	**Value:** `'error-3-nameError'`

3.2 Class Boy

Known Subclasses: error-3-nameError.boy.Geek, error-3-nameError.boy.Generous, error-3-nameError.boy.Miser

:param boy: a boy object :param name: name attribute of the boy object :param attractiveness: attractiveness of the boy :param budget: Budget the boy has :param intelligence_level: Intelligence level of the boy :param min_attraction_requirement: Attraction Requirement of the boy :param committed: Commitment status of the boy 0-single 1-committed :param happiness: Happiness of the boy

3.2.1 Methods

__init__(*self*, *name*, *attractiveness*, *budget*, *intelligence_level*, *min_attraction_requirement*, *committed*, *happiness*)

commit(*self*)

isCommitted(*self*)

3.3 Class Couple

3.3.1 Methods

__init__(*self*, *boy*, *girl*, *happiness*, *compatibility*)

3.4 Class Girl

Known Subclasses: error-3-nameError.girl.Choosy, error-3-nameError.girl.Desperate, error-3-nameError.girl.Normal

:param name: name attribute of the Girl object :param attractiveness: attractiveness of the Girl :param maintainance_budget: maintainance Budget the girl requires :param intelligence_level: Intelligence level of the Girl :param min_attraction_requirement: Attraction Requirement of the girl :param committed: Commitment status of the Girl 0-single 1-committed :param happiness: Happiness of the Girl

3.4.1 Methods

__init__(*self*, *name*, *attractiveness*, *maintainance_budget*, *intelligence_level*, *criterion*, *committed*, *happiness*)

isCommitted(*self*)

commit(*self*)

3.4.2 Instance Variables

Name	Description
happiness	Method to find commitment status
committed	Subclass of Girl class for Choosy

4 Module error-3-nameError.gift'

4.1 Variables

Name	Description
__package__	**Value:** `None`

4.2 Class Gift

Known Subclasses: error-3-nameError.gift.essentialGift, error-3-nameError.gift.luxuryGift, error-3-nameError.gift.utilityGift

:param: price : Price of the gift :param: value: Value of the gift

4.2.1 Methods

__init__(*self*, *price*, *value*, *used*=`False`)

4.3 Class essentialGift

error-3-nameError.gift.Gift

error-3-nameError.gift.essentialGift

4.3.1 Methods

__init__(*self*, *price*, *value*, *used*, *Type*)

Overrides: error-3-nameError.gift.Gift.__init__

4.4 Class luxuryGift

error-3-nameError.gift.Gift

error-3-nameError.gift.luxuryGift

4.4.1 Methods

__**init**__(*self*, *price*, *value*, *used*, *difficulty*, *rating*, *Type*)

Overrides: error-3-nameError.gift.Gift.__init__

4.5 Class utilityGift

error-3-nameError.gift.Gift

error-3-nameError.gift.utilityGift

4.5.1 Methods

__**init**__(*self*, *price*, *value*, *used*, *utility_value*, *utility_class*, *Type*)

Overrides: error-3-nameError.gift.Gift.__init__

5 Module error-3-nameError.girl

5.1 Variables

Name	Description
__package__	**Value:** `'error-3-nameError'`

5.2 Class Girl

Known Subclasses: error-3-nameError.girl.Choosy, error-3-nameError.girl.Desperate, error-3-nameError.girl.Normal

:param name: name attribute of the Girl object :param attractiveness: attractiveness of the Girl :param maintainance_budget: maintainance Budget the girl requires :param intelligence_level: Intelligence level of the Girl :param min_attraction_requirement: Attraction Requirement of the girl :param committed: Commitment status of the Girl 0-single 1-committed :param happiness: Happiness of the Girl

5.2.1 Methods

__init__(*self*, *name*, *attractiveness*, *maintainance_budget*, *intelligence_level*, *criterion*, *committed*, *happiness*)

isCommitted(*self*)

commit(*self*)

5.2.2 Instance Variables

Name	Description
happiness	Method to find commitment status
committed	Subclass of Girl class for Choosy

5.3 Class Choosy

error-3-nameError.girl.Girl

error-3-nameError.girl.Choosy

5.3.1 Methods

__init__(*self*, *name*, *attractiveness*, *maintainance_budget*, *intelligence_level*, *criterion*, *committed*, *happiness*)

Overrides: error-3-nameError.girl.Girl.__init__

getHappiness(*self*, *boy*, *total_gifts*, *total_gift_value*)

Inherited from error-3-nameError.girl.Girl(Section 7.9)

commit(), isCommitted()

5.3.2 Instance Variables

Name	Description
Inherited from error-3-nameError.girl.Girl (Section 7.9)	
committed, happiness	

5.4 Class Normal

error-3-nameError.girl.Girl

error-3-nameError.girl.Normal

5.4.1 Methods

__init__(*self*, *name*, *attractiveness*, *maintainance_budget*, *intelligence_level*, *criterion*, *committed*, *happiness*)

Overrides: error-3-nameError.girl.Girl.__init__

getHappiness(*self*, *boy*, *total_gifts*, *total_gift_value*)

Inherited from error-3-nameError.girl.Girl(Section 7.9)

commit(), isCommitted()

5.4.2 Instance Variables

Name	Description
Inherited from error-3-nameError.girl.Girl (Section 7.9)	

continued on next page

Name	Description
committed, happiness	

5.5 Class Desperate

error-3-nameError.girl.Girl

error-3-nameError.girl.Desperate

5.5.1 Methods

__init__(*self*, *name*, *attractiveness*, *maintainance_budget*, *intelligence_level*, *criterion*, *committed*, *happiness*)

Overrides: error-3-nameError.girl.Girl.__init__

getHappiness(*self*, *boy*, *total_gifts*, *total_gift_value*)

Inherited from error-3-nameError.girl.Girl(Section 7.9)

commit(), isCommitted()

5.5.2 Instance Variables

Name	Description
Inherited from error-3-nameError.girl.Girl (Section 7.9)	
committed, happiness	

6 Module error-3-nameError.q11

6.1 Functions

q3()

7 Module error-3-nameError.utility'

7.1 Functions

generateBoyList()

generateGiftList()

generateGirlList()

giveGifts(*boy_list*, *girl_list*)

k_least_happy_couples(*couple_list*)

k_most_happy_couples(*k*, *couple_list*)

makeCouples(*mode*, *boy_list*, *girl_list*)

randint(*a*, *b*)

Return random integer in range [a, b], including both end points.

randomName(*size*=6, *chars*='ABCDEFGHIJKLMNOPQRSTUVWXYZ')

7.2 Variables

Name	Description
__package__	**Value:** `'error-3-nameError'`

7.3 Class Boy

Known Subclasses: error-3-nameError.boy.Geek, error-3-nameError.boy.Generous, error-3-nameError.boy.Miser

:param boy: a boy object :param name: name attribute of the boy object :param attractiveness: attractiveness of the boy :param budget: Budget the boy has :param intelligence_level: Intelligence level of the boy :param min_attraction_requirement: Attraction Requirement of the boy :param committed: Commitment status of the boy 0-single 1-committed :param happiness: Happiness of the boy

7.3.1 Methods

__init__(*self*, *name*, *attractiveness*, *budget*, *intelligence_level*, *min_attraction_requirement*, *committed*, *happiness*)

commit(*self*)

isCommitted(*self*)

7.4 Class Choosy

error-3-nameError.girl.Girl

error-3-nameError.girl.Choosy

7.4.1 Methods

__init__(*self*, *name*, *attractiveness*, *maintainance_budget*, *intelligence_level*, *criterion*, *committed*, *happiness*)

Overrides: error-3-nameError.girl.Girl.__init__

getHappiness(*self*, *boy*, *total_gifts*, *total_gift_value*)

Inherited from error-3-nameError.girl.Girl(Section 7.9)

commit(), isCommitted()

7.4.2 Instance Variables

Name	Description
Inherited from error-3-nameError.girl.Girl (Section 7.9) committed, happiness	

7.5 Class Couple

7.5.1 Methods

__init__(*self*, *boy*, *girl*, *happiness*, *compatibility*)

7.6 Class Desperate

error-3-nameError.girl.Girl

error-3-nameError.girl.Desperate

7.6.1 Methods

__init__(*self*, *name*, *attractiveness*, *maintainance_budget*, *intelligence_level*, *criterion*, *committed*, *happiness*)

Overrides: error-3-nameError.girl.Girl.__init__

getHappiness(*self*, *boy*, *total_gifts*, *total_gift_value*)

Inherited from error-3-nameError.girl.Girl(Section 7.9)

commit(), isCommitted()

7.6.2 Instance Variables

Name	Description
Inherited from error-3-nameError.girl.Girl (Section 7.9) committed, happiness	

7.7 Class Geek

error-3-nameError.boy.Boy

error-3-nameError.boy.Geek

7.7.1 Methods

__init__(*self*, *name*, *attractiveness*, *budget*, *intelligence_level*, *min_attraction_requirement*, *committed*, *happiness*)

Overrides: error-3-nameError.boy.Boy.__init__

getHappiness(*self*, *girl*, *total_gifts*)

Inherited from error-3-nameError.boy.Boy(Section 7.3)

commit(), isCommitted()

7.8 Class Generous

error-3-nameError.boy.Boy

error-3-nameError.boy.Generous

7.8.1 Methods

__init__(*self*, *name*, *attractiveness*, *budget*, *intelligence_level*, *min_attraction_requirement*, *committed*, *happiness*)

Overrides: error-3-nameError.boy.Boy.__init__

getHappiness(*self*, *girl*, *total_gifts*)

Inherited from error-3-nameError.boy.Boy(Section 7.3)

commit(), isCommitted()

7.9 Class Girl

Known Subclasses: error-3-nameError.girl.Choosy, error-3-nameError.girl.Desperate, error-3-nameError.girl.Normal

:param name: name attribute of the Girl object :param attractiveness: attractiveness of the Girl :param maintainance_budget: maintainance Budget the girl requires :param intelligence_level: Intelligence level of the Girl :param min_attraction_requirement: Attraction Requirement of the girl :param committed: Commitment status of the Girl 0-single 1-committed :param happiness: Happiness of the Girl

7.9.1 Methods

__init__(*self*, *name*, *attractiveness*, *maintainance_budget*, *intelligence_level*, *criterion*, *committed*, *happiness*)

isCommitted(*self*)

commit(*self*)

7.9.2 Instance Variables

Name	Description
happiness	Method to find commitment status
committed	Subclass of Girl class for Choosy

7.10 Class Miser

error-3-nameError.boy.Boy ─┐

error-3-nameError.boy.Miser

7.10.1 Methods

__init__(*self*, *name*, *attractiveness*, *budget*, *intelligence_level*, *min_attraction_requirement*, *committed*, *happiness*)

Overrides: error-3-nameError.boy.Boy.__init__

getHappiness(*self*, *girl*, *total_gifts*)

Inherited from error-3-nameError.boy.Boy(Section 7.3)

commit(), isCommitted()

7.11 Class Normal

error-3-nameError.girl.Girl ─┐

error-3-nameError.girl.Normal

7.11.1 Methods

__init__(*self*, *name*, *attractiveness*, *maintainance_budget*, *intelligence_level*, *criterion*, *committed*, *happiness*)

Overrides: error-3-nameError.girl.Girl.__init__

getHappiness(*self*, *boy*, *total_gifts*, *total_gift_value*)

Inherited from error-3-nameError.girl.Girl(Section 7.9)

commit(), isCommitted()

7.11.2 Instance Variables

Name	Description
Inherited from error-3-nameError.girl.Girl (Section 7.9) committed, happiness	

7.12 Class essentialGift

error-3-nameError.gift.Gift ─┐

error-3-nameError.gift.essentialGift

7.12.1 Methods

__init__(*self*, *price*, *value*, *used*, *Type*)

Overrides: error-3-nameError.gift.Gift.__init__

7.13 Class luxuryGift

error-3-nameError.gift.Gift ─┐

error-3-nameError.gift.luxuryGift

7.13.1 Methods

__init__(*self*, *price*, *value*, *used*, *difficulty*, *rating*, *Type*)

Overrides: error-3-nameError.gift.Gift.__init__

7.14 Class utilityGift

error-3-nameError.gift.Gift ─┐

error-3-nameError.gift.utilityGift

7.14.1 Methods

__init__(*self*, *price*, *value*, *used*, *utility_value*, *utility_class*, *Type*)

Overrides: error-3-nameError.gift.Gift.__init__

8 Module random

```
Random variable generators.

    integers
    --------
           uniform within range

    sequences
    ---------
           pick random element
           pick random sample
           generate random permutation

    distributions on the real line:
    ------------------------------
           uniform
           triangular
           normal (Gaussian)
           lognormal
           negative exponential
           gamma
           beta
           pareto
           Weibull

    distributions on the circle (angles 0 to 2pi)
    ---------------------------------------------
           circular uniform
           von Mises

General notes on the underlying Mersenne Twister core generator:

* The period is 2**19937-1.
* It is one of the most extensively tested generators in existence.
* Without a direct way to compute N steps forward, the semantics of
  jumpahead(n) are weakened to simply jump to another distant state and rely
  on the large period to avoid overlapping sequences.
* The random() method is implemented in C, executes in a single Python step,
  and is, therefore, threadsafe.
```

8.1 Functions

betavariate(*alpha*, *beta*)

Beta distribution.

Conditions on the parameters are alpha > 0 and beta > 0. Returned values range between 0 and 1.

choice(*seq*)

Choose a random element from a non-empty sequence.

expovariate(*lambd*)

Exponential distribution.

lambd is 1.0 divided by the desired mean. It should be nonzero. (The parameter would be called ”lambda”, but that is a reserved word in Python.) Returned values range from 0 to positive infinity if lambd is positive, and from negative infinity to 0 if lambd is negative.

gammavariate(*alpha*, *beta*)

```
Gamma distribution.  Not the gamma function!

Conditions on the parameters are alpha > 0 and beta > 0.

The probability distribution function is:

            x ** (alpha - 1) * math.exp(-x / beta)
  pdf(x) =  --------------------------------------
              math.gamma(alpha) * beta ** alpha
```

gauss(*mu*, *sigma*)

Gaussian distribution.

mu is the mean, and sigma is the standard deviation. This is slightly faster than the normalvariate() function.

Not thread-safe without a lock around calls.

getrandbits(*k*)

Generates a long int with k random bits.

Return Value

x

getstate()

Return internal state; can be passed to setstate() later.

jumpahead(*n*)

Change the internal state to one that is likely far away from the current state. This method will not be in Py3.x, so it is better to simply reseed.

lognormvariate(*mu*, *sigma*)

Log normal distribution.

If you take the natural logarithm of this distribution, you'll get a normal distribution with mean mu and standard deviation sigma. mu can have any value, and sigma must be greater than zero.

normalvariate(*mu*, *sigma*)

Normal distribution.

mu is the mean, and sigma is the standard deviation.

paretovariate(*alpha*)

Pareto distribution. alpha is the shape parameter.

randint(*a*, *b*)

Return random integer in range [a, b], including both end points.

random()

Return Value

x in the interval [0, 1).

randrange(*start*, *stop*=`None`, *step*=`1`, *_int*=`<type 'int'>`, *_maxwidth*=`9007199254740992`)

Choose a random item from range(start, stop[, step]).

This fixes the problem with randint() which includes the endpoint; in Python this is usually not what you want.

sample(*population*, *k*)

Chooses k unique random elements from a population sequence.

Returns a new list containing elements from the population while leaving the original population unchanged. The resulting list is in selection order so that all sub-slices will also be valid random samples. This allows raffle winners (the sample) to be partitioned into grand prize and second place winners (the subslices).

Members of the population need not be hashable or unique. If the population contains repeats, then each occurrence is a possible selection in the sample.

To choose a sample in a range of integers, use xrange as an argument. This is especially fast and space efficient for sampling from a large population: sample(xrange(10000000), 60)

seed(*a*=`None`)

Initialize internal state from hashable object.

None or no argument seeds from current time or from an operating system specific randomness source if available.

If a is not None or an int or long, hash(a) is used instead.

setstate(*state*)

Restore internal state from object returned by getstate().

shuffle(*x*, *random*=`None`)

x, random=random.random -> shuffle list x in place; return None.

Optional arg random is a 0-argument function returning a random float in [0.0, 1.0); by default, the standard random.random.

triangular(*low*=`0.0`, *high*=`1.0`, *mode*=`None`)

Triangular distribution.

Continuous distribution bounded by given lower and upper limits, and having a given mode value in-between.

http://en.wikipedia.org/wiki/Triangular_distribution

uniform(*a*, *b*)

Get a random number in the range [a, b) or [a, b] depending on rounding.

vonmisesvariate(*mu*, *kappa*)

Circular data distribution.

mu is the mean angle, expressed in radians between 0 and 2*pi, and kappa is the concentration parameter, which must be greater than or equal to zero. If kappa is equal to zero, this distribution reduces to a uniform random angle over the range 0 to 2*pi.

weibullvariate(*alpha*, *beta*)

Weibull distribution.

alpha is the scale parameter and beta is the shape parameter.

8.2 Class Random

object

_random.Random

random.Random

Known Subclasses: random.SystemRandom, random.WichmannHill

Random number generator base class used by bound module functions.

Used to instantiate instances of Random to get generators that don't share state. Especially useful for multi-threaded programs, creating a different instance of Random for each thread, and using the jumpahead() method to ensure that the generated sequences seen by each thread don't overlap.

Class Random can also be subclassed if you want to use a different basic generator of your own devising: in that case, override the following methods: random(), seed(), getstate(), setstate() and jumpahead(). Optionally, implement a getrandbits() method so that randrange() can cover arbitrarily large ranges.

8.2.1 Methods

__getstate__(*self*)

__init__(*self*, *x*=`None`)

Initialize an instance.

Optional argument x controls seeding, as for Random.seed().

Overrides: object.__init__

__reduce__(*self*)

helper for pickle

Overrides: object.__reduce__ extit(inherited documentation)

__setstate__(*self*, *state*)

betavariate(*self*, *alpha*, *beta*)

Beta distribution.

Conditions on the parameters are alpha > 0 and beta > 0. Returned values range between 0 and 1.

choice(*self*, *seq*)

Choose a random element from a non-empty sequence.

expovariate(*self*, *lambd*)

Exponential distribution.

lambd is 1.0 divided by the desired mean. It should be nonzero. (The parameter would be called ”lambda”, but that is a reserved word in Python.) Returned values range from 0 to positive infinity if lambd is positive, and from negative infinity to 0 if lambd is negative.

gammavariate(*self*, *alpha*, *beta*)

```
Gamma distribution.  Not the gamma function!

Conditions on the parameters are alpha > 0 and beta > 0.

The probability distribution function is:

            x ** (alpha - 1) * math.exp(-x / beta)
  pdf(x) =  --------------------------------------
              math.gamma(alpha) * beta ** alpha
```

gauss(*self*, *mu*, *sigma*)

Gaussian distribution.

mu is the mean, and sigma is the standard deviation. This is slightly faster than the normalvariate() function.

Not thread-safe without a lock around calls.

getstate(*self*)

Return internal state; can be passed to setstate() later.

Return Value

tuple containing the current state.

Overrides: _random.Random.getstate

jumpahead(*self*, *n*)

Change the internal state to one that is likely far away from the current state. This method will not be in Py3.x, so it is better to simply reseed.

Return Value

None

Overrides: _random.Random.jumpahead

lognormvariate(*self*, *mu*, *sigma*)

Log normal distribution.

If you take the natural logarithm of this distribution, you'll get a normal distribution with mean mu and standard deviation sigma. mu can have any value, and sigma must be greater than zero.

normalvariate(*self*, *mu*, *sigma*)

Normal distribution.

mu is the mean, and sigma is the standard deviation.

paretovariate(*self*, *alpha*)

Pareto distribution. alpha is the shape parameter.

randint(*self*, *a*, *b*)

Return random integer in range [a, b], including both end points.

randrange(*self*, *start*, *stop*=`None`, *step*=`1`, *_int*=`<type 'int'>`, *_maxwidth*=`9007199254740992`)

Choose a random item from range(start, stop[, step]).

This fixes the problem with randint() which includes the endpoint; in Python this is usually not what you want.

sample(*self*, *population*, *k*)

Chooses k unique random elements from a population sequence.

Returns a new list containing elements from the population while leaving the original population unchanged. The resulting list is in selection order so that all sub-slices will also be valid random samples. This allows raffle winners (the sample) to be partitioned into grand prize and second place winners (the subslices).

Members of the population need not be hashable or unique. If the population contains repeats, then each occurrence is a possible selection in the sample.

To choose a sample in a range of integers, use xrange as an argument. This is especially fast and space efficient for sampling from a large population: sample(xrange(10000000), 60)

seed(*self*, *a*=`None`)

Initialize internal state from hashable object.

None or no argument seeds from current time or from an operating system specific randomness source if available.

If a is not None or an int or long, hash(a) is used instead.

Return Value

None

Overrides: _random.Random.seed

setstate(*self*, *state*)

Restore internal state from object returned by getstate().

Return Value

None

Overrides: _random.Random.setstate

shuffle(*self*, *x*, *random*=`None`)

x, random=random.random -> shuffle list x in place; return None.

Optional arg random is a 0-argument function returning a random float in [0.0, 1.0); by default, the standard random.random.

triangular(*self*, *low*=`0.0`, *high*=`1.0`, *mode*=`None`)

Triangular distribution.

Continuous distribution bounded by given lower and upper limits, and having a given mode value in-between.

http://en.wikipedia.org/wiki/Triangular_distribution

uniform(*self*, *a*, *b*)

Get a random number in the range [a, b) or [a, b] depending on rounding.

vonmisesvariate(*self*, *mu*, *kappa*)

Circular data distribution.

mu is the mean angle, expressed in radians between 0 and 2*pi, and kappa is the concentration parameter, which must be greater than or equal to zero. If kappa is equal to zero, this distribution reduces to a uniform random angle over the range 0 to 2*pi.

weibullvariate(*self*, *alpha*, *beta*)

Weibull distribution.

alpha is the scale parameter and beta is the shape parameter.

Inherited from _random.Random

__getattribute__(), __new__(), getrandbits(), random()

Inherited from object

__delattr__(), __format__(), __hash__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

8.2.2 Properties

Name	Description
Inherited from object __class__	

8.2.3 Class Variables

Name	Description
VERSION	**Value:** `3`

8.3 Class SystemRandom

object

_random.Random

random.Random

random.SystemRandom

```
Alternate random number generator using sources provided
by the operating system (such as /dev/urandom on Unix or
CryptGenRandom on Windows).

 Not available on all systems (see os.urandom() for details).
```

8.3.1 Methods

getrandbits(*k*)

Generates a long int with k random bits.

Return Value

x

Overrides: _random.Random.getrandbits

getstate(*self*, **args*, ***kwds*)

Method should not be called for a system random number generator.

Return Value

None

Overrides: _random.Random.getstate

jumpahead(*self*, **args*, ***kwds*)

Stub method. Not used for a system random number generator.

Return Value

None

Overrides: _random.Random.jumpahead

random(*self*)

Get the next random number in the range [0.0, 1.0).

Return Value

x in the interval [0, 1).

Overrides: _random.Random.random

seed(*self*, **args*, ***kwds*)

Stub method. Not used for a system random number generator.

Return Value

None

Overrides: _random.Random.seed

setstate(*self*, **args*, ***kwds*)

Method should not be called for a system random number generator.

Return Value

None

Overrides: _random.Random.setstate

Inherited from random.Random(Section 8.2)

__getstate__(), __init__(), __reduce__(), __setstate__(), betavariate(), choice(), expovariate(), gammavariate(), gauss(), lognormvariate(), normalvariate(), paretovariate(), randint(), randrange(), sample(), shuffle(), triangular(), uniform(), vonmisesvariate(), weibullvariate()

Inherited from _random.Random

__getattribute__(), __new__()

Inherited from object

__delattr__(), __format__(), __hash__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

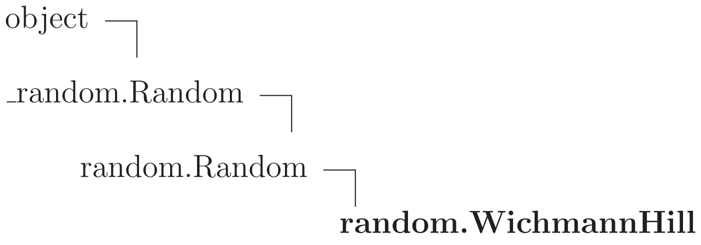
8.3.2 Properties

Name	Description
Inherited from object __class__	

8.3.3 Class Variables

Name	Description
Inherited from random.Random (Section 8.2) VERSION	

8.4 Class WichmannHill



8.4.1 Methods

getstate(*self*)

Return internal state; can be passed to setstate() later.

Return Value

tuple containing the current state.

Overrides: _random.Random.getstate

jumpahead(*self*, *n*)

```
Act as if n calls to random() were made, but quickly.

n is an int, greater than or equal to 0.

Example use:  If you have 2 threads and know that each will
consume no more than a million random numbers, create two Random
objects r1 and r2, then do
    r2.setstate(r1.getstate())
    r2.jumpahead(1000000)
Then r1 and r2 will use guaranteed-disjoint segments of the full
period.
```

Return Value

None

Overrides: _random.Random.jumpahead

random(*self*)

Get the next random number in the range [0.0, 1.0).

Return Value

x in the interval [0, 1).

Overrides: _random.Random.random

seed(*self*, *a*=`None`)

Initialize internal state from hashable object.

None or no argument seeds from current time or from an operating system specific randomness source if available.

If a is not None or an int or long, hash(a) is used instead.

If a is an int or long, a is used directly. Distinct values between 0 and 27814431486575L inclusive are guaranteed to yield distinct internal states (this guarantee is specific to the default Wichmann-Hill generator).

Return Value

None

Overrides: _random.Random.seed

setstate(*self*, *state*)

Restore internal state from object returned by getstate().

Return Value

None

Overrides: _random.Random.setstate

whseed(*self*, *a*=`None`)

Seed from hashable object's hash code.

None or no argument seeds from current time. It is not guaranteed that objects with distinct hash codes lead to distinct internal states.

This is obsolete, provided for compatibility with the seed routine used prior to Python 2.1. Use the .seed() method instead.

Inherited from random.Random(Section 8.2)

__getstate__(), __init__(), __reduce__(), __setstate__(), betavariate(), choice(), expovariate(), gammavariate(), gauss(), lognormvariate(), normalvariate(), paretovariate(), randint(), randrange(), sample(), shuffle(), triangular(), uniform(), vonmisesvariate(), weibullvariate()

Inherited from _random.Random

__getattribute__(), __new__(), getrandbits()

Inherited from object

__delattr__(), __format__(), __hash__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

8.4.2 Properties

Name	Description
Inherited from object __class__	

8.4.3 Class Variables

Name	Description
VERSION	**Value:** `1`

9 Script script-boys_list_txt

10 Script script-couple_list_txt

10.1 Variables

Name	Description
__package__	**Value:** `None`

11 Script script-gift_list_txt

12 Script script-girls_list_txt

13 Module string

A collection of string operations (most are no longer used).

Warning: most of the code you see here isn't normally used nowadays. Beginning with Python 1.6, many of these functions are implemented as methods on the standard string object. They used to be implemented by a built-in module called strop, but strop is now obsolete itself.

Public module variables:

whitespace – a string containing all characters considered whitespace lowercase – a string containing all characters considered lowercase letters uppercase – a string containing all characters considered uppercase letters letters – a string containing all characters considered letters digits – a string containing all characters considered decimal digits hexdigits – a string containing all characters considered hexadecimal digits octdigits – a string containing all characters considered octal digits punctuation – a string containing all characters considered punctuation printable – a string containing all characters considered printable

13.1 Functions

atof(*s*)

Return the floating point number represented by the string s.

Return Value

float

atoi(*s*, *base*=...)

Return the integer represented by the string s in the given base, which defaults to 10. The string s must consist of one or more digits, possibly preceded by a sign. If base is 0, it is chosen from the leading characters of s, 0 for octal, 0x or 0X for hexadecimal. If base is 16, a preceding 0x or 0X is accepted.

Return Value

int

atol(*s*, *base*=...)

Return the long integer represented by the string s in the given base, which defaults to 10. The string s must consist of one or more digits, possibly preceded by a sign. If base is 0, it is chosen from the leading characters of s, 0 for octal, 0x or 0X for hexadecimal. If base is 16, a preceding 0x or 0X is accepted. A trailing L or l is not accepted, unless base is 0.

Return Value

long

capitalize(*s*)

Return a copy of the string s with only its first character capitalized.

Return Value

string

capwords(*s*, *sep*=...)

Split the argument into words using split, capitalize each word using capitalize, and join the capitalized words using join. If the optional second argument sep is absent or None, runs of whitespace characters are replaced by a single space and leading and trailing whitespace are removed, otherwise sep is used to split and join the words.

Return Value

string

center(*s*, *width*, *fillchar*=...)

Return a center version of s, in a field of the specified width. padded with spaces as needed. The string is never truncated. If specified the fillchar is used instead of spaces.

Return Value

string

count(*s*, *sub*, *start*=..., *end*=...)

Return the number of occurrences of substring sub in string s[start:end]. Optional arguments start and end are interpreted as in slice notation.

Return Value

int

expandtabs(*s*, *tabsize*=...)

Return a copy of the string s with all tab characters replaced by the appropriate number of spaces, depending on the current column, and the tabsize (default 8).

Return Value
string

find(*s*, *sub*, *start*=... , *end*=...)

Return the lowest index in s where substring sub is found, such that sub is contained within s[start,end]. Optional arguments start and end are interpreted as in slice notation.

Return -1 on failure.

Return Value
in

index(*s*, *sub*, *start*=... , *end*=...)

Like find but raises ValueError when the substring is not found.

Return Value
int

join(*list*, *sep*=...)

Return a string composed of the words in list, with intervening occurrences of sep. The default separator is a single space.

(joinfields and join are synonymous)

Return Value
string

joinfields(*list*, *sep*=...)

Return a string composed of the words in list, with intervening occurrences of sep. The default separator is a single space.

(joinfields and join are synonymous)

Return Value
string

ljust(*s*, *width*, *fillchar*=...)

Return a left-justified version of s, in a field of the specified width, padded with spaces as needed. The string is never truncated. If specified the fillchar is used instead of spaces.

Return Value
string

lower(*s*)

Return a copy of the string s converted to lowercase.

Return Value
string

lstrip(*s*, *chars*=...)

Return a copy of the string s with leading whitespace removed. If chars is given and not None, remove characters in chars instead.

Return Value
string

replace(*s*, *old*, *new*, *maxreplace*=`-1`)

replace (str, old, new[, maxreplace]) -> string

Return a copy of string str with all occurrences of substring old replaced by new. If the optional argument maxreplace is given, only the first maxreplace occurrences are replaced.

rfind(*s*, *sub*, *start*=... , *end*=...)

Return the highest index in s where substring sub is found, such that sub is contained within s[start,end]. Optional arguments start and end are interpreted as in slice notation.

Return -1 on failure.

Return Value
int

rindex(*s*, *sub*, *start*=... , *end*=...)

Like rfind but raises ValueError when the substring is not found.

Return Value
int

rjust(*s*, *width*, *fillchar*=...)

Return a right-justified version of s, in a field of the specified width, padded with spaces as needed. The string is never truncated. If specified the fillchar is used instead of spaces.

Return Value

string

rsplit(*s*, *sep*=...   , *maxsplit*=...)

Return a list of the words in the string s, using sep as the delimiter string, starting at the end of the string and working to the front. If maxsplit is given, at most maxsplit splits are done. If sep is not specified or is None, any whitespace string is a separator.

Return Value

list of strings

rstrip(*s*, *chars*=...)

Return a copy of the string s with trailing whitespace removed. If chars is given and not None, remove characters in chars instead.

Return Value

string

split(*s*, *sep*=...   , *maxsplit*=...)

Return a list of the words in the string s, using sep as the delimiter string. If maxsplit is given, splits at no more than maxsplit places (resulting in at most maxsplit+1 words). If sep is not specified or is None, any whitespace string is a separator.

(split and splitfields are synonymous)

Return Value

list of strings

splitfields(*s*, *sep*=... , *maxsplit*=...)

Return a list of the words in the string s, using sep as the delimiter string. If maxsplit is given, splits at no more than maxsplit places (resulting in at most maxsplit+1 words). If sep is not specified or is None, any whitespace string is a separator.

(split and splitfields are synonymous)

Return Value

list of strings

strip(*s*, *chars*=...)

Return a copy of the string s with leading and trailing whitespace removed. If chars is given and not None, remove characters in chars instead. If chars is unicode, S will be converted to unicode before stripping.

Return Value

string

swapcase(*s*)

Return a copy of the string s with upper case characters converted to lowercase and vice versa.

Return Value

string

translate(*s*, *table*, *deletions*=...)

Return a copy of the string s, where all characters occurring in the optional argument deletions are removed, and the remaining characters have been mapped through the given translation table, which must be a string of length 256. The deletions argument is not allowed for Unicode strings.

Return Value

string

upper(*s*)

Return a copy of the string s converted to uppercase.

Return Value

string

zfill(*x*, *width*)

Pad a numeric string x with zeros on the left, to fill a field of the specified width. The string x is never truncated.

Return Value

string

13.2 Variables

Name	Description
__package__	**Value:** None
ascii_letters	**Value:** 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
ascii_lowercase	**Value:** 'abcdefghijklmnopqrstuvwxyz'
ascii_uppercase	**Value:** 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
digits	**Value:** '0123456789'
hexdigits	**Value:** '0123456789abcdefABCDEF'
letters	**Value:** 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
lowercase	**Value:** 'abcdefghijklmnopqrstuvwxyz'
octdigits	**Value:** '01234567'
printable	**Value:** '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRST.
punctuation	**Value:** '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
uppercase	**Value:** 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
whitespace	**Value:** '\t\n\x0b\x0c\r '

13.3 Class Formatter

object ─┐
string.Formatter

13.3.1 Methods

check_unused_args(*self*, *used_args*, *args*, *kwargs*)

convert_field(*self*, *value*, *conversion*)

format(*args, **kwargs)

format_field(*self*, *value*, *format_spec*)

get_field(*self*, *field_name*, *args*, *kwargs*)

get_value(*self*, *key*, *args*, *kwargs*)

parse(*self*, *format_string*)

vformat(*self*, *format_string*, *args*, *kwargs*)

Inherited from object

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

13.3.2 Properties

Name	Description
Inherited from object __class__	

13.4 Class Template

object
    string.Template

A string class for supporting $-substitutions.

13.4.1 Methods

__init__(*self*, *template*)

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

Overrides: object.__init__ extit(inherited documentation)

safe_substitute(*args, **kws)

substitute(**args*, ***kws*)

Inherited from object

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

13.4.2 Properties

Name	Description
Inherited from object __class__	

13.4.3 Class Variables

Name	Description
delimiter	**Value:** `'$'`
idpattern	**Value:** `'[_a-z][_a-z0-9]*'`
pattern	**Value:** `re.compile(r'(?ix)\$(?:(?P<escaped>\$)|(?P<named>[_a-z][_`

14 Module sys

```
This module provides access to some objects used or maintained by the
interpreter and to functions that interact strongly with the interpreter.

Dynamic objects:

argv -- command line arguments; argv[0] is the script pathname if known
path -- module search path; path[0] is the script directory, else ’’
modules -- dictionary of loaded modules

displayhook -- called to show results in an interactive session
excepthook -- called to handle any uncaught exception other than SystemExit
  To customize printing in an interactive session or to install a custom
  top-level exception handler, assign other functions to replace these.

exitfunc -- if sys.exitfunc exists, this routine is called when Python exits
  Assigning to sys.exitfunc is deprecated; use the atexit module instead.

stdin -- standard input file object; used by raw_input() and input()
stdout -- standard output file object; used by the print statement
stderr -- standard error object; used for error messages
  By assigning other file objects (or objects that behave like files)
  to these, it is possible to redirect all of the interpreter’s I/O.

last_type -- type of last uncaught exception
last_value -- value of last uncaught exception
last_traceback -- traceback of last uncaught exception
  These three are only available in an interactive session after a
  traceback has been printed.

exc_type -- type of exception currently being handled
exc_value -- value of exception currently being handled
exc_traceback -- traceback of exception currently being handled
  The function exc_info() should be used instead of these three,
  because it is thread-safe.

Static objects:

float_info -- a dict with information about the float inplementation.
long_info -- a struct sequence with information about the long implementation.
maxint -- the largest supported integer (the smallest is -maxint-1)
maxsize -- the largest supported length of containers.
maxunicode -- the largest supported character
```

```
builtin_module_names -- tuple of module names built into this interpreter
version -- the version of this interpreter as a string
version_info -- version information as a named tuple
hexversion -- version information encoded as a single integer
copyright -- copyright notice pertaining to this interpreter
platform -- platform identifier
executable -- absolute path of the executable binary of the Python interpreter
prefix -- prefix used to find the Python library
exec_prefix -- prefix used to find the machine-specific Python library
float_repr_style -- string indicating the style of repr() output for floats
__stdin__ -- the original stdin; don't touch!
__stdout__ -- the original stdout; don't touch!
__stderr__ -- the original stderr; don't touch!
__displayhook__ -- the original displayhook; don't touch!
__excepthook__ -- the original excepthook; don't touch!

Functions:

displayhook() -- print an object to the screen, and save it in __builtin__._
excepthook() -- print an exception and its traceback to sys.stderr
exc_info() -- return thread-safe information about the current exception
exc_clear() -- clear the exception state for the current thread
exit() -- exit the interpreter by raising SystemExit
getdlopenflags() -- returns flags to be used for dlopen() calls
getprofile() -- get the global profiling function
getrefcount() -- return the reference count for an object (plus one :-)
getrecursionlimit() -- return the max recursion depth for the interpreter
getsizeof() -- return the size of an object in bytes
gettrace() -- get the global debug tracing function
setcheckinterval() -- control how often the interpreter checks for events
setdlopenflags() -- set the flags to be used for dlopen() calls
setprofile() -- set the global profiling function
setrecursionlimit() -- set the max recursion depth for the interpreter
settrace() -- set the global debug tracing function
```

14.1 Functions

__displayhook__(*object*)

Print an object to sys.stdout and also save it in __builtin__._

Return Value

None

__excepthook__(*exctype*, *value*, *traceback*)

Handle an exception by displaying it with a traceback on sys.stderr.

Return Value

None

call_tracing(*func*, *args*)

Call func(*args), while tracing is enabled. The tracing state is saved, and restored afterwards. This is intended to be called from a debugger from a checkpoint, to recursively debug some other code.

Return Value

object

callstats()

```
Return a tuple of function call statistics, if CALL_PROFILE was defined
when Python was built.  Otherwise, return None.

When enabled, this function returns detailed, implementation-specific
details about the number of function calls executed. The return value is
a 11-tuple where the entries in the tuple are counts of:
0. all function calls
1. calls to PyFunction_Type objects
2. PyFunction calls that do not create an argument tuple
3. PyFunction calls that do not create an argument tuple
   and bypass PyEval_EvalCodeEx()
4. PyMethod calls
5. PyMethod calls on bound methods
6. PyType calls
7. PyCFunction calls
8. generator calls
9. All other calls
10. Number of stack pops performed by call_function()
```

Return Value

tuple of integers

displayhook(*object*)

Print an object to sys.stdout and also save it in __builtin__._

Return Value

None

exc_clear()

Clear global information on the current exception. Subsequent calls to exc_info() will return (None,None,None) until another exception is raised in the current thread or the execution stack returns to a frame where another exception is being handled.

Return Value

None

exc_info()

Return information about the most recent exception caught by an except clause in the current stack frame or in an older stack frame.

Return Value

(type, value, traceback)

excepthook(*exctype*, *value*, *traceback*)

Handle an exception by displaying it with a traceback on sys.stderr.

Return Value

None

exit(*status*=. . .)

Exit the interpreter by raising SystemExit(status). If the status is omitted or None, it defaults to zero (i.e., success). If the status is an integer, it will be used as the system exit status. If it is another kind of object, it will be printed and the system exit status will be one (i.e., failure).

getcheckinterval()

Return Value

current check interval; see setcheckinterval().

getdefaultencoding()

Return the current default string encoding used by the Unicode implementation.

Return Value

string

getdlopenflags()

Return the current value of the flags that are used for dlopen calls. The flag constants are defined in the ctypes and DLFCN modules.

Return Value
int

getfilesystemencoding()

Return the encoding used to convert Unicode filenames in operating system filenames.

Return Value
string

getprofile()

Return the profiling function set with sys.setprofile. See the profiler chapter in the library manual.

getrecursionlimit()

Return the current value of the recursion limit, the maximum depth of the Python interpreter stack. This limit prevents infinite recursion from causing an overflow of the C stack and crashing Python.

getrefcount(*object*)

Return the reference count of object. The count returned is generally one higher than you might expect, because it includes the (temporary) reference as an argument to getrefcount().

Return Value
integer

getsizeof(*object*, *default*)

Return the size of object in bytes.

Return Value
int

gettrace()

Return the global debug tracing function set with sys.settrace. See the debugger chapter in the library manual.

setcheckinterval(*n*)

Tell the Python interpreter to check for asynchronous events every n instructions. This also affects how often thread switches occur.

setdlopenflags(*n*)

Set the flags used by the interpreter for dlopen calls, such as when the interpreter loads extension modules. Among other things, this will enable a lazy resolving of symbols when importing a module, if called as sys.setdlopenflags(0). To share symbols across extension modules, call as sys.setdlopenflags(ctypes.RTLD_GLOBAL). Symbolic names for the flag modules can be either found in the ctypes module, or in the DLFCN module. If DLFCN is not available, it can be generated from /usr/include/dlfcn.h using the h2py script.

Return Value

None

setprofile(*function*)

Set the profiling function. It will be called on each function call and return. See the profiler chapter in the library manual.

setrecursionlimit(*n*)

Set the maximum depth of the Python interpreter stack to n. This limit prevents infinite recursion from causing an overflow of the C stack and crashing Python. The highest possible limit is platform- dependent.

settrace(*function*)

Set the global debug tracing function. It will be called on each function call. See the debugger chapter in the library manual.

14.2 Variables

Name	Description
__package__	**Value:** `None`
__stderr__	**Value:** `<open file '<stderr>', mode 'w' at 0x107a321e0>`
__stdin__	**Value:** `<open file '<stdin>', mode 'r' at 0x107a320c0>`
__stdout__	**Value:** `<open file '<stdout>', mode 'w' at 0x107a32150>`
api_version	**Value:** `1013`

continued on next page

Name	Description
argv	**Value:** ['/usr/local/bin/epydoc', '-v', '-o', 'Doc', '--pdf', '--...
builtin_module_names	**Value:** ('__builtin__', '__main__', '_ast', '_codecs', '_sre', '_...
byteorder	**Value:** 'little'
copyright	**Value:** 'Copyright (c) 2001-2016 Python Software Foundation.\nAll...
dont_write_bytecode	**Value:** False
exc_type	**Value:** None
exec_prefix	**Value:** '/usr/local/Cellar/python/2.7.13/Frameworks/Python.framew...
executable	**Value:** '/usr/local/opt/python/bin/python2.7'
flags	**Value:** sys.flags(debug=0, py3k_warning=0, division_warning=0, di...
float_info	**Value:** sys.float_info(max=1.7976931348623157e+308, max_exp=1024,...
float_repr_style	**Value:** 'short'
hexversion	**Value:** 34016752
long_info	**Value:** sys.long_info(bits_per_digit=30, sizeof_digit=4)
maxint	**Value:** 9223372036854775807
maxsize	**Value:** 9223372036854775807
maxunicode	**Value:** 65535
meta_path	**Value:** []
modules	**Value:** {'ConfigParser': <module 'ConfigParser' from '/usr/local/...
path	**Value:** ['/Users/abhinav/Documents/Study/PPL/ppl-assignment-Cosmi...
path_hooks	**Value:** [<type 'zipimport.zipimporter'>]
path_importer_cache	**Value:** {'': None, '/Users/abhinav/Documents/Study/PPL/ppl-assign...
platform	**Value:** 'darwin'
prefix	**Value:** '/usr/local/Cellar/python/2.7.13/Frameworks/Python.framew...
py3kwarning	**Value:** False
stderr	**Value:** <open file '<stderr>', mode 'w' at 0x107a321e0>
stdin	**Value:** <open file '<stdin>', mode 'r' at 0x107a320c0>

continued on next page

Name	Description
stdout	**Value:** `<open file '<stdout>', mode 'w' at 0x107a32150>`
subversion	**Value:** `('CPython', '', '')`
version	**Value:** `'2.7.13 (default, Dec 18 2016, 07:03:39) \n[GCC 4.2.1 Com...`
version_info	**Value:** `sys.version_info(major=2, minor=7, micro=13, releaselevel...`
warnoptions	**Value:** `[]`

Index