

Selected Project Euler Exercises

Generated by Doxygen 1.8.9.1

Thu Oct 15 2015 21:44:34

Contents

1	Namespace Index	1
1.1	Packages	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Namespace Documentation	9
5.1	cards Namespace Reference	9
5.1.1	Function Documentation	9
5.1.1.1	cards_cmp	9
5.1.1.2	denom	9
5.2	combinatorics Namespace Reference	9
5.2.1	Function Documentation	9
5.2.1.1	digits	9
5.2.1.2	fib	9
5.2.1.3	has_square	10
5.2.1.4	nub	10
5.3	primes Namespace Reference	10
5.3.1	Function Documentation	10
5.3.1.1	factors	10
5.3.1.2	is_next_prime	10
5.3.1.3	is_prime	10
5.3.1.4	next_prime	10
5.3.1.5	odd_composite	10
5.3.1.6	primes	10
5.3.1.7	primes_seq	10

6	Class Documentation	11
6.1	cards.Hand Class Reference	11
6.1.1	Constructor & Destructor Documentation	11
6.1.1.1	__init__	11
6.1.2	Member Function Documentation	12
6.1.2.1	__eq__	12
6.1.2.2	__ge__	12
6.1.2.3	__gt__	12
6.1.2.4	__le__	12
6.1.2.5	__lt__	12
6.1.2.6	__ne__	12
6.1.2.7	break_tie	12
6.1.2.8	flush	12
6.1.2.9	four_of_a_kind	12
6.1.2.10	full_house	12
6.1.2.11	n_of_a_kind	12
6.1.2.12	one_pair	12
6.1.2.13	royal_flush	12
6.1.2.14	scoring_cards	12
6.1.2.15	straight	12
6.1.2.16	straight_flush	12
6.1.2.17	three_of_a_kind	12
6.1.2.18	two_pairs	12
6.1.2.19	value	12
7	File Documentation	13
7.1	cards.py File Reference	13
7.2	combinatorics.py File Reference	13
7.3	primes.py File Reference	13
	Index	15

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

cards	9
combinatorics	9
primes	10

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

object	
cards.Hand	11

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

cards.Hand	11
--------------------------------------	----

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

cards.py	13
combinatorics.py	13
primes.py	13

Chapter 5

Namespace Documentation

5.1 cards Namespace Reference

Classes

- class `Hand`

Functions

- def `cards_cmp` (a, b)
- def `denom` (card)

5.1.1 Function Documentation

5.1.1.1 `def cards.cards_cmp (a, b)`

5.1.1.2 `def cards.denom (card)`

5.2 combinatorics Namespace Reference

Functions

- def `fib` (n)
- def `digits` (n)
- def `nub` (nums)
- def `has_square` (n)

5.2.1 Function Documentation

5.2.1.1 `def combinatorics.digits (n)`

The digits of the argument, not counting the sign.

5.2.1.2 `def combinatorics.fib (n)`

It is possible to compute N'th Fibonacci number analytically, but I was asked not to do this, and to keep it simple. So, here you have the iterative version.

5.2.1.3 `def combinatorics.has_square (n)`

Returns True iff n contains a perfect square as a factor.

5.2.1.4 `def combinatorics.nub (nums)`

Returns a copy of nums with duplicates removed. The numbers in nums will be sorted in increasing order.

5.3 primes Namespace Reference

Functions

- `def primes_seq (n)`
- `def is_next_prime (n, sieve)`
- `def is_prime (n, sieve)`
- `def next_prime (sieve)`
- `def primes`
- `def odd_composite ()`
- `def factors (n)`

5.3.1 Function Documentation

5.3.1.1 `def primes.factors (n)`

Generates all prime factors of n.

5.3.1.2 `def primes.is_next_prime (n, sieve)`

Tests whether n is prime. Sieve should contain all primes less than n.

5.3.1.3 `def primes.is_prime (n, sieve)`

5.3.1.4 `def primes.next_prime (sieve)`

Given the sieve containing primes searches for the prime greater than the last element of the sieve, such that there are no primes less than this one which are not in the sieve.

5.3.1.5 `def primes.odd_composite ()`

Iterator. Generates odd composite numbers.

5.3.1.6 `def primes.primes (n=None)`

Iterator. Generates primes.

5.3.1.7 `def primes.primes_seq (n)`

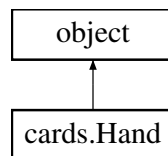
Generates a sequence of primes up to n.

Chapter 6

Class Documentation

6.1 cards.Hand Class Reference

Inheritance diagram for cards.Hand:



Public Member Functions

- def `__init__` (self, cards)
- def `n_of_a_kind` (self, n)
- def `royal_flush` (self)
- def `straight_flush` (self)
- def `four_of_a_kind` (self)
- def `full_house` (self)
- def `flush` (self)
- def `straight` (self)
- def `three_of_a_kind` (self)
- def `two_pairs` (self)
- def `one_pair` (self)
- def `value` (self)
- def `scoring_cards` (self, score)
- def `break_tie` (self, other)
- def `__lt__` (self, other)
- def `__le__` (self, other)
- def `__eq__` (self, other)
- def `__ne__` (self, other)
- def `__gt__` (self, other)
- def `__ge__` (self, other)

6.1.1 Constructor & Destructor Documentation

6.1.1.1 def cards.Hand.__init__(self, cards)

6.1.2 Member Function Documentation

6.1.2.1 `def cards.Hand.__eq__(self, other)`

6.1.2.2 `def cards.Hand.__ge__(self, other)`

6.1.2.3 `def cards.Hand.__gt__(self, other)`

6.1.2.4 `def cards.Hand.__le__(self, other)`

6.1.2.5 `def cards.Hand.__lt__(self, other)`

6.1.2.6 `def cards.Hand.__ne__(self, other)`

6.1.2.7 `def cards.Hand.break_tie(self, other)`

6.1.2.8 `def cards.Hand.flush(self)`

6.1.2.9 `def cards.Hand.four_of_a_kind(self)`

6.1.2.10 `def cards.Hand.full_house(self)`

6.1.2.11 `def cards.Hand.n_of_a_kind(self, n)`

6.1.2.12 `def cards.Hand.one_pair(self)`

6.1.2.13 `def cards.Hand.royal_flush(self)`

6.1.2.14 `def cards.Hand.scoring_cards(self, score)`

6.1.2.15 `def cards.Hand.straight(self)`

6.1.2.16 `def cards.Hand.straight_flush(self)`

6.1.2.17 `def cards.Hand.three_of_a_kind(self)`

6.1.2.18 `def cards.Hand.two_pairs(self)`

6.1.2.19 `def cards.Hand.value(self)`

The documentation for this class was generated from the following file:

- [cards.py](#)

Chapter 7

File Documentation

7.1 cards.py File Reference

Classes

- class [cards.Hand](#)

Namespaces

- [cards](#)

Functions

- def [cards.cards_cmp](#) (a, b)
- def [cards.denom](#) (card)

7.2 combinatorics.py File Reference

Namespaces

- [combinatorics](#)

Functions

- def [combinatorics.fib](#) (n)
- def [combinatorics.digits](#) (n)
- def [combinatorics.nub](#) (nums)
- def [combinatorics.has_square](#) (n)

7.3 primes.py File Reference

Namespaces

- [primes](#)

Functions

- def `primes.primes_seq` (n)
- def `primes.is_next_prime` (n, sieve)
- def `primes.is_prime` (n, sieve)
- def `primes.next_prime` (sieve)
- def `primes.primes`
- def `primes.odd_composite` ()
- def `primes.factors` (n)

Index

- `__eq__`
 - `cards::Hand`, 12
 - `__ge__`
 - `cards::Hand`, 12
 - `__gt__`
 - `cards::Hand`, 12
 - `__init__`
 - `cards::Hand`, 11
 - `__le__`
 - `cards::Hand`, 12
 - `__lt__`
 - `cards::Hand`, 12
 - `__ne__`
 - `cards::Hand`, 12
- `break_tie`
 - `cards::Hand`, 12
- `cards`, 9
 - `cards_cmp`, 9
 - `denom`, 9
- `cards.Hand`, 11
- `cards.py`, 13
- `cards::Hand`
 - `__eq__`, 12
 - `__ge__`, 12
 - `__gt__`, 12
 - `__init__`, 11
 - `__le__`, 12
 - `__lt__`, 12
 - `__ne__`, 12
 - `break_tie`, 12
 - `flush`, 12
 - `four_of_a_kind`, 12
 - `full_house`, 12
 - `n_of_a_kind`, 12
 - `one_pair`, 12
 - `royal_flush`, 12
 - `scoring_cards`, 12
 - `straight`, 12
 - `straight_flush`, 12
 - `three_of_a_kind`, 12
 - `two_pairs`, 12
 - `value`, 12
- `cards_cmp`
 - `cards`, 9
- `combinatorics`, 9
 - `digits`, 9
 - `fib`, 9
 - `has_square`, 9
 - `nub`, 10
 - `combinatorics.py`, 13
- `denom`
 - `cards`, 9
- `digits`
 - `combinatorics`, 9
- `factors`
 - `primes`, 10
- `fib`
 - `combinatorics`, 9
- `flush`
 - `cards::Hand`, 12
- `four_of_a_kind`
 - `cards::Hand`, 12
- `full_house`
 - `cards::Hand`, 12
- `has_square`
 - `combinatorics`, 9
- `is_next_prime`
 - `primes`, 10
- `is_prime`
 - `primes`, 10
- `n_of_a_kind`
 - `cards::Hand`, 12
- `next_prime`
 - `primes`, 10
- `nub`
 - `combinatorics`, 10
- `odd_composite`
 - `primes`, 10
- `one_pair`
 - `cards::Hand`, 12
- `primes`, 10
 - `factors`, 10
 - `is_next_prime`, 10
 - `is_prime`, 10
 - `next_prime`, 10
 - `odd_composite`, 10
 - `primes`, 10
 - `primes_seq`, 10
- `primes.py`, 13
- `primes_seq`
 - `primes`, 10

royal_flush
 cards::Hand, [12](#)

scoring_cards
 cards::Hand, [12](#)

straight
 cards::Hand, [12](#)

straight_flush
 cards::Hand, [12](#)

three_of_a_kind
 cards::Hand, [12](#)

two_pairs
 cards::Hand, [12](#)

value
 cards::Hand, [12](#)