

---

# Simpler

Juan C. Roldán

Sep 15, 2023



**CONTENTS:**

<b>1</b>	<b>simpler package</b>	<b>1</b>
1.1	Submodules	1
1.1.1	simpler.algorithms module	1
1.1.2	simpler.bioinformatics module	2
1.1.3	simpler.connectors module	2
1.1.4	simpler.files module	5
1.1.5	simpler.format module	7
1.1.6	simpler.mail module	7
1.1.7	simpler.math module	8
1.1.8	simpler.profilng module	9
1.1.9	simpler.sparql module	10
1.1.10	simpler.terminal module	10
1.1.11	simpler.tests module	10
1.1.12	simpler.validation module	11
1.1.13	simpler.web module	11
1.2	Module contents	14
<b>2</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>
	<b>Index</b>	<b>19</b>



## SIMPLER PACKAGE

## 1.1 Submodules

### 1.1.1 `simpler.algorithms` module

**class** `simpler.algorithms.DynamicProgramming`(*initial\_state*)

Bases: `object`

Abstract class to solve problems using dynamic programming. To use it, make a child class implementing `alternatives`, `is_final` and `penalty`. Then, make one of such objects providing the initial state in the constructor, and call to `solve()`, providing one search type.

**ALL\_OPTIMAL\_SOLUTIONS** = 3

**ALL\_SOLUTIONS** = 2

**ONE\_OPTIMAL\_SOLUTION** = 1

**ONE\_SOLUTION** = 0

**alternatives**(*state*)

This should return all the alternatives for a given state, without modifying the given state.

**Return type**

`list`

**is\_final**(*state*)

This should return whether a state is a final state or not.

**Return type**

`bool`

**penalty**(*state*)

This should return an upper bound for the penalty of the problem. Ideally, optimal solutions should have no penalty.

**Return type**

`float`

**solve**(*search\_type=0*)

Resolves the Dynamic Programming problem.

`simpler.algorithms.deep_merge(dict1, dict2)`

This function merges two dictionaries.

**Return type**  
dict

### 1.1.2 simpler.bioinformatics module

`simpler.bioinformatics.dna_to_rna(dna)`

Transforms a DNA string into RNA by replacing Ts with Us.

**Return type**  
str

`simpler.bioinformatics.parse_fasta(data_string, first=False)`

Given a string in FASTA format, returns the DNA strings on it.

`simpler.bioinformatics.reverse_complement(seq, is_rna=True)`

Obtains the reverse complement of a DNA or RNA string.

**Return type**  
str

`simpler.bioinformatics.rna_to_dna(rna)`

Transforms a RNA string into DNA by replacing Us with Ts.

**Return type**  
str

`simpler.bioinformatics.rna_to_protein(rna)`

Transforms a RNA string into a CODON string.

**Return type**  
str

### 1.1.3 simpler.connectors module

`class simpler.connectors.Excel(path)`

Bases: object

Pandas Excel backend.

`block_from_code(code)`

Transforms an Excel code like A4:B5 to block delimiters like (3, 0, 5, 2).

**Return type**  
tuple

`cell(row, col, sheet=0)`

Retrieves a cell from the book.

`cells(block, sheet=0)`

Retrieves a square of cells data from a block.

**index\_from\_code**(code)

Transforms an Excel code like ABC to an index like 731.

**Return type**

int

**sheet**(sheet=0)

Loads a sheet given its name or position in the book.

**table**(data, hrows=None, hcols=None, sheet=0)

```
class simpler.connectors.SQL(host='localhost', user=None, password=None, db=None, charset='utf8mb4',
                             collation='utf8mb4_general_ci', use_unicode=True, max_insertions=None,
                             print_queries=False, native_types=True, engine='mysql', force_init=False)
```

Bases: object

Connector for SQL databases with a handful of helpers.

**ENGINES** = ('mysql', 'mariadb', 'mssql', 'postgre')

**apply**(query, \*params)

Applies a modification (update or delete) and returns the number of affected rows.

**Return type**

int

**close**()

Closes the current cursor and connection.

**Return type**

None

**cursor**()

Returns the open cursor and initializes the connection if required.

**delete**(table, filters=<function SQL.<lambda>>>)

Executes a delete operation and returns the number of affected rows, specifying a filters list, i.e. {'a': 4, 'b': None} will be translated into *WHERE A = 4 AND B = NULL*.

**Return type**

int

**escape**(value, is\_literal=True)

Escapes the given value for its injection into the SQL query. By default, the data *is\_literal=True*, which will wrap strings with quotes for its insertion.

**Return type**

str

**execute**(query, params=None, multi=False, commit=False)

Wrapper for the database connector execute method that won't send the params argument if the params are empty, thus avoiding the need to replace % with %%.

**exists**(table, column, value)

Returns True if the value exists in the specified column of the specified table.

**Return type**

bool

**find**(*query*, \**params*)

Returns a {column: value} dict of the first selected row.

**Return type**  
dict**find\_all**(*query*, \**params*)

Returns a list of {column: value} dicts of the selected rows.

**Return type**  
List[dict]**find\_all\_tuples**(*query*, \**params*)

Returns a list of tuples of the selected rows.

**Return type**  
List[tuple]**find\_column**(*query*, \**params*)

Returns the value of the first column of every selected row.

**Return type**  
list**find\_tuple**(*query*, \**params*)

Returns a tuple of the values of the first selected row.

**Return type**  
tuple**find\_value**(*query*, \**params*)

Returns the value of the first column of the first selected row.

**Return type**  
Any**insert**(*query*, \**params*)

Inserts a row and returns its id (if engine="postgres", you'll have to use the RETURNING keyword).

**Return type**  
int**insert\_all**(*table*, *rows*, *tuple\_rows=False*, *commit=True*)Insert a list of rows and returns the id of the last one. By default, these rows are a list of {column: value} dicts, but they can be inserted from tuples of values setting *tuple\_rows* to True.**Return type**  
Optional[int]**iter\_all**(*query*, \**params*)

Returns a generator of {column: value} dicts of the selected rows.

**Return type**  
Generator[dict, None, None]**iter\_all\_tuples**(*query*, \**params*)

Returns a generator of tuples of the selected rows.

**Return type**  
Generator[tuple, None, None]



**iter\_column**(*query*, \**params*)

Returns a generator of the first column of every selected row.

**Return type**

Generator[list, None, None]

**print\_query**(*query*, *params*=None, *color*='yellow', *max\_size*=1000)

Shows a query attempting to inject the parameters, for debugging purposes.

**select**(*table*, *filters*=<function SQL.<lambda>>, *first\_row*=False, *first\_column*=False, *tuple\_rows*=True, *or\_filters*=False)

Executes an select operation and returns the resulting rows, specifying a filters list, i.e. {'a': 4, 'b': None} will be translated into *WHERE A = 4 and B = NULL*.

**Return type**

int

**update**(*table*, *updates*=<function SQL.<lambda>>, *filters*=<function SQL.<lambda>>)

Executes an update operation and returns the number of affected rows, specifying a {column: value} list of updates and a filters list, i.e. {'a': 4, 'b': None} will be translated into *WHERE A = 4 and B = NULL*.

**Return type**

int

## 1.1.4 simpler.files module

**simpler.files.already\_running**(*path\_pidfile*='pid.txt')

Uses a PID file to check if an instance of this script is already running. If it's not running, it will create a PID file (a file with the PID of the current process).

**Return type**

bool

**simpler.files.clear\_global\_mem\_cache**(*global\_name*=None)

Clears the global memory cache.

**simpler.files.cwd**()

**Return type**

None

**simpler.files.decompress**(*input\_file*, *output\_dir*=None, *format*='auto')

Decompress the given file to the output directory regardless of its format.

**Return type**

None

**simpler.files.detect\_format**(*path*, *format*, *accept*=None, *default*=None)

Detects the format of a file from its path.

**Return type**

Optional[str]

**simpler.files.directory\_compare**(*old*, *new*, *kind*='dir', *ignored*=('.class', '.metadata', '.recommenders', '.pyc', '.git', '.svn', '.cached', '\_\_pycache\_\_'))

Compares the files in two directories (old and new) to detect files that have been created, deleted, changed or updated, ignoring the specified files.

**Return type**

None

`simpler.files.disk_cache(method=None, *, seconds=None, directory='.cached', identifier=None)`

The first time the decorated method is called, its result is stored as a pickle file, the next call loads the cached result from the disk. The cached files are used indefinitely unless the *seconds* lifespan is defined. The cached files are stored at *.cached* unless otherwise specified with the *directory* argument. The cache file identifier is generated from the method name plus its arguments, unless otherwise specified with the *identifier* argument.

`simpler.files.find_hidden_compressed(path, byte_limit=None)`

Recursively looks for compressed file signatures in a file.

**Return type**

Set[str]

`simpler.files.import_from_path(path, name, module_name='.')`

Loads the script at the specified path and returns an object given its name.

**Return type**

Any

`simpler.files.load(path, format='auto', encoding='utf-8', inner_args=None, inner_kwargs=None)`

Load a file in a given format.

**Return type**

object

`simpler.files.mem_cache(method=None, *, key=None, maxsize=None, is_global=False, global_name=None)`

Decorator to cache the output of a method. It is indexed by its arguments unless the *key* argument is specified, in which case *key(\*args, \*\*kwargs)* will be called to get the indexing key. If *maxsize* is defined, it is bounded as an LRU cache with *maxsize* elements at most. If *is\_global* is defined, the cache will be stored globally, so that it can be shared accross multiple methods of multiple instances of a class. A *global\_name* can be defined to identify the method; otherwise, the method name will be used.

`simpler.files.register_protocol_handler(protocol, path, link_name=None, content_type=None)`

Register a protocol handler in Windows, so that <protocol>:<address> links call command <path> <address>. I.e, *register\_protocol\_handler('magnet', 'C:/Program Files/qBittorrent/qbittorrent.exe', link\_name='URL:Magnet link', content\_type='application/x-magnet')*.

**Return type**

None

`simpler.files.run_notebook(path)`

Runs a notebook and returns the result.

**Return type**

None

`simpler.files.save(path, content, format='auto', encoding='utf-8', append=False, inner_args=None, inner_kwargs=None)`

Saves a file to the given format.

**Return type**

None

`simpler.files.size(file)`

A way to see the size of a file without loading it to memory.

**Return type**

int

`simpler.files.tvshow_rename(path)`

Rename every TV show of a folder. I.e. `Inception_Season_4_Episode_02_DivX-Total.mkv` would be `04x02.mkv`.

**Return type**

None

### 1.1.5 simpler.format module

`simpler.format.human_bytes(size, decimal_places=2)`

Returns a human readable file size from a number of bytes.

**Return type**

str

`simpler.format.human_date(date)`

Return a date the a human-friendly format “1 month ago”.

**Return type**

str

`simpler.format.human_seconds(seconds)`

Returns a human readable string from a number of seconds.

**Return type**

str

`simpler.format.print_matrix(matrix, rows=None, cols=None, elem_width=None, separator='')`

**Return type**

str

`simpler.format.random_string(length, mask=None)`

Returns a random string.

**Return type**

str

`simpler.format.safe_filename(filename)`

**Return type**

str

### 1.1.6 simpler.mail module

`simpler.mail.compose(from_mail, dest_mail, from_name=None, dest_name=None, text="", text_type='plain',  
subject="", blocking=False)`

`simpler.mail.send(smtp_server, password, mail, message)`

Sends a message.

**Return type**

None

### 1.1.7 simpler.math module

`simpler.math.all_equal(seq)`

Returns true if every element in the sequence has the same value.

**Return type**

list

`simpler.math.base_change(n, base_from, base_to)`

Changes the base of n represented as a list of integers. Example: `base_change([1, 1, 0, 1], 2, 10) = [1, 3]`

**Return type**

list

`simpler.math.clamp(value, smallest=0, largest=1)`

Returns the value clamped between smallest and largest. I.e.: `clamp(10, 2, 8)` would return 8.

**Return type**

float

`simpler.math.date_range(date_start, date_end, step=None)`

`simpler.math.factor(n)`

Returns the factors of n and its exponents.

**Return type**

list

`simpler.math.fibonacci(n)`

Returns the n-th Fibonacci number.

**Return type**

int

`simpler.math.gcd(a, b)`

Greatest common divisor of two numbers.

**Return type**

int

`simpler.math.is_prime(n)`

Checks if a number is prime.

**Return type**

bool

`simpler.math.jaccard(seq1, seq2)`

Returns the Jaccard index of two sequences.

**Return type**

list

`simpler.math.lcm(a, b)`

Least common multiple of two numbers.

**Return type**

int

`simpler.math.levenshtein(seq1, seq2)`

Returns the Levenshtein distance of two sequences.

**Return type**

list

`simpler.math.palindrome_list(k)`

Returns a list of every palindromic number with k digits.

**Return type**

list

`simpler.math.phi(n)`

Returns the Euler's phi function of n.

**Return type**

int

`simpler.math.prime_list(n)`

Returns the list of prime numbers from 2 to n.

**Return type**

list

`simpler.math.snap(value, step=1, offset=0)`

Returns the value snapped to a scale of size step with an optional offset. I.e.: `snap(3.1, 2, 0)` would return 4.

**Return type**

float

`simpler.math.unique(seq, uniqueness_function)`

Returns a list in the same order with just the elements with a unique value on the uniqueness\_function. I.e.: `unique([1, 5, 7, 9], lambda x: x % 3)` would return [1, 5, 9].

**Return type**

list

### 1.1.8 simpler.profiling module

`simpler.profiling.deep_size(obj)`

Get the actual size of an instance, exploring all its references.

`simpler.profiling.tic()`

Captures time

**Return type**

None

`simpler.profiling.toc(show=True, show_label="")`

Shows time since tic() was executed.

**Return type**

float

### 1.1.9 `simpler.sparql` module

`simpler.sparql.dbpedia(query)`

Sends a query to DBPedia and return the results.

`simpler.sparql.entity_types(value)`

Return every entity type with values that contain a given string sorted by frequency.

**Return type**

List[Tuple[str, int]]

### 1.1.10 `simpler.terminal` module

`simpler.terminal.cprint(*args, fg='default', bg='default', **kwargs)`

Same syntax as `print`, with two optional parameters `fg` and `bg` to change the print color. Available colors are: default, black, red, green, yellow, blue, magenta, cyan, light\_gray, dark\_gray, light\_red, light\_green, light\_yellow, light\_blue, light\_magenta, light\_cyan, and white.

`simpler.terminal.getch()`

Reads a single byte from the user input.

### 1.1.11 `simpler.tests` module

`class simpler.tests.Suite(*tests)`

Bases: object

Class for running a test suite. Is built as `Suite(FirstTest, SecondTest...)` where the arguments are an enumeration of subclasses of `simpler.Test` classes that will be run when using this class `run` method. There are a few `run_<format>` methods to get a formatted output.

**run()**

Runs the tests and returns a dictionary of tests, where each test is a dictionary of *case: error* pairs.

**Return type**

Tuple[Dict[str, Tuple[Dict[str, Tuple[Optional[str], float]], float, int, int]], float, int, int]

**run\_html(only\_errors=True)**

Runs the `run` method and returns a table of errors, or None if there isn't any.

**Return type**

Optional[str]

**run\_text(only\_errors=True)**

Runs the `run` method and formats the output as a plain text.

**Return type**

Optional[str]

`class simpler.tests.Test`

Bases: object

Class for a test case. Each test case might contain multiple `test_<something>` methods which are called when running the tests. It is advisable to run it from the `simpler.Suite` class.

```
PREFIX = 'test_'
```

```
run()
```

Internal method used to run the tests.

**Return type**

Dict[str, Tuple[Optional[str], float]]

### 1.1.12 simpler.validation module

```
simpler.validation.assert_exists(path)
```

Asserts that the given path exists within PATH\_STATIC.

**Return type**

None

```
simpler.validation.assert_id(data, name, optional=False, allow_zero=False, default=None)
```

Asserts that data[name] is a valid database id and returns it.

**Return type**

int

```
simpler.validation.assert_mail(data, name, optional=False, default=None)
```

Asserts that data[name] is a valid mail string and returns it.

**Return type**

str

```
simpler.validation.assert_number(data, name, optional=False, min_val=None, max_val=None,
                                  is_integer=None, default=None)
```

Asserts all the requested numeric checks to data[name] and returns it.

**Return type**

int

```
simpler.validation.assert_set(data, name, optional=False, default=None)
```

Asserts that data[name] exists and returns it.

```
simpler.validation.assert_str(data, name, optional=False, min_len=None, max_len=None,
                               has_letter=None, has_number=None, has_symbol=None,
                               has_whitespace=None, has_pattern=None, default=None)
```

Asserts all the requested checks to data[name] and returns it.

**Return type**

str

### 1.1.13 simpler.web module

```
class simpler.web.DownloaderPool(num_workers=100, download_method=None)
```

Bases: object

```
download_worker()
```

```
get(urls)
```

```
spawn_workers()
```

```
class simpler.web.Driver(timeout=3, keystroke_delay=0.005, headless=True, disable_flash=True,  
                        disable_images=True, language='en-US, en', options=None)  
  
    Bases: object  
  
    all_cookies(clear=True, path=None, domain=None, http_only=None, secure=None)  
  
        Return type  
        dict  
  
    all_local_storage(clear=True)  
  
        Return type  
        dict  
  
    all_session_storage(clear=True)  
  
        Return type  
        dict  
  
    attribute(element, attribute, value=None)  
  
        Return type  
        Optional[str]  
  
    box(element)  
  
        Return type  
        Tuple[float, float]  
  
    browse(path)  
  
    click(element)  
  
        Return type  
        None  
  
    console_clear()  
  
    console_messages(group_by_level=False)  
  
        Return type  
        Dict[str, str]  
  
    cookie(name, value=None, expiry=None, delete=False, path=None, domain=None, http_only=None,  
          secure=None)  
  
        Return type  
        Optional[str]  
  
    drag(element, x_offset=0, y_offset=0)  
  
        Return type  
        None  
  
    focus(element)  
  
        Return type  
        None  
  
    has_class(element, class_name)  
  
        Return type  
        bool
```



**hover**(*element*)

**Return type**

None

**local\_storage**(*key*, *value=None*, *delete=False*)

**Return type**

Optional[str]

**press**(*text*)

**scroll**(*element*, *x\_delta=0*, *y\_delta=0*, *mouse\_x=0*, *mouse\_y=0*)

**Return type**

None

**scroll\_into\_view**(*element*)

**select**(*element*, *wait=True*, *all=False*, *raise\_errors=None*)

**session\_storage**(*key*, *value=None*, *delete=False*)

**Return type**

Optional[str]

**style**(*element*, *property*, *value=None*)

**Return type**

Optional[str]

**translate**(*char*)

**Return type**

str

**wait**(*element*, *message=None*, *raise\_errors=True*, *invert=False*)

**Return type**

bool

**wait\_for\_file**(*path*, *message='Timeout waiting for file: '*, *raise\_errors=True*)

**Return type**

bool

**wait\_for\_url**(*url*, *message=None*, *raise\_errors=True*, *invert=False*)

**Return type**

bool

**write**(*element*, *text*, *clear=False*)

**Return type**

None

**simpler.web.download\_file**(*url*, *path=None*, *chunk\_size=100000*, *show\_progress=True*)

Downloads a file keeping track of the progress. Returns the output path.

**Return type**

str

`simpler.web.throttle(seconds=1)`

Sleeps the thread so that the function is called every X seconds.

**Return type**

None

## 1.2 Module contents

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### S

- `simpler`, 14
- `simpler.algorithms`, 1
- `simpler.bioinformatics`, 2
- `simpler.connectors`, 2
- `simpler.files`, 5
- `simpler.format`, 7
- `simpler.mail`, 7
- `simpler.math`, 8
- `simpler.profilng`, 9
- `simpler.sparql`, 10
- `simpler.terminal`, 10
- `simpler.tests`, 10
- `simpler.validation`, 11
- `simpler.web`, 11



## A

all\_cookies() (*simpler.web.Driver* method), 12  
all\_equal() (*in module simpler.math*), 8  
all\_local\_storage() (*simpler.web.Driver* method), 12  
ALL\_OPTIMAL\_SOLUTIONS (*simpler.algorithms.DynamicProgramming* attribute), 1  
all\_session\_storage() (*simpler.web.Driver* method), 12  
ALL\_SOLUTIONS (*simpler.algorithms.DynamicProgramming* attribute), 1  
already\_running() (*in module simpler.files*), 5  
alternatives() (*simpler.algorithms.DynamicProgramming* method), 1  
apply() (*simpler.connectors.SQL* method), 3  
assert\_exists() (*in module simpler.validation*), 11  
assert\_id() (*in module simpler.validation*), 11  
assert\_mail() (*in module simpler.validation*), 11  
assert\_number() (*in module simpler.validation*), 11  
assert\_set() (*in module simpler.validation*), 11  
assert\_str() (*in module simpler.validation*), 11  
attribute() (*simpler.web.Driver* method), 12

## B

base\_change() (*in module simpler.math*), 8  
block\_from\_code() (*simpler.connectors.Excel* method), 2  
box() (*simpler.web.Driver* method), 12  
browse() (*simpler.web.Driver* method), 12

## C

cell() (*simpler.connectors.Excel* method), 2  
cells() (*simpler.connectors.Excel* method), 2  
clamp() (*in module simpler.math*), 8  
clear\_global\_mem\_cache() (*in module simpler.files*), 5  
click() (*simpler.web.Driver* method), 12  
close() (*simpler.connectors.SQL* method), 3  
compose() (*in module simpler.mail*), 7  
console\_clear() (*simpler.web.Driver* method), 12

console\_messages() (*simpler.web.Driver* method), 12  
cookie() (*simpler.web.Driver* method), 12  
cprint() (*in module simpler.terminal*), 10  
cursor() (*simpler.connectors.SQL* method), 3  
cwd() (*in module simpler.files*), 5

## D

date\_range() (*in module simpler.math*), 8  
dbpedia() (*in module simpler.sparql*), 10  
decompress() (*in module simpler.files*), 5  
deep\_merge() (*in module simpler.algorithms*), 1  
deep\_size() (*in module simpler.profilng*), 9  
delete() (*simpler.connectors.SQL* method), 3  
detect\_format() (*in module simpler.files*), 5  
directory\_compare() (*in module simpler.files*), 5  
disk\_cache() (*in module simpler.files*), 6  
dna\_to\_rna() (*in module simpler.bioinformatics*), 2  
download\_file() (*in module simpler.web*), 13  
download\_worker() (*simpler.web.DownloaderPool* method), 11  
DownloaderPool (*class in simpler.web*), 11  
drag() (*simpler.web.Driver* method), 12  
Driver (*class in simpler.web*), 11  
DynamicProgramming (*class in simpler.algorithms*), 1

## E

ENGINES (*simpler.connectors.SQL* attribute), 3  
entity\_types() (*in module simpler.sparql*), 10  
escape() (*simpler.connectors.SQL* method), 3  
Excel (*class in simpler.connectors*), 2  
execute() (*simpler.connectors.SQL* method), 3  
exists() (*simpler.connectors.SQL* method), 3

## F

factor() (*in module simpler.math*), 8  
fibonacci() (*in module simpler.math*), 8  
find() (*simpler.connectors.SQL* method), 3  
find\_all() (*simpler.connectors.SQL* method), 4  
find\_all\_tuples() (*simpler.connectors.SQL* method), 4  
find\_column() (*simpler.connectors.SQL* method), 4

`find_hidden_compressed()` (in module `simpler.files`),  
6  
`find_tuple()` (`simpler.connectors.SQL` method), 4  
`find_value()` (`simpler.connectors.SQL` method), 4  
`focus()` (`simpler.web.Driver` method), 12

## G

`gcd()` (in module `simpler.math`), 8  
`get()` (`simpler.web.DownloaderPool` method), 11  
`getch()` (in module `simpler.terminal`), 10

## H

`has_class()` (`simpler.web.Driver` method), 12  
`hover()` (`simpler.web.Driver` method), 12  
`human_bytes()` (in module `simpler.format`), 7  
`human_date()` (in module `simpler.format`), 7  
`human_seconds()` (in module `simpler.format`), 7

## I

`import_from_path()` (in module `simpler.files`), 6  
`index_from_code()` (`simpler.connectors.Excel`  
method), 2  
`insert()` (`simpler.connectors.SQL` method), 4  
`insert_all()` (`simpler.connectors.SQL` method), 4  
`is_final()` (`simpler.algorithms.DynamicProgramming`  
method), 1  
`is_prime()` (in module `simpler.math`), 8  
`iter_all()` (`simpler.connectors.SQL` method), 4  
`iter_all_tuples()` (`simpler.connectors.SQL` method),  
4  
`iter_column()` (`simpler.connectors.SQL` method), 4

## J

`jaccard()` (in module `simpler.math`), 8

## L

`lcm()` (in module `simpler.math`), 8  
`levenshtein()` (in module `simpler.math`), 8  
`load()` (in module `simpler.files`), 6  
`local_storage()` (`simpler.web.Driver` method), 13

## M

`mem_cache()` (in module `simpler.files`), 6  
module  
    `simpler`, 14  
    `simpler.algorithms`, 1  
    `simpler.bioinformatics`, 2  
    `simpler.connectors`, 2  
    `simpler.files`, 5  
    `simpler.format`, 7  
    `simpler.mail`, 7  
    `simpler.math`, 8  
    `simpler.profiling`, 9

`simpler.sparql`, 10  
`simpler.terminal`, 10  
`simpler.tests`, 10  
`simpler.validation`, 11  
`simpler.web`, 11

## O

`ONE_OPTIMAL_SOLUTION` (`simpler.algorithms.DynamicProgramming`  
attribute), 1  
`ONE_SOLUTION` (`simpler.algorithms.DynamicProgramming`  
attribute), 1

## P

`palindrome_list()` (in module `simpler.math`), 9  
`parse_fasta()` (in module `simpler.bioinformatics`), 2  
`penalty()` (`simpler.algorithms.DynamicProgramming`  
method), 1  
`phi()` (in module `simpler.math`), 9  
`PREFIX` (`simpler.tests.Test` attribute), 10  
`press()` (`simpler.web.Driver` method), 13  
`prime_list()` (in module `simpler.math`), 9  
`print_matrix()` (in module `simpler.format`), 7  
`print_query()` (`simpler.connectors.SQL` method), 5

## R

`random_string()` (in module `simpler.format`), 7  
`register_protocol_handler()` (in module `simpler.files`), 6  
`reverse_complement()` (in module `simpler.bioinformatics`), 2  
`rna_to_dna()` (in module `simpler.bioinformatics`), 2  
`rna_to_protein()` (in module `simpler.bioinformatics`),  
2  
`run()` (`simpler.tests.Suite` method), 10  
`run()` (`simpler.tests.Test` method), 11  
`run_html()` (`simpler.tests.Suite` method), 10  
`run_notebook()` (in module `simpler.files`), 6  
`run_text()` (`simpler.tests.Suite` method), 10

## S

`safe_filename()` (in module `simpler.format`), 7  
`save()` (in module `simpler.files`), 6  
`scroll()` (`simpler.web.Driver` method), 13  
`scroll_into_view()` (`simpler.web.Driver` method), 13  
`select()` (`simpler.connectors.SQL` method), 5  
`select()` (`simpler.web.Driver` method), 13  
`send()` (in module `simpler.mail`), 7  
`session_storage()` (`simpler.web.Driver` method), 13  
`sheet()` (`simpler.connectors.Excel` method), 3  
`simpler`  
    module, 14  
`simpler.algorithms`



- module, 1
- simpler.bioinformatics
  - module, 2
- simpler.connectors
  - module, 2
- simpler.files
  - module, 5
- simpler.format
  - module, 7
- simpler.mail
  - module, 7
- simpler.math
  - module, 8
- simpler.profiling
  - module, 9
- simpler.sparql
  - module, 10
- simpler.terminal
  - module, 10
- simpler.tests
  - module, 10
- simpler.validation
  - module, 11
- simpler.web
  - module, 11
- size() (*in module simpler.files*), 6
- snap() (*in module simpler.math*), 9
- solve() (*simpler.algorithms.DynamicProgramming method*), 1
- spawn\_workers() (*simpler.web.DownloaderPool method*), 11
- SQL (*class in simpler.connectors*), 3
- style() (*simpler.web.Driver method*), 13
- Suite (*class in simpler.tests*), 10

## T

- table() (*simpler.connectors.Excel method*), 3
- Test (*class in simpler.tests*), 10
- throttle() (*in module simpler.web*), 13
- tic() (*in module simpler.profiling*), 9
- toc() (*in module simpler.profiling*), 9
- translate() (*simpler.web.Driver method*), 13
- tvshow\_rename() (*in module simpler.files*), 6

## U

- unique() (*in module simpler.math*), 9
- update() (*simpler.connectors.SQL method*), 5

## W

- wait() (*simpler.web.Driver method*), 13
- wait\_for\_file() (*simpler.web.Driver method*), 13
- wait\_for\_url() (*simpler.web.Driver method*), 13
- write() (*simpler.web.Driver method*), 13