Unittest for `caching`

August 15, 2025

Contents

1 Test Information

1.1 Test Candidate Information

The Module caching is designed to store information in json or pickle files to support them much faster then generating them from the original source file. For more Information read the documentation.

Library Information	
Name	caching
State	Released
Supported Interpreters	python3
Version	52c295e7e5e9060dd96adbed34253518
Dependencies	

1.2 Unittest Information

Unittest Information	
Version	17bb378e039385c5fbedba201e1a6df9
Testruns with	python 3.13.5 (final)

1.3 Test System Information

System Information	
Architecture	64bit
Distribution	Debian GNU/Linux 13 trixie
Hostname	ahorn
Kernel	6.12.38+deb13-amd64 (#1 SMP PREEMPT_DYNAMIC Debian 6.12.38-1 (2025-07-16))
Machine	x86_64
Path	/home/dirk/work/unittest_collection/caching
System	Linux
Username	dirk

2 Statistic

2.1 Test-Statistic for testrun with python 3.13.5 (final)

Number of tests	**15**
Number of successfull tests	**15**
Number of possibly failed tests	**0**
Number of failed tests	**0**
Executionlevel	Full Test (all defined tests)
Time consumption	8.090s

2.2 Coverage Statistic

Module- or Filename	Line-Coverage	Branch-Coverage
caching	97.3%	95.8%
caching.__init__.py	97.3%	

3 Tested Requirements

3.1 Cache generation (json /pickle)

3.1.1 Data generation from source instance, if no cache is available

Description
If the cache is not available, the data shall be generated from the source instance.

Reason for the implementation
There shall be the posibility to create the cache on demand, so the fallback is to generate the data from the source instance.

Fitcriterion
Caching is called without previous cache generation and the data from the source instance is completely available.

Testresult
This test was passed with the state: **Success**. See also full trace in section A.1.1!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/caching/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 19:13:27,542
Finished-Time:	2025-08-15 19:13:27,545
Time-Consumption	0.003s

Testsummary:

Info	Prepare: Cleanup before testcase execution
Info	Prepare: First usage of 'property_cache_json' with a class holding the data to be cached
Success	Data from cached instance with key=str is correct (Content '__string__' and Type is <class 'str'>).
Success	Data from cached instance with key=unicode is correct (Content '__unicode__' and Type is <class 'str'>).
Success	Data from cached instance with key=integer is correct (Content 34 and Type is <class 'int'>).
Success	Data from cached instance with key=float is correct (Content 2.71828 and Type is <class 'float'>).
Success	Data from cached instance with key=list is correct (Content ['one', 2, 3, '4'] and Type is <class 'list'>).
Success	Data from cached instance with key=dict is correct (Content {'1': '1', '2': 2, '3': 'three', '4': '4'} and Type is <class 'dict'>).
Success	Data from cached instance with key=none is correct (Content 'not None' and Type is <class 'str'>).
Success	Data from cached instance with key=unknown_key is correct (Content 5 and Type is <class 'int'>).

3.1.2 Create complete cache from the given data instance

Description

There shall be a method caching all information from the given instance.

Reason for the implementation

Independent usage of data generation and data usage (e.g. the user requesting the data is not able to create the data).

Fitcriterion

Caching is called twice with different data instances and the cached data from the first call is completely available.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.2!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/caching/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 19:13:27,545
Finished-Time:	2025-08-15 19:13:27,548
Time-Consumption	0.003s

Testsummary:

Info	Prepare: Cleanup before testcase execution
Info	Prepare: First usage of 'property_cache_pickle' with a class holding the data to be cached
Success	Data from cached instance with key=str is correct (Content 'string' and Type is <class 'str'>).
Success	Data from cached instance with key=unicode is correct (Content 'unicode' and Type is <class 'str'>).
Success	Data from cached instance with key=integer is correct (Content 17 and Type is <class 'int'>).
Success	Data from cached instance with key=float is correct (Content 3.14159 and Type is <class 'float'>).
Success	Data from cached instance with key=list is correct (Content [1, 'two', '3', 4] and Type is <class 'list'>).
Success	Data from cached instance with key=dict is correct (Content {'1': 1, '2': 'two', '3': '3', '4': 4} and Type is <class 'dict'>).
Success	Data from cached instance with key=none is correct (Content None and Type is <class 'NoneType'>).
Success	Data from cached instance with key=unknown_key is correct (Content 5 and Type is <class 'int'>).

3.1.3 Create cache partially from a given data instance by get method

Description

On getting data from the cached instance, the information shall be stored in the cache file.

Reason for the implementation

There shall be the posibility to create the cache on demand, so the fallback is to generate the data from the source instance.

Fitcriterion
Caching is called twice with different data instances and the cached data from the first call is available for all keys cached on the first run.

Testresult
This test was passed with the state: **Success**. See also full trace in section A.1.3!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/caching/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 19:13:27,548
Finished-Time:	2025-08-15 19:13:27,554
Time-Consumption	0.006s

Testsummary:

Info	Prepare: Cleanup before testcase execution
Info	Prepare: First usage of 'property_cache_json' with a class holding the data to be cached
Success	Data from cached instance with key=str is correct (Content 'string' and Type is <class 'str'>).
Success	Data from cached instance with key=unicode is correct (Content '__unicode__' and Type is <class 'str'>).
Success	Data from cached instance with key=integer is correct (Content 17 and Type is <class 'int'>).
Success	Data from cached instance with key=float is correct (Content 2.71828 and Type is <class 'float'>).
Success	Data from cached instance with key=list is correct (Content [1, 'two', '3', 4] and Type is <class 'list'>).
Success	Data from cached instance with key=dict is correct (Content {'1': 1, '2': 'two', '3': '3', '4': 4} and Type is <class 'dict'>).
Success	Data from cached instance with key=none is correct (Content None and Type is <class 'NoneType'>).
Success	Data from cached instance with key=unknown_key is correct (Content 5 and Type is <class 'int'>).

3.1.4 Ignore corrupt cache file

Description
Ignore corrupt cachefile, while loading cache.

Reason for the implementation
Suppress exceptions while caching.

Fitcriterion
Loading cache results in no exception, when cache file is empty.

Testresult
This test was passed with the state: **Success**. See also full trace in section A.1.4!

Testrun:	python 3.13.5 (final)

Caller:	/home/dirk/work/unittest_collection/caching/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 19:13:27,555
Finished-Time:	2025-08-15 19:13:27,558
Time-Consumption	0.003s

Testsummary:

Info	Prepare: Cleanup before testcase execution
Info	Creating empty cache file /home/dirk/work/unittest_collection/caching/unittest/output_data/cache_data_test_corrupt_cache.json.
Success	Empty cache file ignored on loading cache.

3.2 Load spreading for full update

3.2.1 Full update with delay between each data generation for the cache

Description

The full update method shall pause for a given time between every cached item.

Reason for the implementation

Load spreading in case of cyclic called `.full_update()`.

Fitcriterion

The time consumption of the method `.full_update(<sleep_time>)` shall consume n times the given `sleep_time`. Where n is the number of items which will be cahed from the source instance.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.5!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/caching/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 19:13:27,558
Finished-Time:	2025-08-15 19:13:33,566
Time-Consumption	6.008s

Testsummary:

Info	Prepare: Cleanup before testcase execution
Success	Consumed time for full_update is greater expectation (Content 6.006384372711182 and Type is <class 'float'>).
Success	Consumed time for full_update is greater expectation (Content 6.006384372711182 and Type is <class 'float'>).

3.2.2 No cache generation if disabled

Description

The cache shall be generated by the `.get()` method, only if the cache instance parameter `store_on_get` is set to `True`.

Reason for the implementation
Independent usage of data generation and data usage (e.g. the user requesting the data is not able to create the data).

Fitcriterion
Create a caching instance with `store_on_get` set to `False`. Get every item of the source instance with the `.get()` method and check that no cache file exists.

Testresult
This test was passed with the state: **Success**. See also full trace in section A.1.6!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/caching/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 19:13:33,566
Finished-Time:	2025-08-15 19:13:33,570
Time-Consumption	0.004s

Testsummary:

Info	Prepare: Cleanup before testcase execution
Success	Data from cached instance with key=str is correct (Content 'string' and Type is <class 'str'>).
Success	Data from cached instance with key=unicode is correct (Content 'unicode' and Type is <class 'str'>).
Success	Data from cached instance with key=integer is correct (Content 17 and Type is <class 'int'>).
Success	Data from cached instance with key=float is correct (Content 3.14159 and Type is <class 'float'>).
Success	Data from cached instance with key=list is correct (Content [1, 'two', '3', 4] and Type is <class 'list'>).
Success	Data from cached instance with key=dict is correct (Content {'1': 1, '2': 'two', '3': '3', '4': 4} and Type is <class 'dict'>).
Success	Data from cached instance with key=none is correct (Content None and Type is <class 'NoneType'>).
Success	The cache file (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_data_test_full_update_sleep.json) shall not exist is correct (Content False and Type is <class 'bool'>).

3.3 Dump cache conditions

3.3.1 Dump cache if time is expired

Description
Dump the cached item, if this item is older then the given expirery time.

Reason for the implementation
Ensure, that the cache is updated from time to time. For example for items which do not change very often.

Fitcriterion
Create a cache instance, cache some data. Intialise a second caching instance with a different source instance and a

expire time. Wait for longer than the given expiry time and check that the items from the second source instance are returned.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.7!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/caching/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 19:13:33,571
Finished-Time:	2025-08-15 19:13:35,588
Time-Consumption	2.017s

Testsummary:

Info	Prepare: Cleanup before testcase execution
Info	Prepare: First usage of 'property_cache_json' with a class holding the data to be cached
Success	Data from cached instance with key=str is correct (Content 'string' and Type is <class 'str'>).
Success	Data from cached instance with key=unicode is correct (Content 'unicode' and Type is <class 'str'>).
Success	Data from cached instance with key=integer is correct (Content 17 and Type is <class 'int'>).
Success	Data from cached instance with key=float is correct (Content 3.14159 and Type is <class 'float'>).
Success	Data from cached instance with key=list is correct (Content [1, 'two', '3', 4] and Type is <class 'list'>).
Success	Data from cached instance with key=dict is correct (Content {'1': 1, '2': 'two', '3': '3', '4': 4} and Type is <class 'dict'>).
Success	Data from cached instance with key=none is correct (Content None and Type is <class 'NoneType'>).
Success	Data from cached instance with key=str is correct (Content '__string__' and Type is <class 'str'>).
Success	Data from cached instance with key=unicode is correct (Content '__unicode__' and Type is <class 'str'>).
Success	Data from cached instance with key=integer is correct (Content 34 and Type is <class 'int'>).
Success	Data from cached instance with key=float is correct (Content 2.71828 and Type is <class 'float'>).
Success	Data from cached instance with key=list is correct (Content ['one', 2, 3, '4'] and Type is <class 'list'>).
Success	Data from cached instance with key=dict is correct (Content {'1': '1', '2': 2, '3': 'three', '4': '4'} and Type is <class 'dict'>).
Success	Data from cached instance with key=none is correct (Content 'not None' and Type is <class 'str'>).

3.3.2 Dump cache if data version increases

Description

Dump the complete cache, if the *data version* of the source instance is increased.

Reason for the implementation

The data version is part of the source instance. Increasing the data version indicates, that the source instance generates

the data in another way or the structure of the data is changed. In that condition, the cache needs to be ignored.

Fitcriterion
Create a cached instance and cache some items. Generate a second cached instance with different source data and a increased data version. Ensure, that the cache instance returns the values from the second source. It is required to set `load_all_on_init` to `False` and `store_on_get` to `True`.

Testresult
This test was passed with the state: **Success**. See also full trace in section A.1.8!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/caching/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 19:13:35,588
Finished-Time:	2025-08-15 19:13:35,596
Time-Consumption	0.008s

Testsummary:

Info	Prepare: Cleanup before testcase execution
Info	Prepare: First usage of 'property_cache_json' with a class holding the data to be cached
Success	Data from cached instance with key=str is correct (Content '__string__' and Type is <class 'str'>).
Success	Data from cached instance with key=unicode is correct (Content '__unicode__' and Type is <class 'str'>).
Success	Data from cached instance with key=integer is correct (Content 34 and Type is <class 'int'>).
Success	Data from cached instance with key=float is correct (Content 2.71828 and Type is <class 'float'>).
Success	Data from cached instance with key=list is correct (Content ['one', 2, 3, '4'] and Type is <class 'list'>).
Success	Data from cached instance with key=dict is correct (Content {'1': '1', '2': 2, '3': 'three', '4': '4'} and Type is <class 'dict'>).
Success	Data from cached instance with key=none is correct (Content 'not None' and Type is <class 'str'>).

3.3.3 Dump cache if data uid is changed

Description
Dump the complete cache, if the *data uid* of the source instance is changed.

Reason for the implementation
The data uid is part of the source instance. Changing the data uid indicates, that the source of the data created by the source instance is changed (e.g. the uid of a file or folder) and the cache needs to be ignored.

Fitcriterion
Create a cached instance and cache some items. Generate a second cached instance with different source data and a changed data uid. Ensure, that the cache instance returns the values from the second source. It is required to set `load_all_on_init` to `False` and `store_on_get` to `True`.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.9!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/caching/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 19:13:35,596
Finished-Time:	2025-08-15 19:13:35,606
Time-Consumption	0.010s

Testsummary:

Info	Prepare: Cleanup before testcase execution
Info	Prepare: First usage of 'property_cache_json' with a class holding the data to be cached
Success	Data from cached instance with key=str is correct (Content '__string__' and Type is <class 'str'>).
Success	Data from cached instance with key=unicode is correct (Content '__unicode__' and Type is <class 'str'>).
Success	Data from cached instance with key=integer is correct (Content 34 and Type is <class 'int'>).
Success	Data from cached instance with key=float is correct (Content 2.71828 and Type is <class 'float'>).
Success	Data from cached instance with key=list is correct (Content ['one', 2, 3, '4'] and Type is <class 'list'>).
Success	Data from cached instance with key=dict is correct (Content {'1': '1', '2': 2, '3': 'three', '4': '4'} and Type is <class 'dict'>).
Success	Data from cached instance with key=none is correct (Content 'not None' and Type is <class 'str'>).

3.3.4 Dump cache if storage version is changed

Description

Dump the complete cache, if the *storage version* of the caching class is changed.

Reason for the implementation

The storage version is part of the caching class. Changing the storage version indicates, that the previously stored cache is not compatible due to new data storage and the cache needs to be ignored.

Fitcriterion

Create a cached instance and cache some items. Generate a second cached instance with different source data and a changed storage version. Ensure, that the cache instance returns the values from the second source. It is required to set `load_all_on_init` to `False` and `store_on_get` to `True`.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.10!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/caching/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 19:13:35,606
Finished-Time:	2025-08-15 19:13:35,620

Time-Consumption	0.014s

Testsummary:

Info	Prepare: Cleanup before testcase execution
Info	Prepare: First usage of 'property_cache_json' with a class holding the data to be cached
Success	Data from cached instance with key=str is correct (Content '__string__' and Type is <class 'str'>).
Success	Data from cached instance with key=unicode is correct (Content '__unicode__' and Type is <class 'str'>).
Success	Data from cached instance with key=integer is correct (Content 34 and Type is <class 'int'>).
Success	Data from cached instance with key=float is correct (Content 2.71828 and Type is <class 'float'>).
Success	Data from cached instance with key=list is correct (Content ['one', 2, 3, '4'] and Type is <class 'list'>).
Success	Data from cached instance with key=dict is correct (Content {'1': '1', '2': 2, '3': 'three', '4': '4'} and Type is <class 'dict'>).
Success	Data from cached instance with key=none is correct (Content 'not None' and Type is <class 'str'>).

3.3.5 Dump cache if stored value is 'None'

Description

Dump the cached item, if the stored value is `None`.

Reason for the implementation

If no information is stored in the cache, the data shall be generated by the source instance.

Fitcriterion

Create a cached instance and cache some items. One needs to have `None` as value. Generate a second cached instance with different source data (especially, the previous item with value `None` needs to have a not `None` value. Ensure, that the caching instance returns not `None` from the second source.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.11!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/caching/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 19:13:35,621
Finished-Time:	2025-08-15 19:13:35,626
Time-Consumption	0.005s

Testsummary:

Info	Prepare: Cleanup before testcase execution
Info	Prepare: First usage of 'property_cache_json' with a class holding the data to be cached
Success	Data from cached instance with key=str is correct (Content 'string' and Type is <class 'str'>).
Success	Data from cached instance with key=unicode is correct (Content 'unicode' and Type is <class 'str'>).

Success	Data from cached instance with key=integer is correct (Content 17 and Type is <class 'int'>).
Success	Data from cached instance with key=float is correct (Content 3.14159 and Type is <class 'float'>).
Success	Data from cached instance with key=list is correct (Content [1, 'two', '3', 4] and Type is <class 'list'>).
Success	Data from cached instance with key=dict is correct (Content {'1': 1, '2': 'two', '3': '3', '4': 4} and Type is <class 'dict'>).
Success	Data from cached instance with key=none is correct (Content 'not None' and Type is <class 'str'>).

3.4 Definition of uncached data

3.4.1 Define uncached data

Description
It shall be possible to define items which are not cached.

Reason for the implementation
If there is dynamic changed data in the source instance, it shall be possible to define these items as non cached to get them always from the source instance.

Fitcriterion
Create a cached instance and cache some items. Generate a second cached instance with different source data and set at least one item as source item. This item should be previously cached. Ensure, that the source item isis the one from the second source instance.

Testresult
This test was passed with the state: Success. See also full trace in section A.1.12!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/caching/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 19:13:35,626
Finished-Time:	2025-08-15 19:13:35,631
Time-Consumption	0.005s

Testsummary:

Info	Prepare: Cleanup before testcase execution
Info	Prepare: First usage of 'property_cache_json' with a class holding the data to be cached
Success	Data from cached instance with key=str is correct (Content '__string__' and Type is <class 'str'>).
Success	Data from cached instance with key=unicode is correct (Content 'unicode' and Type is <class 'str'>).
Success	Data from cached instance with key=integer is correct (Content 17 and Type is <class 'int'>).
Success	Data from cached instance with key=float is correct (Content 2.71828 and Type is <class 'float'>).
Success	Data from cached instance with key=list is correct (Content [1, 'two', '3', 4] and Type is <class 'list'>).

Success	Data from cached instance with key=dict is correct (Content {'1': 1, '2': 'two', '3': '3', '4': 4} and Type is <class 'dict'>).
Success	Data from cached instance with key=none is correct (Content None and Type is <class 'NoneType'>).

3.5 Callback on data storage

3.5.1 If no data is changed, no callback will be executed

Description

The store callback shall not be executed, if no cache is stored.

Reason for the implementation

Do actions, if cache data is stored to disk.

Fitcriterion

Initialise the cache instance without storing cache data. Ensure, that the callback is never executed.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.13!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/caching/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 19:13:35,631
Finished-Time:	2025-08-15 19:13:35,632
Time-Consumption	0.001s

Testsummary:

Info	Prepare: Cleanup before testcase execution
Info	Installing save_callback with no get or full_update execution.
Success	Save callback execution counter is correct (Content 0 and Type is <class 'int'>).
Success	Save callback execution counter is correct (Content None and Type is <class 'NoneType'>).

3.5.2 Callback execution in case of a full update

Description

The storage callback shall be called once on every `full_update()`.

Reason for the implementation

Do actions, if cache data is stored to disk.

Fitcriterion

Initialise the cache instance and ensure, that the callback is executed as often as the `.full_update()` method is executed.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.14!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/caching/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 19:13:35,632
Finished-Time:	2025-08-15 19:13:35,634
Time-Consumption	0.002s

Testsummary:

Info	Prepare: Cleanup before testcase execution
Info	Installing save_callback and execute full_update.
Success	Save callback execution counter is correct (Content 1 and Type is <class 'int'>).
Success	Save callback execution counter is correct (Content <caching.property_cache_json object at 0x7f7c48d81b30> and Type is <class 'caching.property_cache_json'>).

3.5.3 Callback execution in case of get function

Description

The storage callback, shall be called once on every `.get()`, if `storage_on_get` is set to `True`.

Reason for the implementation

Do actions, if cache data is stored to disk.

Fitcriterion

Initialise the cache instance and ensure, that the callback is executed as often as the `.get()` method is executed.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.15!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/caching/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 19:13:35,634
Finished-Time:	2025-08-15 19:13:35,636
Time-Consumption	0.002s

Testsummary:

Info	Prepare: Cleanup before testcase execution
Info	Installing save_callback and execute a single get.
Info	Installing save_callback and execute a single get.
Success	Save callback execution counter is correct (Content 2 and Type is <class 'int'>).
Success	Save callback execution counter is correct (Content <caching.property_cache_json object at 0x7f7c48d80cd0> and Type is <class 'caching.property_cache_json'>).

A Trace for testrun with python 3.13.5 (final)

A.1 Tests with status Info (15)

A.1.1 REQ-0003

Testresult

This test was passed with the state: **Success**.

Info	Prepare: Cleanup before testcase execution

```
Deleting cache file from filesystem to ensure identical conditions for each test run.
```

Info	Prepare: First usage of 'property_cache_json' with a class holding the data to be cached

Success	Data from cached instance with key=str is correct (Content '__string__' and Type is <class 'str'>).

```
Cache file does not exists (yet).
Loading property for key='str' from source instance
Result (Data from cached instance with key=str): '__string__' (<class 'str'>)
Expectation (Data from cached instance with key=str): result = '__string__' (<class 'str'>)
```

Success	Data from cached instance with key=unicode is correct (Content '__unicode__' and Type is <class 'str'>).

```
Loading property for key='unicode' from source instance
Result (Data from cached instance with key=unicode): '__unicode__' (<class 'str'>)
Expectation (Data from cached instance with key=unicode): result = '__unicode__' (<class
↪  'str'>)
```

Success	Data from cached instance with key=integer is correct (Content 34 and Type is <class 'int'>).

```
Loading property for key='integer' from source instance
Result (Data from cached instance with key=integer): 34 (<class 'int'>)
Expectation (Data from cached instance with key=integer): result = 34 (<class 'int'>)
```

Success	Data from cached instance with key=float is correct (Content 2.71828 and Type is <class 'float'>).

```
Loading property for key='float' from source instance
Result (Data from cached instance with key=float): 2.71828 (<class 'float'>)
Expectation (Data from cached instance with key=float): result = 2.71828 (<class 'float'>)
```

Success	Data from cached instance with key=list is correct (Content ['one', 2, 3, '4'] and Type is <class 'list'>).

```
Loading property for key='list' from source instance
Result (Data from cached instance with key=list): [ 'one', 2, 3, '4' ] (<class 'list'>)
```

```
Expectation (Data from cached instance with key=list): result = [ 'one', 2, 3, '4' ] (<class
↪  'list'>)
```

Success	Data from cached instance with key=dict is correct (Content {'1': '1', '2': 2, '3': 'three', '4': '4'} and Type is <class 'dict'>).

```
Loading property for key='dict' from source instance
Result (Data from cached instance with key=dict): { '1': '1', '2': 2, '3': 'three', '4': '4' }
↪  (<class 'dict'>)
Expectation (Data from cached instance with key=dict): result = { '1': '1', '2': 2, '3':
↪  'three', '4': '4' } (<class 'dict'>)
```

Success	Data from cached instance with key=none is correct (Content 'not None' and Type is <class 'str'>).

```
Loading property for key='none' from source instance
Result (Data from cached instance with key=none): 'not None' (<class 'str'>)
Expectation (Data from cached instance with key=none): result = 'not None' (<class 'str'>)
```

Success	Data from cached instance with key=unknown_key is correct (Content 5 and Type is <class 'int'>).

```
Key 'unknown_key' is not in cached_keys. Uncached data will be returned.
Result (Data from cached instance with key=unknown_key): 5 (<class 'int'>)
Expectation (Data from cached instance with key=unknown_key): result = 5 (<class 'int'>)
```

A.1.2 REQ-0001

Testresult

This test was passed with the state: **Success**.

Info	Prepare: Cleanup before testcase execution

```
Deleting cache file from filesystem to ensure identical conditions for each test run.
```

Info	Prepare: First usage of 'property_cache_pickle' with a class holding the data to be cached

```
Cache file does not exists (yet).
Loading all data from source - ['str', 'unicode', 'integer', 'float', 'list', 'dict', 'none']
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_load_on_init.pkl)
```

Success	Data from cached instance with key=str is correct (Content 'string' and Type is <class 'str'>).

```
Loading properties from cache (/home/dirk/work/unittest_collection/caching/unittest/output_da
↪  ta/cache_data_test_load_on_init.pkl)
Providing property for 'str' from cache
Result (Data from cached instance with key=str): 'string' (<class 'str'>)
```

```
Expectation (Data from cached instance with key=str): result = 'string' (<class 'str'>)
```

Success Data from cached instance with key=unicode is correct (Content 'unicode' and Type is <class 'str'>).

```
Providing property for 'unicode' from cache
Result (Data from cached instance with key=unicode): 'unicode' (<class 'str'>)
Expectation (Data from cached instance with key=unicode): result = 'unicode' (<class 'str'>)
```

Success Data from cached instance with key=integer is correct (Content 17 and Type is <class 'int'>).

```
Providing property for 'integer' from cache
Result (Data from cached instance with key=integer): 17 (<class 'int'>)
Expectation (Data from cached instance with key=integer): result = 17 (<class 'int'>)
```

Success Data from cached instance with key=float is correct (Content 3.14159 and Type is <class 'float'>).

```
Providing property for 'float' from cache
Result (Data from cached instance with key=float): 3.14159 (<class 'float'>)
Expectation (Data from cached instance with key=float): result = 3.14159 (<class 'float'>)
```

Success Data from cached instance with key=list is correct (Content [1, 'two', '3', 4] and Type is <class 'list'>).

```
Providing property for 'list' from cache
Result (Data from cached instance with key=list): [ 1, 'two', '3', 4 ] (<class 'list'>)
Expectation (Data from cached instance with key=list): result = [ 1, 'two', '3', 4 ] (<class
↪  'list'>)
```

Success Data from cached instance with key=dict is correct (Content {'1': 1, '2': 'two', '3': '3', '4': 4} and Type is <class 'dict'>).

```
Providing property for 'dict' from cache
Result (Data from cached instance with key=dict): { '1': 1, '2': 'two', '3': '3', '4': 4 }
↪  (<class 'dict'>)
Expectation (Data from cached instance with key=dict): result = { '1': 1, '2': 'two', '3':
↪  '3', '4': 4 } (<class 'dict'>)
```

Success Data from cached instance with key=none is correct (Content None and Type is <class 'NoneType'>).

```
Providing property for 'none' from cache
Result (Data from cached instance with key=none): None (<class 'NoneType'>)
Expectation (Data from cached instance with key=none): result = None (<class 'NoneType'>)
```

Success Data from cached instance with key=unknown_key is correct (Content 5 and Type is <class 'int'>).

```
Key 'unknown_key' is not in cached_keys. Uncached data will be returned.
Result (Data from cached instance with key=unknown_key): 5 (<class 'int'>)
Expectation (Data from cached instance with key=unknown_key): result = 5 (<class 'int'>)
```

A.1.3 REQ-0005

Testresult

This test was passed with the state: **Success**.

Info	Prepare: Cleanup before testcase execution

```
Cache file does not exist on filesystem.
```

Info	Prepare: First usage of 'property_cache_json' with a class holding the data to be cached

```
Cache file does not exists (yet).
Loading property for key='str' from source instance
Adding key=str, value=string with timestamp=1755278007 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪ a_test_load_on_init.json)
Loading property for key='integer' from source instance
Adding key=integer, value=17 with timestamp=1755278007 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪ a_test_load_on_init.json)
Loading property for key='list' from source instance
Adding key=list, value=[1, 'two', '3', 4] with timestamp=1755278007 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪ a_test_load_on_init.json)
Loading property for key='dict' from source instance
Adding key=dict, value={'1': 1, '2': 'two', '3': '3', '4': 4} with timestamp=1755278007 to
↪ chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪ a_test_load_on_init.json)
Loading property for key='none' from source instance
Adding key=none, value=None with timestamp=1755278007 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪ a_test_load_on_init.json)
```

Success	Data from cached instance with key=str is correct (Content 'string' and Type is <class 'str'>).

```
Loading properties from cache (/home/dirk/work/unittest_collection/caching/unittest/output_da
↪ ta/cache_data_test_load_on_init.json)
Providing property for 'str' from cache
Result (Data from cached instance with key=str): 'string' (<class 'str'>)
Expectation (Data from cached instance with key=str): result = 'string' (<class 'str'>)
```

Success	Data from cached instance with key=unicode is correct (Content '__unicode__' and Type is <class 'str'>).

```
Loading property for key='unicode' from source instance
```

```
Adding key=unicode, value=__unicode__ with timestamp=1755278007 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_load_on_init.json)
Result (Data from cached instance with key=unicode): '__unicode__' (<class 'str'>)
Expectation (Data from cached instance with key=unicode): result = '__unicode__' (<class
↪  'str'>)
```

Success Data from cached instance with key=integer is correct (Content 17 and Type is <class 'int'>).

```
Providing property for 'integer' from cache
Result (Data from cached instance with key=integer): 17 (<class 'int'>)
Expectation (Data from cached instance with key=integer): result = 17 (<class 'int'>)
```

Success Data from cached instance with key=float is correct (Content 2.71828 and Type is <class 'float'>).

```
Loading property for key='float' from source instance
Adding key=float, value=2.71828 with timestamp=1755278007 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_load_on_init.json)
Result (Data from cached instance with key=float): 2.71828 (<class 'float'>)
Expectation (Data from cached instance with key=float): result = 2.71828 (<class 'float'>)
```

Success Data from cached instance with key=list is correct (Content [1, 'two', '3', 4] and Type is <class 'list'>).

```
Providing property for 'list' from cache
Result (Data from cached instance with key=list): [ 1, 'two', '3', 4 ] (<class 'list'>)
Expectation (Data from cached instance with key=list): result = [ 1, 'two', '3', 4 ] (<class
↪  'list'>)
```

Success Data from cached instance with key=dict is correct (Content {'1': 1, '2': 'two', '3': '3', '4': 4} and Type is <class 'dict'>).

```
Providing property for 'dict' from cache
Result (Data from cached instance with key=dict): { '1': 1, '2': 'two', '3': '3', '4': 4 }
↪  (<class 'dict'>)
Expectation (Data from cached instance with key=dict): result = { '1': 1, '2': 'two', '3':
↪  '3', '4': 4 } (<class 'dict'>)
```

Success Data from cached instance with key=none is correct (Content None and Type is <class 'NoneType'>).

```
Providing property for 'none' from cache
Result (Data from cached instance with key=none): None (<class 'NoneType'>)
Expectation (Data from cached instance with key=none): result = None (<class 'NoneType'>)
```

Success Data from cached instance with key=unknown_key is correct (Content 5 and Type is <class 'int'>).

```
Key 'unknown_key' is not in cached_keys. Uncached data will be returned.
Result (Data from cached instance with key=unknown_key): 5 (<class 'int'>)
Expectation (Data from cached instance with key=unknown_key): result = 5 (<class 'int'>)
```

A.1.4 REQ-0015

Testresult
This test was passed with the state: **Success**.

Info	Prepare: Cleanup before testcase execution

```
Deleting cache file from filesystem to ensure identical conditions for each test run.
```

Info	Creating empty cache file /home/dirk/work/unittest_collection/caching/unittest/output_data/cache_data_test_corrupt_cache.json.

Success	Empty cache file ignored on loading cache.

```
Exception while loading cache file /home/dirk/work/unittest_collection/caching/unittest/outpu
↪ t_data/cache_data_test_corrupt_cache.json
```

A.1.5 REQ-0004

Testresult
This test was passed with the state: **Success**.

Info	Prepare: Cleanup before testcase execution

```
Cache file does not exist on filesystem.
```

Success	Consumed time for full_update is greater expectation (Content 6.006384372711182 and Type is <class 'float'>).

```
Cache file does not exists (yet).
Loading all data from source - ['str', 'unicode', 'integer', 'float', 'list', 'dict', 'none']
Loading all data from source - ['str', 'unicode', 'integer', 'float', 'list', 'dict', 'none']
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪ a_test_full_update_sleep.json)
Result (Consumed time for full_update): 6.006384372711182 (<class 'float'>)
Expectation (Consumed time for full_update): result > 6.0 (<class 'float'>)
```

Success	Consumed time for full_update is greater expectation (Content 6.006384372711182 and Type is <class 'float'>).

```
Result (Consumed time for full_update): 6.006384372711182 (<class 'float'>)
Expectation (Consumed time for full_update): result < 6.5 (<class 'float'>)
```

A.1.6 REQ-0002

Testresult
This test was passed with the state: **Success**.

Info	Prepare: Cleanup before testcase execution

```
Deleting cache file from filesystem to ensure identical conditions for each test run.
```

Success	Data from cached instance with key=str is correct (Content 'string' and Type is <class 'str'>).

```
Cache file does not exists (yet).
Loading all data from source - ['str', 'unicode', 'integer', 'float', 'list', 'dict', 'none']
Providing property for 'str' from cache
Result (Data from cached instance with key=str): 'string' (<class 'str'>)
Expectation (Data from cached instance with key=str): result = 'string' (<class 'str'>)
```

Success	Data from cached instance with key=unicode is correct (Content 'unicode' and Type is <class 'str'>).

```
Providing property for 'unicode' from cache
Result (Data from cached instance with key=unicode): 'unicode' (<class 'str'>)
Expectation (Data from cached instance with key=unicode): result = 'unicode' (<class 'str'>)
```

Success	Data from cached instance with key=integer is correct (Content 17 and Type is <class 'int'>).

```
Providing property for 'integer' from cache
Result (Data from cached instance with key=integer): 17 (<class 'int'>)
Expectation (Data from cached instance with key=integer): result = 17 (<class 'int'>)
```

Success	Data from cached instance with key=float is correct (Content 3.14159 and Type is <class 'float'>).

```
Providing property for 'float' from cache
Result (Data from cached instance with key=float): 3.14159 (<class 'float'>)
Expectation (Data from cached instance with key=float): result = 3.14159 (<class 'float'>)
```

Success	Data from cached instance with key=list is correct (Content [1, 'two', '3', 4] and Type is <class 'list'>).

```
Providing property for 'list' from cache
Result (Data from cached instance with key=list): [ 1, 'two', '3', 4 ] (<class 'list'>)
Expectation (Data from cached instance with key=list): result = [ 1, 'two', '3', 4 ] (<class
↪  'list'>)
```

Success	Data from cached instance with key=dict is correct (Content {'1': 1, '2': 'two', '3': '3', '4': 4} and Type is <class 'dict'>).

```
Providing property for 'dict' from cache
```

```
Result (Data from cached instance with key=dict): { '1': 1, '2': 'two', '3': '3', '4': 4 }
↪  (<class 'dict'>)
Expectation (Data from cached instance with key=dict): result = { '1': 1, '2': 'two', '3':
↪  '3', '4': 4 } (<class 'dict'>)
```

Success Data from cached instance with key=none is correct (Content None and Type is <class 'NoneType'>).

```
Providing property for 'none' from cache
Result (Data from cached instance with key=none): None (<class 'NoneType'>)
Expectation (Data from cached instance with key=none): result = None (<class 'NoneType'>)
```

Success The cache file (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_data_test_full_update_sleep.json) shall not exist is correct (Content False and Type is <class 'bool'>).

```
Result (The cache file (/home/dirk/work/unittest_collection/caching/unittest/output_data/cach
↪  e_data_test_full_update_sleep.json) shall not exist): False (<class 'bool'>)
Expectation (The cache file (/home/dirk/work/unittest_collection/caching/unittest/output_data
↪  /cache_data_test_full_update_sleep.json) shall not exist): result = False (<class 'bool'>)
```

A.1.7 REQ-0006

Testresult

This test was passed with the state: **Success**.

Info Prepare: Cleanup before testcase execution

```
Deleting cache file from filesystem to ensure identical conditions for each test run.
```

Info Prepare: First usage of 'property_cache_json' with a class holding the data to be cached

```
Cache file does not exists (yet).
Loading all data from source - ['str', 'unicode', 'integer', 'float', 'list', 'dict', 'none']
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_dump_cache.json)
```

Success Data from cached instance with key=str is correct (Content 'string' and Type is <class 'str'>).

```
Loading properties from cache (/home/dirk/work/unittest_collection/caching/unittest/output_da
↪  ta/cache_data_test_dump_cache.json)
Providing property for 'str' from cache
Result (Data from cached instance with key=str): 'string' (<class 'str'>)
Expectation (Data from cached instance with key=str): result = 'string' (<class 'str'>)
```

Success Data from cached instance with key=unicode is correct (Content 'unicode' and Type is <class 'str'>).

```
Providing property for 'unicode' from cache
```

```
Result (Data from cached instance with key=unicode): 'unicode' (<class 'str'>)
Expectation (Data from cached instance with key=unicode): result = 'unicode' (<class 'str'>)
```

Success	Data from cached instance with key=integer is correct (Content 17 and Type is <class 'int'>).

```
Providing property for 'integer' from cache
Result (Data from cached instance with key=integer): 17 (<class 'int'>)
Expectation (Data from cached instance with key=integer): result = 17 (<class 'int'>)
```

Success	Data from cached instance with key=float is correct (Content 3.14159 and Type is <class 'float'>).

```
Providing property for 'float' from cache
Result (Data from cached instance with key=float): 3.14159 (<class 'float'>)
Expectation (Data from cached instance with key=float): result = 3.14159 (<class 'float'>)
```

Success	Data from cached instance with key=list is correct (Content [1, 'two', '3', 4] and Type is <class 'list'>).

```
Providing property for 'list' from cache
Result (Data from cached instance with key=list): [ 1, 'two', '3', 4 ] (<class 'list'>)
Expectation (Data from cached instance with key=list): result = [ 1, 'two', '3', 4 ] (<class
↪  'list'>)
```

Success	Data from cached instance with key=dict is correct (Content {'1': 1, '2': 'two', '3': '3', '4': 4} and Type is <class 'dict'>).

```
Providing property for 'dict' from cache
Result (Data from cached instance with key=dict): { '1': 1, '2': 'two', '3': '3', '4': 4 }
↪  (<class 'dict'>)
Expectation (Data from cached instance with key=dict): result = { '1': 1, '2': 'two', '3':
↪  '3', '4': 4 } (<class 'dict'>)
```

Success	Data from cached instance with key=none is correct (Content None and Type is <class 'NoneType'>).

```
Providing property for 'none' from cache
Result (Data from cached instance with key=none): None (<class 'NoneType'>)
Expectation (Data from cached instance with key=none): result = None (<class 'NoneType'>)
```

Success	Data from cached instance with key=str is correct (Content '__string__' and Type is <class 'str'>).

```
The cached value is old, cached value will be ignored
Loading property for key='str' from source instance
Adding key=str, value=__string__ with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=str): '__string__' (<class 'str'>)
Expectation (Data from cached instance with key=str): result = '__string__' (<class 'str'>)
```

Success Data from cached instance with key=unicode is correct (Content '__unicode__' and Type is <class 'str'>).

```
The cached value is old, cached value will be ignored
Loading property for key='unicode' from source instance
Adding key=unicode, value=__unicode__ with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪ a_test_dump_cache.json)
Result (Data from cached instance with key=unicode): '__unicode__' (<class 'str'>)
Expectation (Data from cached instance with key=unicode): result = '__unicode__' (<class
↪ 'str'>)
```

Success Data from cached instance with key=integer is correct (Content 34 and Type is <class 'int'>).

```
The cached value is old, cached value will be ignored
Loading property for key='integer' from source instance
Adding key=integer, value=34 with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪ a_test_dump_cache.json)
Result (Data from cached instance with key=integer): 34 (<class 'int'>)
Expectation (Data from cached instance with key=integer): result = 34 (<class 'int'>)
```

Success Data from cached instance with key=float is correct (Content 2.71828 and Type is <class 'float'>).

```
The cached value is old, cached value will be ignored
Loading property for key='float' from source instance
Adding key=float, value=2.71828 with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪ a_test_dump_cache.json)
Result (Data from cached instance with key=float): 2.71828 (<class 'float'>)
Expectation (Data from cached instance with key=float): result = 2.71828 (<class 'float'>)
```

Success Data from cached instance with key=list is correct (Content ['one', 2, 3, '4'] and Type is <class 'list'>).

```
The cached value is old, cached value will be ignored
Loading property for key='list' from source instance
Adding key=list, value=['one', 2, 3, '4'] with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪ a_test_dump_cache.json)
Result (Data from cached instance with key=list): [ 'one', 2, 3, '4' ] (<class 'list'>)
Expectation (Data from cached instance with key=list): result = [ 'one', 2, 3, '4' ] (<class
↪ 'list'>)
```

Success	Data from cached instance with key=dict is correct (Content {'1': '1', '2': 2, '3': 'three', '4': '4'} and Type is <class 'dict'>).

```
The cached value is old, cached value will be ignored
Loading property for key='dict' from source instance
Adding key=dict, value={'1': '1', '2': 2, '3': 'three', '4': '4'} with timestamp=1755278015 to
↪  chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat⌋
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=dict): { '1': '1', '2': 2, '3': 'three', '4': '4' }
↪  (<class 'dict'>)
Expectation (Data from cached instance with key=dict): result = { '1': '1', '2': 2, '3':
↪  'three', '4': '4' } (<class 'dict'>)
```

Success	Data from cached instance with key=none is correct (Content 'not None' and Type is <class 'str'>).

```
The cached value is old, cached value will be ignored
Loading property for key='none' from source instance
Adding key=none, value=not None with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat⌋
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=none): 'not None' (<class 'str'>)
Expectation (Data from cached instance with key=none): result = 'not None' (<class 'str'>)
```

A.1.8 REQ-0007

Testresult

This test was passed with the state: **Success**.

Info	Prepare: Cleanup before testcase execution

```
Deleting cache file from filesystem to ensure identical conditions for each test run.
```

Info	Prepare: First usage of 'property_cache_json' with a class holding the data to be cached

```
Cache file does not exists (yet).
Loading all data from source - ['str', 'unicode', 'integer', 'float', 'list', 'dict', 'none']
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat⌋
↪  a_test_dump_cache.json)
```

Success	Data from cached instance with key=str is correct (Content '__string__' and Type is <class 'str'>).

```
Loading properties from cache (/home/dirk/work/unittest_collection/caching/unittest/output_da⌋
↪  ta/cache_data_test_dump_cache.json)
```

```
Data version increased, ignoring previous cache data
Loading property for key='str' from source instance
Adding key=str, value=__string__ with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=str): '__string__' (<class 'str'>)
Expectation (Data from cached instance with key=str): result = '__string__' (<class 'str'>)
```

Success Data from cached instance with key=unicode is correct (Content '__unicode__' and Type is <class 'str'>).

```
Loading property for key='unicode' from source instance
Adding key=unicode, value=__unicode__ with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=unicode): '__unicode__' (<class 'str'>)
Expectation (Data from cached instance with key=unicode): result = '__unicode__' (<class
↪  'str'>)
```

Success Data from cached instance with key=integer is correct (Content 34 and Type is <class 'int'>).

```
Loading property for key='integer' from source instance
Adding key=integer, value=34 with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=integer): 34 (<class 'int'>)
Expectation (Data from cached instance with key=integer): result = 34 (<class 'int'>)
```

Success Data from cached instance with key=float is correct (Content 2.71828 and Type is <class 'float'>).

```
Loading property for key='float' from source instance
Adding key=float, value=2.71828 with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=float): 2.71828 (<class 'float'>)
Expectation (Data from cached instance with key=float): result = 2.71828 (<class 'float'>)
```

Success Data from cached instance with key=list is correct (Content ['one', 2, 3, '4'] and Type is <class 'list'>).

```
Loading property for key='list' from source instance
Adding key=list, value=['one', 2, 3, '4'] with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=list): [ 'one', 2, 3, '4' ] (<class 'list'>)
Expectation (Data from cached instance with key=list): result = [ 'one', 2, 3, '4' ] (<class
↪  'list'>)
```

Success	Data from cached instance with key=dict is correct (Content {'1': '1', '2': 2, '3': 'three', '4': '4'} and Type is <class 'dict'>).

```
Loading property for key='dict' from source instance
Adding key=dict, value={'1': '1', '2': 2, '3': 'three', '4': '4'} with timestamp=1755278015 to
↪  chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat⌋
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=dict): { '1': '1', '2': 2, '3': 'three', '4': '4' }
↪  (<class 'dict'>)
Expectation (Data from cached instance with key=dict): result = { '1': '1', '2': 2, '3':
↪  'three', '4': '4' } (<class 'dict'>)
```

Success	Data from cached instance with key=none is correct (Content 'not None' and Type is <class 'str'>).

```
Loading property for key='none' from source instance
Adding key=none, value=not None with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat⌋
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=none): 'not None' (<class 'str'>)
Expectation (Data from cached instance with key=none): result = 'not None' (<class 'str'>)
```

A.1.9 REQ-0008

Testresult

This test was passed with the state: **Success**.

Info	Prepare: Cleanup before testcase execution

```
Deleting cache file from filesystem to ensure identical conditions for each test run.
```

Info	Prepare: First usage of 'property_cache_json' with a class holding the data to be cached

```
Cache file does not exists (yet).
Loading all data from source - ['str', 'unicode', 'integer', 'float', 'list', 'dict', 'none']
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat⌋
↪  a_test_dump_cache.json)
```

Success	Data from cached instance with key=str is correct (Content '__string__' and Type is <class 'str'>).

```
Loading properties from cache (/home/dirk/work/unittest_collection/caching/unittest/output_da⌋
↪  ta/cache_data_test_dump_cache.json)
Source uid changed, ignoring previous cache data
```

```
Loading property for key='str' from source instance
Adding key=str, value=__string__ with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=str): '__string__' (<class 'str'>)
Expectation (Data from cached instance with key=str): result = '__string__' (<class 'str'>)
```

Success Data from cached instance with key=unicode is correct (Content '__unicode__' and Type is <class 'str'>).

```
Loading property for key='unicode' from source instance
Adding key=unicode, value=__unicode__ with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=unicode): '__unicode__' (<class 'str'>)
Expectation (Data from cached instance with key=unicode): result = '__unicode__' (<class
↪  'str'>)
```

Success Data from cached instance with key=integer is correct (Content 34 and Type is <class 'int'>).

```
Loading property for key='integer' from source instance
Adding key=integer, value=34 with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=integer): 34 (<class 'int'>)
Expectation (Data from cached instance with key=integer): result = 34 (<class 'int'>)
```

Success Data from cached instance with key=float is correct (Content 2.71828 and Type is <class 'float'>).

```
Loading property for key='float' from source instance
Adding key=float, value=2.71828 with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=float): 2.71828 (<class 'float'>)
Expectation (Data from cached instance with key=float): result = 2.71828 (<class 'float'>)
```

Success Data from cached instance with key=list is correct (Content ['one', 2, 3, '4'] and Type is <class 'list'>).

```
Loading property for key='list' from source instance
Adding key=list, value=['one', 2, 3, '4'] with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=list): [ 'one', 2, 3, '4' ] (<class 'list'>)
Expectation (Data from cached instance with key=list): result = [ 'one', 2, 3, '4' ] (<class
↪  'list'>)
```

Success	Data from cached instance with key=dict is correct (Content {'1': '1', '2': 2, '3': 'three', '4': '4'} and Type is <class 'dict'>).

```
Loading property for key='dict' from source instance
Adding key=dict, value={'1': '1', '2': 2, '3': 'three', '4': '4'} with timestamp=1755278015 to
↪  chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat⌋
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=dict): { '1': '1', '2': 2, '3': 'three', '4': '4' }
↪  (<class 'dict'>)
Expectation (Data from cached instance with key=dict): result = { '1': '1', '2': 2, '3':
↪  'three', '4': '4' } (<class 'dict'>)
```

Success	Data from cached instance with key=none is correct (Content 'not None' and Type is <class 'str'>).

```
Loading property for key='none' from source instance
Adding key=none, value=not None with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat⌋
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=none): 'not None' (<class 'str'>)
Expectation (Data from cached instance with key=none): result = 'not None' (<class 'str'>)
```

A.1.10 REQ-0009

Testresult

This test was passed with the state: **Success**.

Info	Prepare: Cleanup before testcase execution

```
Deleting cache file from filesystem to ensure identical conditions for each test run.
```

Info	Prepare: First usage of 'property_cache_json' with a class holding the data to be cached

```
Cache file does not exists (yet).
Loading all data from source - ['str', 'unicode', 'integer', 'float', 'list', 'dict', 'none']
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat⌋
↪  a_test_dump_cache.json)
```

Success	Data from cached instance with key=str is correct (Content '__string__' and Type is <class 'str'>).

```
Loading properties from cache (/home/dirk/work/unittest_collection/caching/unittest/output_da⌋
↪  ta/cache_data_test_dump_cache.json)
Storage version changed, ignoring previous cache data
```

```
Loading property for key='str' from source instance
Adding key=str, value=__string__ with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=str): '__string__' (<class 'str'>)
Expectation (Data from cached instance with key=str): result = '__string__' (<class 'str'>)
```

Success Data from cached instance with key=unicode is correct (Content '__unicode__' and Type is <class 'str'>).

```
Loading property for key='unicode' from source instance
Adding key=unicode, value=__unicode__ with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=unicode): '__unicode__' (<class 'str'>)
Expectation (Data from cached instance with key=unicode): result = '__unicode__' (<class
↪  'str'>)
```

Success Data from cached instance with key=integer is correct (Content 34 and Type is <class 'int'>).

```
Loading property for key='integer' from source instance
Adding key=integer, value=34 with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=integer): 34 (<class 'int'>)
Expectation (Data from cached instance with key=integer): result = 34 (<class 'int'>)
```

Success Data from cached instance with key=float is correct (Content 2.71828 and Type is <class 'float'>).

```
Loading property for key='float' from source instance
Adding key=float, value=2.71828 with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=float): 2.71828 (<class 'float'>)
Expectation (Data from cached instance with key=float): result = 2.71828 (<class 'float'>)
```

Success Data from cached instance with key=list is correct (Content ['one', 2, 3, '4'] and Type is <class 'list'>).

```
Loading property for key='list' from source instance
Adding key=list, value=['one', 2, 3, '4'] with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=list): [ 'one', 2, 3, '4' ] (<class 'list'>)
Expectation (Data from cached instance with key=list): result = [ 'one', 2, 3, '4' ] (<class
↪  'list'>)
```

Success	Data from cached instance with key=dict is correct (Content {'1': '1', '2': 2, '3': 'three', '4': '4'} and Type is <class 'dict'>).

```
Loading property for key='dict' from source instance
Adding key=dict, value={'1': '1', '2': 2, '3': 'three', '4': '4'} with timestamp=1755278015 to
↪  chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat⌋
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=dict): { '1': '1', '2': 2, '3': 'three', '4': '4' }
↪  (<class 'dict'>)
Expectation (Data from cached instance with key=dict): result = { '1': '1', '2': 2, '3':
↪  'three', '4': '4' } (<class 'dict'>)
```

Success	Data from cached instance with key=none is correct (Content 'not None' and Type is <class 'str'>).

```
Loading property for key='none' from source instance
Adding key=none, value=not None with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat⌋
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=none): 'not None' (<class 'str'>)
Expectation (Data from cached instance with key=none): result = 'not None' (<class 'str'>)
```

A.1.11 REQ-0014

Testresult

This test was passed with the state: **Success**.

Info	Prepare: Cleanup before testcase execution

```
Deleting cache file from filesystem to ensure identical conditions for each test run.
```

Info	Prepare: First usage of 'property_cache_json' with a class holding the data to be cached

```
Cache file does not exists (yet).
Loading all data from source - ['str', 'unicode', 'integer', 'float', 'list', 'dict', 'none']
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat⌋
↪  a_test_dump_cache.json)
```

Success	Data from cached instance with key=str is correct (Content 'string' and Type is <class 'str'>).

```
Loading properties from cache (/home/dirk/work/unittest_collection/caching/unittest/output_da⌋
↪  ta/cache_data_test_dump_cache.json)
Providing property for 'str' from cache
```

```
Result (Data from cached instance with key=str): 'string' (<class 'str'>)
Expectation (Data from cached instance with key=str): result = 'string' (<class 'str'>)
```

Success	Data from cached instance with key=unicode is correct (Content 'unicode' and Type is <class 'str'>).

```
Providing property for 'unicode' from cache
Result (Data from cached instance with key=unicode): 'unicode' (<class 'str'>)
Expectation (Data from cached instance with key=unicode): result = 'unicode' (<class 'str'>)
```

Success	Data from cached instance with key=integer is correct (Content 17 and Type is <class 'int'>).

```
Providing property for 'integer' from cache
Result (Data from cached instance with key=integer): 17 (<class 'int'>)
Expectation (Data from cached instance with key=integer): result = 17 (<class 'int'>)
```

Success	Data from cached instance with key=float is correct (Content 3.14159 and Type is <class 'float'>).

```
Providing property for 'float' from cache
Result (Data from cached instance with key=float): 3.14159 (<class 'float'>)
Expectation (Data from cached instance with key=float): result = 3.14159 (<class 'float'>)
```

Success	Data from cached instance with key=list is correct (Content [1, 'two', '3', 4] and Type is <class 'list'>).

```
Providing property for 'list' from cache
Result (Data from cached instance with key=list): [ 1, 'two', '3', 4 ] (<class 'list'>)
Expectation (Data from cached instance with key=list): result = [ 1, 'two', '3', 4 ] (<class
↪  'list'>)
```

Success	Data from cached instance with key=dict is correct (Content {'1': 1, '2': 'two', '3': '3', '4': 4} and Type is <class 'dict'>).

```
Providing property for 'dict' from cache
Result (Data from cached instance with key=dict): { '1': 1, '2': 'two', '3': '3', '4': 4 }
↪  (<class 'dict'>)
Expectation (Data from cached instance with key=dict): result = { '1': 1, '2': 'two', '3':
↪  '3', '4': 4 } (<class 'dict'>)
```

Success	Data from cached instance with key=none is correct (Content 'not None' and Type is <class 'str'>).

```
Providing property for 'none' from cache
Loading property for key='none' from source instance
Adding key=none, value=not None with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat⌋
↪  a_test_dump_cache.json)
Result (Data from cached instance with key=none): 'not None' (<class 'str'>)
Expectation (Data from cached instance with key=none): result = 'not None' (<class 'str'>)
```

A.1.12 REQ-0010

Testresult
This test was passed with the state: **Success**.

Info	Prepare: Cleanup before testcase execution

```
Deleting cache file from filesystem to ensure identical conditions for each test run.
```

Info	Prepare: First usage of 'property_cache_json' with a class holding the data to be cached

```
Cache file does not exists (yet).
Loading all data from source - ['str', 'unicode', 'integer', 'float', 'list', 'dict', 'none']
Loading all data from source - ['str', 'unicode', 'integer', 'float', 'list', 'dict', 'none']
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat⌋
↪ a_test_source_key_def.json)
```

Success	Data from cached instance with key=str is correct (Content '__string__' and Type is <class 'str'>).

```
Loading properties from cache (/home/dirk/work/unittest_collection/caching/unittest/output_da⌋
↪ ta/cache_data_test_source_key_def.json)
Loading property for key='str' from source instance
Adding key=str, value=__string__ with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat⌋
↪ a_test_source_key_def.json)
Result (Data from cached instance with key=str): '__string__' (<class 'str'>)
Expectation (Data from cached instance with key=str): result = '__string__' (<class 'str'>)
```

Success	Data from cached instance with key=unicode is correct (Content 'unicode' and Type is <class 'str'>).

```
Providing property for 'unicode' from cache
Result (Data from cached instance with key=unicode): 'unicode' (<class 'str'>)
Expectation (Data from cached instance with key=unicode): result = 'unicode' (<class 'str'>)
```

Success	Data from cached instance with key=integer is correct (Content 17 and Type is <class 'int'>).

```
Providing property for 'integer' from cache
Result (Data from cached instance with key=integer): 17 (<class 'int'>)
Expectation (Data from cached instance with key=integer): result = 17 (<class 'int'>)
```

Success	Data from cached instance with key=float is correct (Content 2.71828 and Type is <class 'float'>).

```
Loading property for key='float' from source instance
Adding key=float, value=2.71828 with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/cache_dat⌋
↪ a_test_source_key_def.json)
```

```
Result (Data from cached instance with key=float): 2.71828 (<class 'float'>)
Expectation (Data from cached instance with key=float): result = 2.71828 (<class 'float'>)
```

Success	Data from cached instance with key=list is correct (Content [1, 'two', '3', 4] and Type is <class 'list'>).

```
Providing property for 'list' from cache
Result (Data from cached instance with key=list): [ 1, 'two', '3', 4 ] (<class 'list'>)
Expectation (Data from cached instance with key=list): result = [ 1, 'two', '3', 4 ] (<class
↪  'list'>)
```

Success	Data from cached instance with key=dict is correct (Content {'1': 1, '2': 'two', '3': '3', '4': 4} and Type is <class 'dict'>).

```
Providing property for 'dict' from cache
Result (Data from cached instance with key=dict): { '1': 1, '2': 'two', '3': '3', '4': 4 }
↪  (<class 'dict'>)
Expectation (Data from cached instance with key=dict): result = { '1': 1, '2': 'two', '3':
↪  '3', '4': 4 } (<class 'dict'>)
```

Success	Data from cached instance with key=none is correct (Content None and Type is <class 'NoneType'>).

```
Providing property for 'none' from cache
Result (Data from cached instance with key=none): None (<class 'NoneType'>)
Expectation (Data from cached instance with key=none): result = None (<class 'NoneType'>)
```

A.1.13 REQ-0011

Testresult

This test was passed with the state: **Success**.

Info	Prepare: Cleanup before testcase execution

```
Deleting cache file from filesystem to ensure identical conditions for each test run.
```

Info	Installing save_callback with no get or full_update execution.

Success	Save callback execution counter is correct (Content 0 and Type is <class 'int'>).

```
Result (Save callback execution counter): 0 (<class 'int'>)
Expectation (Save callback execution counter): result = 0 (<class 'int'>)
```

Success	Save callback execution counter is correct (Content None and Type is <class 'NoneType'>).

```
Result (Save callback execution counter): None (<class 'NoneType'>)
Expectation (Save callback execution counter): result = None (<class 'NoneType'>)
```

A.1.14 REQ-0012

Testresult

This test was passed with the state: **Success**.

Info	Prepare: Cleanup before testcase execution

```
Cache file does not exist on filesystem.
```

Info	Installing save_callback and execute full_update.

```
Cache file does not exists (yet).
Loading all data from source - ['str', 'unicode', 'integer', 'float', 'list', 'dict', 'none']
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/save_call
↪ back_callback.json)
```

Success	Save callback execution counter is correct (Content 1 and Type is <class 'int'>).

```
Result (Save callback execution counter): 1 (<class 'int'>)
Expectation (Save callback execution counter): result = 1 (<class 'int'>)
```

Success	Save callback execution counter is correct (Content <caching.property_cache_json object at 0x7f7c48d81b30> and Type is <class 'caching.property_cache_json'>).

```
Result (Save callback execution counter): <caching.property_cache_json object at
↪ 0x7f7c48d81b30> (<class 'caching.property_cache_json'>)
Expectation (Save callback execution counter): result = <caching.property_cache_json object at
↪ 0x7f7c48d81b30> (<class 'caching.property_cache_json'>)
```

A.1.15 REQ-0013

Testresult

This test was passed with the state: **Success**.

Info	Prepare: Cleanup before testcase execution

```
Deleting cache file from filesystem to ensure identical conditions for each test run.
```

Info	Installing save_callback and execute a single get.

```
Cache file does not exists (yet).
Loading property for key='str' from source instance
Adding key=str, value=string with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/save_call
↪ back_callback.json)
```

Info Installing save_callback and execute a single get.

```
Loading property for key='unicode' from source instance
Adding key=unicode, value=unicode with timestamp=1755278015 to chache
cache-file stored (/home/dirk/work/unittest_collection/caching/unittest/output_data/save_call
↪ back_callback.json)
```

Success Save callback execution counter is correct (Content 2 and Type is <class 'int'>).

```
Result (Save callback execution counter): 2 (<class 'int'>)
Expectation (Save callback execution counter): result = 2 (<class 'int'>)
```

Success Save callback execution counter is correct (Content <caching.property_cache_json object at 0x7f7c48d80cd0> and Type is <class 'caching.property_cache_json'>).

```
Result (Save callback execution counter): <caching.property_cache_json object at
↪ 0x7f7c48d80cd0> (<class 'caching.property_cache_json'>)
Expectation (Save callback execution counter): result = <caching.property_cache_json object at
↪ 0x7f7c48d80cd0> (<class 'caching.property_cache_json'>)
```

B Test-Coverage

B.1 caching

The line coverage for caching was 97.3%
The branch coverage for caching was 95.8%

B.1.1 caching.__init__.py

The line coverage for caching.__init__.py was 97.3%
The branch coverage for caching.__init__.py was 95.8%

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
"""
caching (Caching Module)
========================

**Author:**

* Dirk Alders <sudo-dirk@mount-mockery.de>

**Description:**

    This Module supports functions and classes for caching e.g. properties of other instances.

**Submodules:**
```

```
17
18 * :class:`caching.property_cache_json`
19 * :class:`caching.property_cache_pickle`
20
21 **Unittest:**
22
23     See also the :download:`unittest <caching/_testresults_/unittest.pdf>` documentation.
24 """
25 __DEPENDENCIES__ = []
26
27 import json
28 import logging
29 import os
30 import pickle
31 import time
32
33 try:
34     from config import APP_NAME as ROOT_LOGGER_NAME
35 except ImportError:
36     ROOT_LOGGER_NAME = 'root'
37 logger = logging.getLogger(ROOT_LOGGER_NAME).getChild(__name__)
38
39 __DESCRIPTION__ = """The Module {\\tt %s} is designed to store information in {\\tt json} or {\\
       tt pickle} files to support them much faster then generating them from the original source
       file.
40 For more Information read the documentation.""" % __name__.replace('_', '\\_')
41 """The Module Description"""
42 __INTERPRETER__ = (3, )
43 """The Tested Interpreter-Versions"""
44
45
46 class property_cache_pickle(object):
47     """
48     This class caches the data from a given `source_instance`. It takes the data from the cache
       instead of generating the data from the `source_instance`,
49     if the conditions for the cache usage are given.
50
51     .. admonition:: Required properties for the `source_instance`
52
53                 * **uid():** returns the unique id of the source's source or None, if you don't
       want to use the unique id.
54                 * **keys():** returns a list of all available keys.
55                 * **data_version():** returns a version number of the current data (it should be
       increased, if the get method of the source instance returns improved values or the data
       structure had been changed).
56                 * **get(key, default):** returns the property for a key. If key does not exists,
       default will be returned.
57
58     :param source_instance: The source instance holding the data
59     :type source_instance: instance
60     :param cache_filename: File name, where the properties are stored as cache
61     :type cache_filename: str
62     :param load_all_on_init: True will load all data from the source instance, when the cache
       will be initialised the first time.
63     :type load_all_on_init: bool
64     :param callback_on_data_storage: The callback will be executed every time when the cache file
        is stored. It will be executed with the instance of this class as first argument.
65     :type callback_on_data_storage: method
66     :param max_age: The maximum age of the cached data in seconds or None for no maximum age.
67     :type max_age: int or None
68     :param store_on_get: False will prevent cache storage with execution of the `.get(key,
       default)` method. You need to store the cache somewhere else.
```

```
69      :type store_on_get: bool
70
71      .. admonition:: The cache will be used, if all following conditions are given
72
73                  * The key is in the list returned by `.keys()` method of the `source_instance`
74                  * The key is not in the list of keys added by the `.add_source_get_keys()` method
    .
75                  * The cache age is less then the given max_age parameter or the given max_age is
    None.
76                  * The uid of the source instance (e.g. a checksum or unique id of the source) is
    identically to to uid stored in the cache.
77                  * The data version of the `source_instance` is <= the data version stored in the
    cache.
78                  * The value is available in the previous stored information
79
80      **Example:**
81
82      .. literalinclude:: caching/_examples_/property_cache_pickle.py
83
84      Will result on the first execution to the following output (with a long execution time):
85
86      .. literalinclude:: caching/_examples_/property_cache_pickle.log_1st
87
88      With every following execution the time cosumption my by much smaller:
89
90      .. literalinclude:: caching/_examples_/property_cache_pickle.log
91      """
92      DATA_VERSION_TAG = '_property_cache_data_version_'
93      STORAGE_VERSION_TAG = '_storage_version_'
94      UID_TAG = '_property_cache_uid_'
95      DATA_TAG = '_data_'
96      AGE_TAG = '_age_'
97      #
98      STORAGE_VERSION = 1
99
100     def __init__(self, source_instance, cache_filename, load_all_on_init=False,
    callback_on_data_storage=None, max_age=None, store_on_get=True, return_source_on_none=False):
101         self._source_instance = source_instance
102         self._cache_filename = cache_filename
103         self._load_all_on_init = load_all_on_init
104         self._callback_on_data_storage = callback_on_data_storage
105         self._max_age = max_age
106         self._store_on_get = store_on_get
107         self._return_source_on_none = return_source_on_none
108         #
109         self._source_get_keys = []
110         self._cached_props = None
111
112     def add_source_get_keys(self, keys):
113         """
114         This will add one or more keys to a list of keys which will always be provided by the `
    source_instance` instead of the cache.
115
116         :param keys: The key or keys to be added
117         :type keys: list, tuple, str
118         """
119         if type(keys) in [list, tuple]:
120             self._source_get_keys.extend(keys)
121         else:
122             self._source_get_keys.append(keys)
123
124     def full_update(self, sleep_between_keys=0):
```

```
125         """
126         With the execution of this method, the complete source data which needs to be cached,
    will be read from the source instance
127         and the resulting cache will be stored to the given file.
128 
129         :param sleep_between_keys: Time to sleep between each source data generation
130         :type sleep_between_keys: float, int
131 
132         .. hint:: Use this method, if you initiallised the class with `store_on_get=False`
133         """
134         self._load_source(sleep_between_keys=sleep_between_keys)
135         self._save_cache()
136 
137     def get(self, key, default=None):
138         """
139         Method to get the cached property. If the key does not exists in the cache or `
    source_instance`, `default` will be returned.
140 
141         :param key: key for value to get.
142         :param default: value to be returned, if key does not exists.
143         :returns: value for a given key or default value.
144         """
145         # Init cache
146         if self._cached_props is None:
147             self._init_cache()
148         # Identify old cache
149         if self._max_age is None:
150             cache_old = False
151         else:
152             cache_old = time.time() - self._cached_props[self.AGE_TAG].get(self._key_filter(key),
     0) > self._max_age
153             if cache_old:
154                 logger.debug("The cached value is old, cached value will be ignored")
155         # Return cached value
156         if not cache_old and key not in self._source_get_keys and self._key_filter(key) in self.
    _cached_props[self.DATA_TAG]:
157             logger.debug("Providing property for '%s' from cache", key)
158             rv = self._cached_props[self.DATA_TAG].get(self._key_filter(key), default)
159             if rv is not None or not self._return_source_on_none:
160                 return rv
161         # Create cache and return value
162         if key in self._source_instance.keys():
163             logger.debug("Loading property for key='%s' from source instance", key)
164             val = self._source_instance.get(key, None)
165             if self._store_on_get:
166                 tm = int(time.time())
167                 logger.debug("Adding key=%s, value=%s with timestamp=%d to chache", key, val, tm)
168                 self._cached_props[self.DATA_TAG][self._key_filter(key)] = val
169                 self._cached_props[self.AGE_TAG][self._key_filter(key)] = tm
170                 self._save_cache()
171             else:
172                 return val
173             cached_data = self._cached_props[self.DATA_TAG].get(self._key_filter(key), default)
174             if cached_data is None and self._return_source_on_none:
175                 return self._source_instance.get(key, default)
176             return cached_data
177         else:
178             if key not in self._source_instance.keys():
179                 logger.debug("Key '%s' is not in cached_keys. Uncached data will be returned.",
    key)
180             else:
```

```
181                 logger.debug("Key '%s' is excluded by .add_source_get_keys(). Uncached data will
    be returned.", key)
182             return self._source_instance.get(key, default)
183
184     def _data_version(self):
185         if self._cached_props is None:
186             return None
187         else:
188             return self._cached_props.get(self.DATA_VERSION_TAG, None)
189
190     def _storage_version(self):
191         if self._cached_props is None:
192             return None
193         else:
194             return self._cached_props.get(self.STORAGE_VERSION_TAG, None)
195
196     def _init_cache(self):
197         load_cache = self._load_cache()
198         uid = self._source_instance.uid() != self._uid()
199         try:
200             data_version = self._source_instance.data_version() > self._data_version()
201         except TypeError:
202             data_version = True
203         try:
204             storage_version = self._storage_version() != self.STORAGE_VERSION
205         except TypeError:
206             storage_version = True
207         #
208         if not load_cache or uid or data_version or storage_version:
209             if load_cache:
210                 if self._uid() is not None and uid:
211                     logger.debug("Source uid changed, ignoring previous cache data")
212                 if self._data_version() is not None and data_version:
213                     logger.debug("Data version increased, ignoring previous cache data")
214                 if storage_version:
215                     logger.debug("Storage version changed, ignoring previous cache data")
216             self._cached_props = {self.AGE_TAG: {}, self.DATA_TAG: {}}
217             if self._load_all_on_init:
218                 self._load_source()
219             self._cached_props[self.UID_TAG] = self._source_instance.uid()
220             self._cached_props[self.DATA_VERSION_TAG] = self._source_instance.data_version()
221             self._cached_props[self.STORAGE_VERSION_TAG] = self.STORAGE_VERSION
222
223     def _load_only(self):
224         with open(self._cache_filename, 'rb') as fh:
225             self._cached_props = pickle.load(fh)
226         logger.debug('Loading properties from cache (%s)', self._cache_filename)
227
228     def _load_cache(self):
229         if os.path.exists(self._cache_filename):
230             try:
231                 self._load_only()
232             except:
233                 logger.exception("Exception while loading cache file %s", self._cache_filename)
234             else:
235                 return True
236         else:
237             logger.debug('Cache file does not exists (yet).')
238         return False
239
240     def _key_filter(self, key):
241         return key
242
```

```
243     def _load_source(self, sleep_between_keys=0):
244         if self._cached_props is None:
245             self._init_cache()
246         logger.debug('Loading all data from source - %s', repr(self._source_instance.keys()))
247         for key in self._source_instance.keys():
248             if key not in self._source_get_keys:
249                 self._cached_props[self.DATA_TAG][self._key_filter(key)] = self._source_instance.
    get(key)
250                 self._cached_props[self.AGE_TAG][self._key_filter(key)] = int(time.time())
251                 time.sleep(sleep_between_keys)
252 
253     def _save_only(self):
254         with open(self._cache_filename, 'wb') as fh:
255             pickle.dump(self._cached_props, fh)
256             logger.debug('cache-file stored (%s)', self._cache_filename)
257 
258     def _save_cache(self):
259         self._save_only()
260         if self._callback_on_data_storage is not None:
261             self._callback_on_data_storage(self)
262 
263     def _uid(self):
264         if self._cached_props is None:
265             return None
266         else:
267             return self._cached_props.get(self.UID_TAG, None)
268 
269 
270 class property_cache_json(property_cache_pickle):
271     """
272     See also parent :py:class:`property_cache_pickle` for detailed information.
273 
274     .. important::
275         * This class uses json. You should **only** use keys of type string!
276         * Unicode types are transfered to strings
277 
278         See limitations of json.
279 
280     **Example:**
281 
282     .. literalinclude:: caching/_examples_/property_cache_json.py
283 
284     Will result on the first execution to the following output (with a long execution time):
285 
286     .. literalinclude:: caching/_examples_/property_cache_json.log_1st
287 
288     With every following execution the time cosumption my by much smaller:
289 
290     .. literalinclude:: caching/_examples_/property_cache_json.log
291     """
292 
293     def _load_only(self):
294         with open(self._cache_filename, 'r') as fh:
295             self._cached_props = json.load(fh)
296         logger.debug('Loading properties from cache (%s)', self._cache_filename)
297 
298     def _save_only(self):
299         with open(self._cache_filename, 'w') as fh:
300             json.dump(self._cached_props, fh, sort_keys=True, indent=4)
301             logger.debug('cache-file stored (%s)', self._cache_filename)
```