Text

Description automatically generated **ISO/IEC JTC 1/SC 29/WG 3** N01737

**ISO/IEC JTC 1/SC 29/WG 3**

**MPEG Systems   
Convenorship: KATS (Korea, Republic of)**

**Document type:** Output Document

**Title:** Working Draft of ISO/IEC DIS 23090-43 Conformance and reference software for avatar representation format

**Status:** Approved

**Date of document:** 2026-01-30

**Source:** ISO/IEC JTC 1/SC 29/WG 3

**Expected action:**

**Action due date:** 2026-01-30

**No. of pages:** 19 (with cover page)

**Email of Convenor:** young.L@samsung.com

**Committee URL:** <https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg3>

**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 3**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC 1/SC 29/WG 3 N** **01737**

**January 2026 – Online**

|  |  |
| --- | --- |
| **Title** | **Working Draft of ISO/IEC 23090-43 Conformance and reference software for avatar representation format** |
| **Source** | **WG 03, MPEG Systems** |
| **Editor** | **João Regateiro (Interdigital)** |
| **Serial Number** | **25926** |

**ISO /IEC WD 23090-43**

ISO/IEC TC JTC 1/SC 29/WG 3

Date: 2026-01-30

**Information technology — Coded representation of immersive media — Part 43: Conformance and reference software for avatar representation format**

WD stage

**Warning for WDs and CDs**

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

© ISO 2026

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO’s member body in the country of the requester.

ISO copyright office

CP 401 • Ch. de Blandonnet 8

CH-1214 Vernier, Geneva

Phone: +41 22 749 01 11

Email: copyright@iso.org

Website: www.iso.org

Published in Switzerland

Contents

[Foreword vi](#_Toc220514597)

[Introduction vii](#_Toc220514598)

[1 Scope *(mandatory)* 1](#_Toc220514599)

[2 Normative references *(mandatory)* 1](#_Toc220514600)

[3 Terms and definitions *(mandatory)* 1](#_Toc220514601)

[4 Reference software for ISO/IEC-23090-39 MPEG-I Avatar Representation Format 1](#_Toc220514602)

[4.1 General 1](#_Toc220514603)

[4.2 Description 2](#_Toc220514604)

[4.3 Dependencies 2](#_Toc220514605)

[4.4 Usage 4](#_Toc220514606)

[4.4.1 arfref 4](#_Toc220514607)

[4.4.2 arfviewer 4](#_Toc220514608)

[4.5 Development 4](#_Toc220514609)

[4.5.1 Git repositories 4](#_Toc220514610)

[4.5.2 Build tools 5](#_Toc220514611)

[4.5.3 Assets organization 5](#_Toc220514612)

[4.5.4 Install arcore C++ library dependencies 5](#_Toc220514613)

[4.5.5 Build arcore C++ library 5](#_Toc220514614)

[4.5.6 Build arcore Python package 6](#_Toc220514615)

[4.5.7 Generator 6](#_Toc220514616)

[4.5.8 Issue tracking 7](#_Toc220514617)

[4.6 License 7](#_Toc220514618)

[5 ISO/IEC-23090-39 MPEG-I avatar representation format conformance 8](#_Toc220514619)

[5.1 Overview 8](#_Toc220514620)

[6 Conformance software for ISO/IEC-23090-39 MPEG-I avatar representation format 8](#_Toc220514621)

[6.1 Overview 8](#_Toc220514622)

[6.2 arf-validator 8](#_Toc220514623)

[6.2.1 Overview 8](#_Toc220514624)

[6.2.2 Software repository 8](#_Toc220514625)

[6.3 License 8](#_Toc220514626)

[7 References 8](#_Toc220514627)

[Annex A (informative) Annex title e.g. Example of a figure and a table 10](#_Toc220514628)

[A.1 Clause title 10](#_Toc220514629)

[A.1.1 Subclause title 10](#_Toc220514630)

[A.1.2 Subclause title 10](#_Toc220514631)

[A.1.2.1 Subclause title 10](#_Toc220514632)

[A.1.2.2 Subclause title 10](#_Toc220514633)

[A.2 Clause title 10](#_Toc220514634)

[Bibliography 11](#_Toc220514635)

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see [www.iso.org/directives](https://www.iso.org/directives-and-policies.html)).

ISO draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO takes no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO *[had/had not]* received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at [www.iso.org/patents](http://www.iso.org/patents). ISO shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see [www.iso.org/iso/foreword.html](https://www.iso.org/foreword-supplementary-information.html).

This document was prepared by Technical Committee ISO/IEC/JTC 1 *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

A list of all parts in the ISO 23090 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at [www.iso.org/members.html](https://www.iso.org/members.html).

Introduction

The conformance and reference software of ISO/IEC 23090-39 serves following main purposes:

* Validation of the written specification of the parts of ISO/IEC 23090-39;
* Clarification of the written specification of the parts of ISO/IEC 23090-39;
* Conformance testing for checking interoperability for the various applications against the reference software which aims to be complaint with ISO/IEC 23090-39

**Information technology — Coded representation of immersive media — Part 43: Conformance and reference software for avatar representation format**

# Scope

This document specifies the conformance and reference software implementing the normative clauses of ISO/IEC 23090‑39.

# Normative references *(mandatory)*

[Editor’s Note: to revisit.]

*Two options of text (remove the inappropriate option).*

*1) The normative references shall be introduced by the following wording.*

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO #####‑#, *General title — Part #: Title of part*

ISO #####‑##:20##, *General title — Part ##: Title of part*

*2) If no references exist, include the following phrase below the clause title:*

There are no normative references in this document.

# Terms and definitions *(mandatory)*

[Editor’s Note: to revisit.]

For the purposes of this document, the terms and definitions given in ISO/IEC 23090-39 apply.

No terms and definitions are listed in this document.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

* ISO Online browsing platform: available at <https://www.iso.org/obp>
* IEC Electropedia: available at <https://www.electropedia.org/>

3.1

term

text of the definition

Note 1 to entry: Text of the note.

[SOURCE: …]

3.2

term

text of the definition

# Reference software for ISO/IEC-23090-39 MPEG-I Avatar Representation Format

## General

The reference software is accessible form the link below: [MPEG / Systems / SceneDescription / ARF / Software / arfref · GitLab](https://git.mpeg.expert/MPEG/Systems/SceneDescription/arf/software/arfref)

An architectural diagram for the reference software is shown in Figure 1.

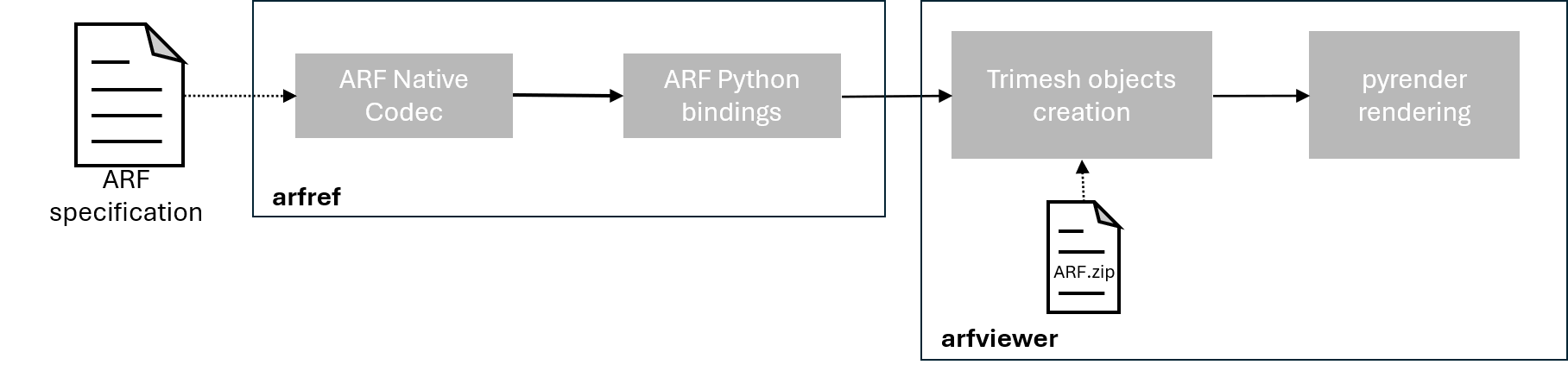


Figure 1 arfref reference software

## Description

The MPEG avatar representation format reference software is composed of two core software modules. The first module is the *arfref* module implemented in C++ and Python for reading ARF container files and streams. The second module is *arfviewer*, a rendering tool implemented in python.

The *arfref* module supports the following features:

* Parsing of arf containers
* Provides helper functions for decoding the assets
* Partial support for glTF 2.0 encoding and decoding of meshes.
* Animation mapping (e.g., “AnimationLink” objects).
* Animation stream decoding.

These features are available through the Python language.

The *arfviewer* module supports the following features:

* Avatar animation units (AAUs) as defined in ISO/IEC-23090-39 MPEG Avatar Representation Format.
  + Time-sequence blendshape weights, with optional confidence metrics for each blendshape,
  + Joint transformations for detailed skeletal animation
* The AAU format organizes data block chronologically, ensuring consistency and ease of parsing
* Inverse kinematics system to handle missing joint information from tracking system, ensuring plausible skeletal animations
* Blendshape animator module
  + Manages neutral mesh vertices alongside blendshape deltas,
  + Dynamically calculates vertex position through weighted summation.

## Dependencies

The following list of libraries are C++ dependencies for *arfref*.

|  |  |  |
| --- | --- | --- |
| **Library Package** | **Description** | **Version** |
| zlib | A Massively spiffy yet delicately unobtrusive compression library | 1.3.1 |
| libpng | The free reference library for reading and writing PNGs | 1.6.50 |
| libzip | A C library for reading, creating, and modifying zip archives | 1.11.4 |
| libjpeg-turbo | A JPEG image codec that uses SIMD instructions | 3.1.1 |

The following list of library packages are Python dependencies for *arfref*.

|  |  |  |
| --- | --- | --- |
| **Library Package** | **Description** | **Version** |
| Jinja2 | Jinja is a fast, expressive, extensible templating engine. | 3.1.6 |
| PyYAML | PyYAML is a YAML parser and emitter for Python. | 6.0.2 |
| build | The 'build' package is a simple Python build frontend. | 1.3.0 |
| colorama | Simple cross-platform API for printing colored terminal text | 0.4.6 |
| msgpack | messagepack is an efficient binary serialization format. | 1.1.1 |
| msgpack-numpy | msgpack-numpy provides routines for serializing and deserializing numpy data using msgpack | 0.4.8 |
| numpy | NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more. | 1.26.4 |
| packaging | The packaging library provides core utilities for Python packages | 25.0 |
| pillow | This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. | 11.3.0 |
| pybind11 | pybind11 is a lightweight header-only library that exposes C++ types in Python and vice versa, mainly to create Python bindings of existing C++ code. | 3.0.1 |
| pyproject\_hooks | This is a low-level library for calling build-backends in pyproject.toml-based project. | 1.2.0 |

The following list of Python packages are dependencies for *arfviewer*.

|  |  |  |
| --- | --- | --- |
| **Library Package** | **Description** | **Version** |
| PyOpenGL | Standard OpenGL bindings for Python | 3.1.10 |
| PyOpenGL-accelerate | Cython-coded accelerators for PyOpenGL | 3.1.10 |
| contourpy | Python library for calculating contours of 2D quadrilateral grids | 1.3.3 |
| cycler | Composable style cycles | 0.12.1 |
| fonttools | Tools to manipulate font files | 4.60.0 |
| freetype-py | Freetype python bindings | 2.5.1 |
| imageio | Library for reading and writing a wide range of image, video, scientific, and volumetric data formats. | 2.37.0 |
| imgui | Cython-based Python bindings for dear imgui | 2.0.0 |
| kiwisolver | A fast implementation of the Cassowary constraint solver | 1.4.9 |
| llvmlite | lightweight wrapper around basic LLVM functionality | 0.44.0 |
| matplotlib | Python plotting package | 3.10.6 |
| networkx | Python package for creating and manipulating graphs and networks | 3.5 |
| numba | compiling Python code using LLVM | 0.61.2 |
| numpy | Fundamental package for array computing in Python | 2.2.6 |
| packaging | Core utilities for Python packages | 25.0 |
| pillow | Python Imaging Library (Fork) | 11.3.0 |
| pyglet | pyglet is a cross-platform games and multimedia package. | 2.1.8 |
| pyparsing | pyparsing - Classes and methods to define and execute parsing grammars | 3.2.5 |
| pyrender | Easy-to-use Python renderer for 3D visualization | 0.3.1 |
| python-dateutil | Extensions to the standard Python datetime module | 2.9.0.post0 |
| scipy | Fundamental algorithms for scientific computing in Python | 1.16.1 |
| six | Python 2 and 3 compatibility utilities | 1.17.0 |
| trimesh | Import, export, process, analyze and view triangular meshes. | 4.8.1 |

## Usage

### arfref

The *arfref* software is available as a ready-to-build Python package, using the standard Python setup tools, assuming the C++ library dependencies (zlib, libpng, …) are made available.

### arfviewer

The *arfviewer* software is available as a git repository as indicated in clause 4.1. It requires the installation of the arfref wheel package. If no wheel package is available for the OS/Python version of the project, the wheel package needs to be created (see clause 4.5).

The *arfviewer* operates through a straightforward command-line interface:

|  |
| --- |
| python -m arfviewer path/to/my\_avatar.zip --animation path/to/weights.aau |

Where:

* *my\_avatar.zip* contains the referenced mesh assets and the arf document describing the avatar components and their relation.
* *--animation* initiates the animation stored in the supplied *weights.aau* animation file.

## Development

### Git repositories

The source code is available as indicated in clause 4.1. It is based on the following git repositories:

* arlib: the master repository, contains build scripts and git references to the other repositories (a.k.a. git submodules). This repository must be cloned in order to build the other components.
* arcore: the *arcore* software repository. It cannot be used alone, and must be cloned as a git submodule of arlib.
* arviewer: the *arviewer* software repository. If no development is required and if an *arcore* python package for the OS/Python version of the project is available, this repository can be directly cloned. Otherwise, it is better to clone it through arlib.

### Build tools

The following tools are mandatory building projects:

* Powershell
* 64-bit C++17 compiler
* CMake 3.21 or higher

For creating Python packages, conda is highly recommended. Any implementation that follows the conda API suits (Anaconda, miniconda, miniforge, etc.).

For creating C# libraries, a C# 13 compiler is required.

### Assets organization

Once the arlib repository and its git submodules are cloned, the following folders are available:

* avatar\_core: the *arcore* software repository.
  + cpp:
    - src: C++ library sources
    - test: C++ unit tests
    - run: C++ binaries (utilities for debugging/testing)
    - 3rdparty: Third-party C++ source code
  + python:
    - arcore: Python source code
    - pybind: Python bindings source code
    - test: Python unit tests
  + cs:
    - bindings : C# bindings source code
    - test : C# unit tests
  + test\_data: data for unit testing
  + gen:
    - generator: generator source code
    - specification: inputs for the generator
      * common: default definitions
      * arf: definition for ARF
      * gltf: definition for glTF 2.0
* avatar\_viewer: the *arviewer* software repository
* cache: stores third party library content, for instance, the sources code and building of libpng.
* build: build files, like CMake and object files.
* dist: release files and packages/archives. These files can be used for deployment.

### Install arcore C++ library dependencies

The Linux version requires the installation of zlib, libpng, libzip and libjpeg-turbo. For instance, for Ubuntu:

sudo apt install zlib1g-dev

sudo apt install libpng-dev

sudo apt install libzip-dev zipcmp zipmerge ziptool

sudo apt install libjpeg8-dev libjpeg-turbo8-dev libturbojpeg0-dev

For Windows, the build scripts will download and build these libraries.

### Build arcore C++ library

The arcore C++ library can be built using the following Powershell commands:

arlib root folder>cd avatar\_core\cpp

cpp> .\make.ps1 deps

cpp> .\make.ps1 configure

cpp> .\make.ps1 build

cpp> .\make.ps1 dist

The deps mode checks that dependencies are available. If not, it tries to download and build them (Windows). Rerun this command if dependencies changed.

The configure mode runs the CMake configuration. It must be run every time a source file was added or removed changed (output in <arlib folder>/build).

The build mode runs the CMake build (output in <arlib folder>/build).

The dist mode runs the CMake install (output in <arlib folder>/dist). The result is the headers and dynamic libraries and an archive with all those files.

### Build arcore Python package

As it is based on the C++ library, the C++ library dependencies must be installed (see clause 4.5.4). Manually building the C++ library is not required, as it is triggered by the following commands. If this automation fails, a manual building of the C++ library can be a solution.

It is highly recommended to create a python environment for building the package, with a preference for conda:

arlib root folder> cd avatar\_core\python

python> conda create -n avatar python==3.10.9

python> conda activate avatar

python> cd avatar\_core\python

python> pip install -r requirements.txt

The packages listed in <arlib folder>/python/requirements.txt are mandatory for building arcore Python package.

The arcore Python package can be built using the following Powershell commands:

arlib root folder>cd avatar\_core\python

cpp> .\make.ps1 deps

cpp> .\make.ps1 dist

The deps mode checks that dependencies are available and compiled, especially the arcore C++ library. If not, the dependencies and their dependencies are built. Rerun this command if dependencies changed, for instance if the source code of arcore C++ library changed.

The dist mode builds the Python package. The result is a wheel file available in <arlib folder>/dist.

For development of arcore Python source files, the dist mode can be replaced by the develop mode for convenience (e.g. run .\make.ps1 develop). In this later case, no package is created, and the Python source files are linked to the environment (e.g., --editable mode of pip install).

### Generator

The arcore software has source files generated by Python scripts (ARF and glTF codecs). The repository already contains the generated source files, no need to generate them unless modifications are required.

The generator files are in <arlib folder>\avatar\_core\gen\specification\{format}\types, where {format} is gltf or arf. There are JSON files in these folders which define the format properties.

Python env creation (once):

conda create -n avatar\_gen python==3.10.9

conda activate avatar\_gen

cd avatar\_core\gen

pip install -r requirements.txt

Run generator (example for ARF):

conda activate avatar\_gen

cd avatar\_core

.\gen.ps1 --spec arf

All languages are updated in this example (C, C++, C# and Python).

### Issue tracking

Issues related to the reference software are tracked in [REF]

## License

The copyright in the *arcore* reference software is being made available under the following licence:

|  |
| --- |
| The copyright in this software is being made available under the BSD License, included below. This software may be subject to other third party and contributor rights, including patent rights, and no such rights are granted under this license.  Copyright (c) <YEAR>, ISO/IEC SC29 WG03  All rights reserved.  Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:  \* Redistributions of source code must retain the above copyright notice,  this list of conditions and the following disclaimer.  \* Redistributions in binary form must reproduce the above copyright notice,  this list of conditions and the following disclaimer in the documentation  and/or other materials provided with the distribution.  \* Neither the name of the <ORGANIZATION> nor the names of its  contributors may be used to endorse or promote products derived from this  software without specific prior written permission.  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. |

The copyright in the *arviewer* software is being made available under the following licence:

|  |
| --- |
| This viewer is provided as-is for viewing ARF avatars. |

# ISO/IEC-23090-39 MPEG-I avatar representation format conformance

[Editor’s Note: No contributions.]

## Overview

Clause 5 of ISO/IEC 23090-39 defines features which describe methods for …[TODO].

[NOTE: There are no contributions to fill in this clause. clause 5 is purely illustrative and serves as reference template]

# Conformance software for ISO/IEC-23090-39 MPEG-I avatar representation format

## Overview

The software tool is written in X programming language. The tool can be used as a command line application. The tool outputs a validation report in X format.

A high-level design for the MPEG-I avatar representation format validation and conformance software is shown in Figure X. The conformance software includes a …

[Editor’s Note: There are no contributions to fill in this Clause. Clause 6 is purely illustrative and serves as reference template]

## arf-validator

### Overview

The arf-validator tool reads an arf container file and performs a schema check against the specification. Upon a validation check, the tool generates a validation report. The validation report lists potential issues and their severity in the arf document as shown in Figure X.

### Software repository

The MPEG arf-validator is accessible form the link below: [LINK]

## License

The copyright in this software is being made available under [License to add]

# References

1. Python venv, Online, <https://docs.python.org/3/library/venv.html>
2. (informative)  
     
   Annex title e.g. Example of a figure and a table
   1. Clause title

*Use subclauses if required e.g. A.1.1, A.1.2, etc. or A.1.1.1, A.1.1.2, etc.. For example:*

* + 1. Subclause title

Type text.

* + 1. Subclause title
       1. Subclause title

Type text.

* + - 1. Subclause title

Type text.

* 1. Clause title

An example of a figure is given in Figure A.1.

Bibliography

[1] ISO #####‑#, *General title — Part #: Title of part*

[2] ISO #####‑##:20##, *General title — Part ##: Title of part*