MinecraftElements

Release 0.4.4

Dana Hughes

Jan 14, 2022

CONTENTS:

CHAPTER

ONE

LICENSE

MIT License

Copyright (c) 2020 Dana Hughes

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

CHAPTER

TWO

OVERVIEW

The MinecraftElements module consists of a collection of enumerations of Minecraft elements. Enumerations are defined for most types of element (e.g., blocks, items, etc.)

This module is primarily designed to be used as a helper for other projects, and defines little in terms of functionality. However, by maintaining a single module, any dependent code will essentially be using a global enumeration of elements. In addition to the standard benefits of using enumerated values, this should help ensure that modules can interface easily without having to map enumerated (or worse, hardcoded) values of available Minecraft elements.

2.1 Enumeration Formats

Enumerations are provided for the elements listed below.

- Blocks
- Colors
- Entities
- Facing
- Flowers
- Items
- Monster Eggs
- Stones
- Woods

The enumeration for each element type is listed in a python file with the corresponding name. Enumerations match the names for elements provided by Project Malmo in the Types schema (see Malmo Types Schema). Each file contains a docstring with further details on the specific element.

Each enumeration extends Python's IntEnum, allowing enumerated elements to be treated as if they were integer values. Additionally, enumerated values are unique, so that there is a one-to-one mapping between enumerations and the corresponding values (i.e., no aliasing).

CHAPTER

THREE

INSTALLATION

3.1 Requirements

This module may be used with Python version 3.4 and later. Installation is best performed using pip.

3.2 Installation using pip

This module can be installed as a package using *pip*. Assuming *pip* is installed, the package can be installed (for users) from the root folder by:

```
pip install --user -e .
```

Once installed, the module can be imported from arbitrary locations.

3.3 Adding as a Submodule to a Project

As an alternative, this repository could be cloned into existing code as a *submodule*. This can be done with the following command:

```
git submodule add https://gitlab.com/cmu_asist/MinecraftElements
```

If the generated folder is empty, a recursive update may need to be performed:

```
git submodule update --init --recursive
```

(See Working with Submodules)

Additionally, cloning a repo with this submodule will require the *recursive* flag:

```
git clone --recursive <project url>
```

3.4 Documentation

Documentation for this module can be built using Sphinx, which is also installed with pip. The source and build files for the documentation is in the *docs* folder.

3.4.1 Requirements

Note that build targets may require additional dependencies to be installed. For building a PDF of the documentation using Latex, basic Latex dependencies need to be installed. On Ubuntu-based systems, this can be done by installing the following packages using *aptitude*:

```
apt-get install texlive-latex-recommended
texlive-latex-extra texlive-fonts-recommended
```

3.4.2 Building Documentation

To build a PDF of the *MinecraftElements Manual*, from the *docs* folder, run the build command with the latexpdf target:

```
cd docs
make latexpdf
```

This will create a *build* subfolder, and *latex* subfolder within build. The final pdf file, *MinecraftElements_Manual.pdf* will be written to the *latex* subfolder, which can be moved to the root directory easily. In Linux or OSX, this is done with:

```
mv build/latex/MinecraftElements_Manual.pdf ..
```

While in Windows, this would be done by:

```
move build\latex\MinecraftElements_Manual.pdf ..
```

CHAPTER

FOUR

USAGE

MinecraftElements is intended to serve as a simple enumeration and namespace for elements related to Minecraft (e.g., types of blocks). The enumerations use Python's *enum* module, allowing for flexible representation of elements (i.e., as strings, objects, or integers):

```
import MinecraftElements

someStoneBlock = MinecraftElements.Block.stone
```

Alternatively, specific (or all) components can be imported, if namespacing won't be an issue:

```
from MinecraftElements import *
someStoneBlock = Block.stone
```

Enumerated values can also be instantiated using the string name of the element through the use of the *__getitem__* method:

```
from MinecraftElements import Block
someStoneBlock = Block["stone"]
```

As the string names match those used in Malmo's *Types* schema, this should simplify conversion between data collected from Malmo to an enumeration.

Finally, enumerations can be treated as integer objects. This is useful for incorporating with certain types of data structures, e.g., numpy arrays:

```
from MinecraftElements import Block
import numpy as np

blockArray = np.array([Block.stone, Block.air, Block.stone], dtype=np.uint8)
```

Note that all enumerated values are non-negative, and less than 236, allowing for compatible representation using 8-bit unsigned bytes.

CHAPTER

FIVE

MINECRAFT ELEMENTS API

The MinecraftElements package provides a set of int enumerations of common Minecraft elements—blocks, entities, properties, etc. The enumerations allows for specific Minecraft concepts to be represented as an integer, string, or enumerated object, as well as ensuring common representation between projects, and avoiding subtle bugs generated by using incorrect numerical values or misspelled string.

5.1 Example Usage

MinecraftElement enumerations can be instantiated in several ways:

1. As an explicit enumeration:

```
>>> block = MinecraftElements.Block.clay
```

2. From strings:

```
>>> block = MinecraftElements.Block['clay']
```

3. From integer values:

```
>>> block = MinecraftElements.Block(82)
```

Conversely, an instance of the enumeration can be readily converted to the integer or string representation

1. To integer:

```
>>> blockInt = block.value
```

2. To string:

```
>>> blockStr = block.name
```

3. enumerations can be used in place of integer values, such as in a numpy array:

```
>>> import numpy as np
>>> blockArray = np.array([MinecraftElements.Block.Clay,
                           MinecraftElements.Block.dirt], dtype=np.uint16)
```

The available values of an enumeration can be determined by converting the enumeration to a list (or other collection):

```
>>> enumMembers = list(MinecraftElements.Block)
```

5.2 Enumerations

Enumeration-specific documentation and class definitions are provided for each enumeration in the module.

Enumeration of the blocks available in Minecraft. Enumerated names and values are such that they match the namespace ID and numeric ID utilized by Minecraft. Information about individual blocks is available in the Official Minecraft Block Wiki.

Additional ASIST-related block types (e.g., block_victim_1, marker_block, etc.) are also included. While enumerations for Minecraft specific blocks correspond to the numeric ID of the blocks used by Minecraft, the same does not hold true for ASIST-related block types. As such, any numeric ID below 256 should be considered an original Minecraft block, while those with value 256 and above are ASIST-related blocks.

Finally, an *UNKNOWN* block type is also included. This block type allows for graceful handling of blocks whose string values are not currently part of the Block enumeration Enumeration of the colors available in Minecraft. The colors in Minecraft consist of the following:

```
white
orange
magenta
light_blue
yellow
lime
pink
gray
silver
cyan
purple
blue
brown
green
red
black
```

Information about individual colors is available at Official Minecraft Dye Wiki. Enumeration of the entities available in Minecraft. Information about individual entities is available at Official Minecraft Entity Wiki. Enumeration of the facing property of blocks available in Minecraft.

The actual values that can be used are dependent on the type of block described, but consist of one of the following:

```
down
up
north
northwest
northeast
north-northwest
north-northeast
south
southwest
southeast
south-southwest
south-southeast
west
west-northwest
west-southwest
```

(continues on next page)

(continued from previous page)

```
east
east-northeast
east-southeast
up_x
up_z
down_x
down_z  Information
```

Information about block facing is available at Official Minecraft Block State Wiki. Enumeration of flower types available in Minecraft.

The flower types available in Minecraft consist of the following:

```
dandelion
poppy
blue_orchid
allium
houstonia
red_tulip
orange_tulip
white_tulip
pink_tulip
oxeye_daisy
```

Information about flowers is available at Official Minecraft Flower Wiki. Enumeration of block half properties available in Minecraft.

The half property for blocks can take on one of the following six values, which are paired by complement:

```
top / bottom
head / foot
upper / lower
```

Information about half properties is available at Official Minecraft Block State Wiki. Enumeration of block hinge properties available in Minecraft. The hinge property is used by door blocks, and can take on one of the value of *left* or *right*.

Information about hinge properties is available at Official Minecraft Block State Wiki. Enumeration of items available in Minecraft.

Information about items is available at Official Minecraft Item Wiki. Enumeration of types of monster eggs available in Minecraft. MonsterEgg types match stone types, and can take one of the following values:

```
cobblestone
stone_brick
mossy_brick
cracked_brick
chiseled_brick
```

Information about items is available at Official Minecraft Monster Egg Wiki. Enumeration of types of shapes used in Minecraft. Shapes are block properties for stairs in Minecraft, and can take on eof the following values:

```
straight
inner_left
inner_right
```

(continues on next page)

(continued from previous page)

```
outer_left
outer_right
```

Information about items is available at Official Minecraft Stairs Wiki. Enumeration of stone types available in Minecraft.

The stone types available in Minecraft consist of the following:

```
stone
granite
smooth_granite
diorite
smooth_diorite
adensite
smooth_adensite
```

Information about stone types is available at Official Minecraft Stone Wiki. Enumeration of wood types available in Minecraft.

The wood types available in Minecraft consist of the following:

```
oak
spruce
birch
jungle
acacia
dark_oak
```

Information about wood types is available at Official Minecraft Wood Wiki.

5.2.1 Class Definitions

class MinecraftElements.blocks.**Block**(*value*)
Enumeration of block types available in Minecraft and ASIST-related blocks.

The Block enumeration extends IntEnum, so that individual elements can be treated as one would treat any other int. Additionally, Block enumerations are unique.

class MinecraftElements.colors.**Color**(*value*)
Enumeration of colors available in Minecraft.

The Color enumeration extends IntEnum, so that individual elements can be treated as one would treat any other int.

class MinecraftElements.entities.**Entity**(*value*)
Enumeration of entities available in Minecraft.

The Entity enumeration extends IntEnum, so that individual elements can be treated as one would treat any other int.

class MinecraftElements.facing.**Facing**(*value*)
Enumeration of facings available in Minecraft.

The Facing enumeration extends IntEnum, so that individual elements can be treated as one would treat any other int.

class MinecraftElements.flowers.**Flower**(*value*)
Enumeration of flowers available in Minecraft.

The Flower enumeration extends IntEnum, so that individual elements can be treated as one would treat any other int.

class MinecraftElements.halfs.**Half**(*value*)
Enumeration of halves available in Minecraft. Half values are used as attributes to specific block types (e.g., doors, beds).

The Half enumeration extends IntEnum, so that individual elements can be treated as one would treat any other int.

class MinecraftElements.hinge.**Hinge**(*value*)
Enumeration of hinging available in Minecraft. Hinges are attributes of door blocks, and can take the value of 'left' or 'right'.

The Hinge enumeration extends IntEnum, so that individual elements can be treated as one would treat any other int.

class MinecraftElements.items.**Item**(*value*)
Enumeration of items available in Minecraft.

The Item enumeration extends IntEnum, so that individual elements can be treated as one would treat any other int.

NOTE: Malmo lists two instances of 'bucket' in it's schema; only one instance of 'bucket' is implemented in this enumeration.

class MinecraftElements.monster_eggs.**MonsterEgg**(*value*)
Enumeration of monster eggs available in Minecraft. MonsterEggs are blocks that release a monster when destroyed.

The MonsterEgg enumeration extends IntEnum, so that individual elements can be treated as one would treat any other int.

class MinecraftElements.shapes.**Shape**(*value*)
Enumeration of shapes available in Minecraft. Shape values are attributes used by stairs.

The Shape enumeration extends IntEnum, so that individual elements can be treated as one would treat any other int.

class MinecraftElements.stones.**Stone**(*value*)
Enumeration of stone types available in Minecraft.

The Stone enumeration extends IntEnum, so that individual elements can be treated as one would treat any other int.

class MinecraftElements.woods.**Wood**(*value*)
Enumeration of wood types available in Minecraft.

The Wood enumeration extends IntEnum, so that individual elements can be treated as one would treat any other int.

CHAPTER

SIX

CHANGELOG

- 29 June 2021
  - Updated Python docstrings to work with Sphinx
  - Added Sphinx Documentation

CHAPTER

SEVEN

INDICES AND TABLES

PYTHON MODULE INDEX

INDEX

S

W