Agentverse: allowed imports

Introduction

In theAgentverse ↗(opens in a new tab) code editor, you have the freedom to import and utilize a selected set of modules to create your code, while maintaining security and control. These pre-approved modules offer a diverse range of functionalities, allowing you to build complex agents.

The modules available on the Agentverse are :

- uagents ↗
- .
- json ↗
- .
- requests ↗
- .
- cosmpy ↗
- .
- datetime ↗
- .
- enum ↗
- .
- pydantic ↗
- .
- math ↗
- .
- time ↗
- .
- uuid ↗
- .
- ai_engine ↗
- .
- random ↗
- .
- MySQLdb ↗
- .
- pymongo ↗
- .
- _strptime ↗
- .
- hashlib ↗
- .
- hmac ↗
- .
- base64 ↗
- .
- re ↗
- .

Allowed imports

uagents

Build fast and lightweight for decentralized scenarios using theuagents Framework. Checkout the↗ documentation and theuagents package ↗(opens in a new tab) for more information.

- Available classes
- :Model
- ,Context
- ,Protocol
- .
- Example
- :
- from
- uagents
- import
- Context

- ,
- Model
- class
- Message
- (
- Model
-):
- text
- :
- str
- @agent
- .
- on_interval
- (period
- =
- 2.0
-)
- async
- def
- print_message
- (
- ctx
- :
- Context):
- msg
- =
- Message
- (text
- =
- f
- "Hello there my wallet address is
- {
- ctx.wallet
- }
- ."
-)
- print
- (msg.text)

json

Simply interact with JSON data structures. This package provides functionality for encoding and decoding JSON (JavaScript Object Notation) data.

- Available functions
- :dumps
- ,loads
- .
- Example
- :
- import
- json
- some_json
- =
- '{ "agent":"Alice", "age":"1 day"}'
- # JSON to python dictionary

- python_dict
- =
- json
- .
- loads
- (some_json)
- print
- (python_dict[
- "age"

-])

- ## python dictionary to JSON:

- some_json_2
- =
- json
- .
- dumps
- (python_dict)

- ## the result is a JSON string equal to some_json:

- print
- (some_json_2)

requests

This package allows you to interact with HTTP requests and responses.

- Available functions
- :get
- ,post
- ,put
- ,patch
- ,delete
- .
- Example
- :
- import
- requests
- response
- =
- requests
- .
- get
- (
- 'https://api.github.com'
-)
- if
- response
- .
- status_code
- ==
- 200
- :
- print
- (
- 'Success!'
-)
- elif
- response
- .
- status_code
- ==
- 404
- :
- print
- (
- 'Not Found.'
-)
- print
- (response.headers)

cosmpy

A Python library for interacting with Cosmos-based blockchains. Checkout the[CosmPy ↗](#) documentation and the[CosmPy package ↗(opens in a new tab)](#) for more information.

- Full access to all functions and features
- .
- Example
- :
- from
- cosmpy
- import
- aerial

Define network configuration, faucet and ledger

- network
- =
- aerial
- .
- client
- .
- NetworkConfig
- .
- fetchai_stable_testnet
- ()
- faucet_api
- =
- aerial
- .
- faucet
- .
- FaucetApi
- (network)
- ledger
- =
- aerial
- .
- client
- .
- LedgerClient
- (network)
- MINIMUM_BALANCE
- =
- 100000000000000000
- @agent
- .
- on_interval
- (period
- =
- 20.0
-)
- async
- def
- get_tokens
- (
- ctx
- :
- Context):
- agent_balance
- =
- ledger
- .
- query_bank_balance
- (ctx.wallet)
- if
- agent_balance
- <
- MINIMUM_BALANCE

- :
- print
- (
- "Providing wealth to agent..."
-)
- faucet_api
- .
- get_wealth
- (ctx.wallet)

datetime

A library allowing you to handle date and time operations with ease.

- Full access to all functions and features
- .
- Example
- :
- import
- datetime
- today
- =
- datetime
- .
- datetime
- .
- now
- ()
- print
- (
- f
- "Today is:
- {
- today.year
- }
- :
- {
- today.month
- }
- :
- {
- today.day
- }
- "
-)

enum

A library to create enumerations with symbolic names and unique values.

- Full access to all functions and features
- .
- Example
- :
- from
- enum
- import
- Enum
- class
- Agent
- (
- Enum
-):
- alice
- =
- 1
- bob
- =

- 2
- carl
- =
- 3
- print
- (Agent.bob.value)

pydantic

A package to ensure data validation and settings management. It simplifies the process of defining and validating data models by providing a way to declare and enforce data types, constraints, and validation rules on Python data structures.

- Full access to all functions and features
- .
- Example
- :
- from
- pydantic
- import
- BaseModel
- data
- =
- {
- "name"
- :
- "alice"
- ,
- "age"
- :
- 21
- }
- class
- User
- (
- BaseModel
-):
- name
- :
- str
- age
- :
- int
- user
- =
- User
- (

- data)
- print
- (user)

math

A library that provides access to various mathematical functions and constants for performing mathematical operations.

- Full access to all functions and features
- .
- Example
- :
- import
- math
- # Basic mathematical operations
- print
- (math.

- sqrt
- (
- 25
-))

- ## Output: 5.0

- print
- (math.
- pow
- (
- 2
- ,
- 3
-))

- ## Output: 8.0
- ## Trigonometric functions

- print
- (math.
- cos
- (
- 0
-))

- ## Output: 1.0
- ## Logarithmic and exponential functions

- print
- (math.
- log
- (
- 10
-))

- ## Output: 2.302585092994046

time

This provides functions for working with time-related functions, including time measurement, manipulation, and conversions. It allows Python programs to handle time expressions, perform time-related calculations, and work with time values in different formats.

- Full access to all functions and features
- .
- Example
- :
- import
- time
- current_time
- =
- time
- .
- strftime
- (
- "%H:%M:%S"
-)

- ## Get the current time in HH:MM:SS format

- # Get the current time in HH:MM:SS format

- print
- (
- "Current time is:"
- , current_time)

uuid

It stands for "Universally Unique Identifier". It is a Python library used for generating universally unique identifiers based on various algorithms. These identifiers are unique across space and time, making them highly suitable for various purposes where uniqueness is essential, such as database keys, identifiers in distributed systems, and more.

- Full access to all functions and features
- .
- Example
- :
- import
- uuid

- # Generating a version 4 (random) UUID

- unique_id
- =
- uuid
- .
- uuid4
- ()
- print
- (unique_id)

ai_engine

Integrate Agents with the AI Engine to perform a wide variety tasks, including in booking services, make reservations, and provide answers to different queries. Checkout theAI Engine ↗ documentation and theAI Engine package ↗(opens in a new tab) for further information.

- Full access to all functions and features
- .
- Example
- :
- from
- uagents
- import
- Context
- ,
- Model
- ,
- Protocol
- from
- ai_engine
- import
- UAgentResponse
- ,
- UAgentResponseType
- simples
- =
- Protocol
- (name
- =
- "simples"
- , version
- =
- "v1.1"
-)
- class
- Request

- (
- Model
-):
- message
- :
- str
- @simples
- .
- on_message
- (model
- =
- Request, replies
- =
- {UAgentResponse})
- async
- def
- handle_message
- (
- ctx
- :
- Context
- ,
- sender
- :
- str
- ,
- msg
- :
- Request):
- await
- ctx
- .
- send
- (sender,
- UAgentResponse
- (message
- =
- "0"
- , type
- =
- UAgentResponseType.FINAL))

random

This package is used for generating random numbers, managing random selections, and handling random data. It provides various functions for generating random values, shuffling sequences, and making random selections.

- Full access to all functions and features
- .
- Example
- :

- # Generate a random integer within a specified range

- random_integer
- =
- random
- .
- randint
- (
- 1
- ,
- 10
-)
- print
- (random_integer)

- # Output: (any integer between 1 and 10)
- # Shuffle a list

- my_list
- =
- [
- 1
- ,
- 2
- ,
- 3
- ,
- 4
- ,
- 5
-]
- random
- .
- shuffle
- (my_list)
- print
- (my_list)

- # Output: [3, 1, 4, 5, 2] (shuffled list)
- # Select a random item from a list

- my_item
- =
- random
- .
- choice
- (my_list)
- print
- (my_item)

- # Output: (any element from the list)

MySQLdb

MySQLdb is a Python library for accessing MySQL databases. It provides a Python interface to MySQL, allowing you to interact with MySQL databases from within your Python code.

- Full access to all functions and features
- .
- Example
- :
- import
- MySQLdb

- # Connect to the MySQL database

- connection
- =
- MySQLdb
- .
- connect
- (host
- =

- 'localhost'
- , user
- =
- 'username'
- , passwd
- =
- 'password'
- , db
- =
- 'database_name'
-)
- try
- :
- ## Create a cursor object to execute SQL queries
- cursor
- =
- connection
- .
- cursor
- ()
- ## Example query: Select all rows from a table
- cursor
- .
- execute
- (
- "SELECT * FROM your_table"
-)
- ## Print the fetched rows
- for
- row
- in
- cursor
- .
- fetchall
- ():
- print
- (row)
- finally
- :
- ## Close the cursor and connection
- cursor
- .
- close
- ()
- connection
- .
- close
- ()

pymongo

pymongo allows Python applications to interact with MongoDB databases, making it easy to perform various database operations such as inserting, updating, deleting, and querying documents.

- Full access to all functions and features
- .

- Example
- :
- from
- pymongo
- import
- MongoClient

- ## Connect to MongoDB server

- client
- =
- MongoClient
- (
- 'mongodb://localhost:27017/'
-)

- ## Access a specific database

- db
- =
- client
- [
- 'my_database'
-]

- ## Access a specific collection within the database

- collection
- =
- db
- [
- 'my_collection'
-]

- ## Insert a document into the collection

- document
- =
- {
- 'name'
- :
- 'John'
- ,
- 'age'
- :
- 30
- ,
- 'city'
- :
- 'New York'
- }
- collection
- .
- insert_one
- (document)

- ## Query documents from the collection

- query
- =
- {
- 'city'

- :
- 'New York'
- }
- result
- =
- collection
- .
- find
- (query)

- ## Print the documents returned by the query

- for
- doc
- in
- result
- :
- print
- (doc)

- ## Close the connection to MongoDB

- client
- .
- close
- ()

_strptime

_strptime is a method from thedatetime module in Python, specifically used for parsing date and time strings into datetime objects.Example :

from datetime import datetime

date_string

"2024-04-23" format_string =

"%Y-%m- %d " date_object = datetime . strptime (date_string, format_string) print (date_object)

hashlib

hashlib is a Python library that provides interfaces to various secure hash and message digest algorithms. It's commonly used for generating hash values for data integrity verification, password hashing, digital signatures, and more.

- Full access to all functions and features
- .
- Example
- :
- import
- hashlib

- ## String to hash

- data
- =
- "Hello, world!"

- ## Create a SHA-256 hash object

- hash_object
- =

- hashlib
- .
- sha256
- ()

- ## Update the hash object with the data

- hash_object
- .
- update
- (data.
- encode
- ())

- ## Get the hexadecimal digest (hash value) of the data

- hash_value
- =
- hash_object
- .
- hexdigest
- ()
- print
- (
- "SHA-256 Hash:"
- , hash_value)

hmac

hmac stands forHash-based Message Authentication Code . It's a Python module that implements the HMAC algorithm as described in RFC 2104. HMAC is a cryptographic hash function that adds a shared secret key to the hashing process, providing a way to verify both the integrity and authenticity of a message.

- Full access to all functions and features
- .
- Example
- :
- import
- hmac
- import
- hashlib

- ## Secret key (shared between sender and receiver)

- secret_key
- =
- b
- 'my_secret_key'

- ## Message to authenticate

- message
- =
- b
- 'Hello, world!'

- ## Generate HMAC using SHA-256 hash function and secret key

- hmac_digest
- =

- hmac
- .
- new
- (secret_key, message, hashlib.sha256).
- hexdigest
- ()
- print
- (
- "HMAC:"
- , hmac_digest)

base64

base64 is a Python module that provides functions for encoding and decoding binary data using the Base64 encoding scheme. Base64 encoding converts binary data into a ASCII string format, which is useful for transmitting binary data over protocols that require plain text, such as email or HTTP headers.

- Full access to all functions and features
- .
- Example
- :
- import
- base64
- ## String to encode
- data
- =
- "Hello, world!"
- ## Encode the string to Base64
- encoded_data
- =
- base64
- .
- b64encode
- (data.
- encode
- ())
- print
- (
- "Encoded:"
- , encoded_data)
- ## Decode the Base64-encoded data
- decoded_data
- =
- base64
- .
- b64decode
- (encoded_data).
- decode
- ()
- print
- (
- "Decoded:"
- , decoded_data)

re

re is a Python module providing support for working with regular expressions. Regular expressions (often abbreviated as regex or regexp) are sequences of characters that define a search pattern, which is useful for finding, replacing, or manipulating strings based on specific patterns.

- Full access to all functions and features
- .
- Example
- :
- import
- re
- # String containing a pattern
- text
- =
- "The quick brown fox jumps over the lazy dog."
- # Define a regular expression pattern
- pattern
- =
- r
- 'fox'
- # Search for the pattern in the string
- match
- =
- re
- .
- search
- (pattern, text)
- if
- match
- :
- print
- (
- "Pattern found at index:"
- , match.
- start
- ())
- else
- :
- print
- (
- "Pattern not found."
-)

Multi-file Support

The Agentverse Code Editor enhances your agent development experience with multi-file support, enabling you to tackle complex projects with ease. Leverage this feature to:

- Interact between files
- : simply import functions, classes, and variables from one file to another.
- Modular development
- : divide your projects into manageable components for streamlined creation.
- Code reuse
- : utilize modules across various sections of your project for efficient development.
- Enhanced organization
- : maintain a structured and organized codebase for easier maintenance.

If you want to createnew files you just need to click on+ New File onAgentverse ↗(opens in a new tab) inside your managed agent.

You can create a Python message file with the followingModel class:

from uagents import Model

class

Sentence (Model): text :

str Then, you can just import theSentence data model to youragent.py file and make use of it:

from uagents import Context from message import Sentence

@agent . on_interval (period = 2.0) async

def

print_message (ctx : Context): msg =

Sentence (text = f "Hello there my wallet address is { ctx.wallet } .") print (msg.text) Explore theAgentverse guides ↗ andAgentverse concepts ↗ for additional guides and documentation resources!

For additional information on services, head over toAgentverse Services ↗ .

Was this page helpful?

Agentverse: Dice Roll agent Agentverse: Mailbox