Embrace AI: Kata 1

Rinat Abdullin

Exported on 2025-08-01 07:06:31

Table of Contents

1 TimeToAct DocumentAI Spec 4

1.1 Document Layout 4

2 Specifications 6

2.1 Empty text 6

2.2 Body 6

2.3 Head 6

2.4 Blocks 7

2.5 Dictionaries 7

3 Lists 9

3.1 Ordered Lists 9

3.2 Unordered lists 10

3.3 Mixed lists 11

3.4 Lists with content 12

|  |
| --- |
| AI in Coding helps a lot with high-level tasks like prototyping, reasoning and finding bugs.  Please use your favourite tools (no limitations) to implement as much of this spec as possible in a language of your choice. Imagine that your team will have to support this code for a few years, so you want to do a thorough job here. |

# TimeToAct DocumentAI Spec

Let's define a simple document format that could describe contract, procedure or any other business document in a **structured way**. It may be used to load this business data into AI Assistants like in ERC.

We’ll work with the documents.

Our documents will consist from blocks. Block is a logical piece of text like pagragraph. It can optionally have a head, number and body. Block body could contain:

* another block
* text
* list
* dictionary

Blocks could contain heterogenous content - texts, other blocks, dictionaries.

List could contain only similar block items that also have number.

## Document Layout

The document below describes a simple text format that could be deterministically parsed into JSON objects. This document is also a test suite! Code admonitions always come in pairs: first input and then json.

When parser is implemented, parsed input should always produce output that is structurally similar to the expected json. Headline before the code blocks is the name of the text.

Data structures to parse document into could look like this in python:

|  |
| --- |
| **from** typing **import** List, Optional, Union, Dict, Literal  **from** pydantic **import** BaseModel, Field    # This type alias helps with readability and forward references.  ContentNode **=** Union[str, "Block", "ListBlock", "Dictionary"]      **class** Dictionary(BaseModel):  """  A distinct dictionary structure for key-value pairs.  """  kind: Literal["dict"]  items: Dict[str, str] **=** Field(default\_factory**=**dict)      **class** Block(BaseModel):  """  A general**-**purpose container **for** a 'section' **or** item.    **-** 'number' can store a section number (e.g., "5", "5.1") **if** applicable.  **-** 'head' **is** an optional heading **for** the block.  **-** 'body' can hold any mix of strings, sub**-**blocks, dictionaries, **or** lists.  """  kind: Literal["block"]  number: Optional[str] **=** None  head: Optional[str] **=** None  body: List[ContentNode] **=** Field(default\_factory**=**list)      **class** ListBlock(BaseModel):  """  A container **for** a list of items, each item being a 'Block'.  """  kind: Literal["list"]  items: List[Block] **=** Field(default\_factory**=**list)      # Important for forward references within union types  Block.model\_rebuild() |

# Specifications

## Empty text

Empty text results in an empty document block

|  |
| --- |
|  |

|  |
| --- |
| {  "kind": "block"  } |

## Body

Plain text goes into the block body straight away. Different paragraphs are separated by the new lines.

|  |
| --- |
| First paragraph.  Second paragraph. |

It will be parsed into:

|  |
| --- |
| {  "kind": "block",  "body": [  "First paragraph.",  "Second paragraph."  ]  } |

Note, that we strip and skip empty lines!

|  |
| --- |
| First paragraph.    Second paragraph. |

results in:

|  |
| --- |
| {  "kind": "block",  "body": [  "First paragraph.",  "Second paragraph."  ]  } |

## Head

Text marked with <head> goes directly into the head of the current block.

|  |
| --- |
| <head>Test Document</head>  Content |

|  |
| --- |
| { "kind": "block",  "head": "Test Document",  "body": [  "Content"  ]  } |

## Blocks

You've seen that the document is parsed in a root block. But everything is a block and blocks can be nested explicitly

|  |
| --- |
| <head>AI Coding Kata</head>  Let's get started with the kata  <block>  <head>Preface</head>  Here is a little story  </block> |

This is how things get extracted:

|  |
| --- |
| {  "kind": "block",  "head": "AI Coding Kata",  "body": [  "Let's get started with the kata",  {  "kind": "block",  "head": "Preface",  "body": [  "Here is a little story"  ]  }  ]  } |

## Dictionaries

Dictionaries are used to capture key-value pairs, by default they are separated by ::

|  |
| --- |
| <dict sep=":">  Key One: Value One  Key Two: Value Two  Key Three: Value Three  </dict> |

this will be parsed into:

|  |
| --- |
| {  "kind": "block",  "body": [  {  "kind": "dict",  "items": {  "Key One": "Value One",  "Key Two": "Value Two",  "Key Three": "Value Three"  }  }  ]  } |

But we can also have non-standard separator and empty values.

|  |
| --- |
| <dict sep="-">  Title - AI Coding - for TAT  Kata Number -  </dict> |

|  |
| --- |
| {  "kind": "block",  "body": [  {  "kind": "dict",  "items": {  "Title": "AI Coding - for TAT",  "Kata Number": ""  }  }  ]  } |

# Lists

Lists are very important! By default, each non-empty line is a list item. They go inside the root block. There are multiple kinds:

* . for ordered lists that are dot-separated
* \* for bulleted lists

Note, that list item goes to head and number goes to number.

## Ordered Lists

|  |
| --- |
| <list kind=".">  1. First  2. Second  </list> |

|  |
| --- |
| {  "kind": "block",  "body": [  {  "kind": "list",  "items": [  { "kind": "block", "number": "1.", "head": "First" },  { "kind": "block", "number": "2.", "head": "Second" }  ]  }  ]  } |

As a convenience, nested lists are automatically detected.

|  |
| --- |
| <list kind=".">  1. First  2. Second  2.1. Subitem 1  2.2. Subitem 2  </list> |

|  |
| --- |
| {  "kind": "block",  "body": [  {  "kind": "list",  "items": [  { "kind": "block", "number": "1.", "head": "First" },  { "kind": "block", "number": "2.", "head": "Second", "body": [  { "kind": "list", "items": [  { "kind": "block", "number": "2.1.", "head": "Subitem 1" },  { "kind": "block", "number": "2.2.", "head": "Subitem 2" }  ] }  ] }  ]  }  ]  } |

## Unordered lists

We can have unordered lists, too:

|  |
| --- |
| <list kind="\*">  • First  • Second    • Third  </list> |

They get parsed like this:

|  |
| --- |
| {  "kind": "block",  "body": [  {  "kind": "list",  "items": [  { "kind": "block", "number":"•", "head": "First" },  { "kind": "block", "number":"•", "head": "Second" },  { "kind": "block", "number":"•", "head": "Third" }  ]  }  ]  } |

And nesting can be done with "o"

|  |
| --- |
| <list kind="\*">  • First  o Subitem  • Second  • Third  </list> |

this would be loaded as:

|  |
| --- |
| {  "kind": "block",  "body": [  {  "kind": "list",  "items": [  { "kind": "block", "number":"•", "head": "First", "body": [  { "kind": "list", "items": [  { "kind": "block", "number":"o", "head": "Subitem" }  ] }  ] },  { "kind": "block", "number":"•", "head": "Second" },  { "kind": "block", "number":"•", "head": "Third" }  ]}  ]} |

## Mixed lists

We can mix lists, but would need to designate different types separately with tags.

|  |
| --- |
| <list kind=".">  1. Beginning  2. Main  2.1. Subsection  <list kind="\*">  \* Bullet 1  \* Bullet 2  </list>  3. Ending  </list> |

this would be parsed into:

|  |
| --- |
| {  "kind": "block",  "body": [  {  "kind": "list",  "items": [  { "kind": "block", "number":"1.", "head": "Beginning" },  { "kind": "block", "number":"2.", "head": "Main", "body": [  { "kind": "list", "items": [  { "kind": "block", "number":"\*", "head": "Bullet 1" },  { "kind": "block", "number":"\*", "head": "Bullet 2" }  ] }  ] },  { "kind": "block", "number":"2.1.", "head": "Subsection" },  { "kind": "block", "number":"3.", "head": "Ending" }  ]  }  ]  } |

## Lists with content

Obviously, lists can have additional content. If something in the current list doesn't match the prefix, then it is treated as block body:

|  |
| --- |
| <list kind=".">  1. First  First body  2. Second  Some more text  <dict sep=":">  Key: Value  Another Key: Another Value  </dict>  </list> |

it would parse into:

|  |
| --- |
| {  "kind": "block",  "body": [  {  "kind": "list",  "items": [  { "kind": "block", "number":"1.", "head": "First", "body": [  "First body"  ] },  { "kind": "block", "number":"2.", "head": "Second", "body": [  "Some more text",  {  "kind": "dict",  "items": {  "Key": "Value",  "Another Key": "Another Value"  }  }  ] }  ]  }  ]  } |