API Guidance and Best Practices

Alex Kaszynski
Roberto Pastor Muela

November 15, 2022

Ansys

Table of Contents

Remote API - Overview

- Today, APIs are playing an increasingly important role in our lives.
- They provide the means for our computers to interact with the outside world, and they allow us to access information and services that we wouldn’t otherwise be able to use.
- In the future, they will become even more important, as they will be used to control and manage our increasingly complex systems.

API Example

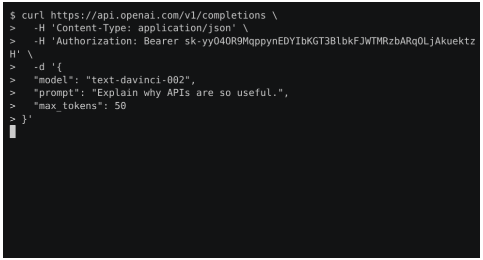
This slide was entirely generated from an API based on this statement:
“Public software APIs will be so fundamental to our modern products that will be unable to comprehend how we survived without them.”

OpenAI API Query

- POST a simple API query to OpenAI's GPT-3 API.
- Using the API defined at OpenAI GPT-3 - API to submit a `curl` request.

OpenAI API

```
curl https://api.openai.com/v1/completions \
  -H 'Content-Type: application/json' \
  -H 'Authorization: Bearer YOUR_API_KEY' \
  -d '{
  "model": "text-davinci-002",
  "prompt": "Say this is a test",
  "max_tokens": 6,
}'
```



```
$ curl https://api.openai.com/v1/completions \
>   -H 'Content-Type: application/json' \
>   -H 'Authorization: Bearer sk-yyO4OR9MqppynEDYIbKGT3BlbkFJWTMRzbARqOLjAkuektz
H' \
>   -d '{
>   "model": "text-davinci-002",
>   "prompt": "Explain why APIs are so useful.",
>   "max_tokens": 50
> }'
```

OpenAI API Query - Python

Alternatively, you can use their Python client library to submit a request:

OpenAI API

```
>>> import os
>>> import openai
>>> openai.api_key = os.getenv("OPENAI_API_KEY")
>>> response = openai.Completion.create(
...     model="text-davinci-002",
...     prompt="What is the answer to life the universe and everything?",
...     max_tokens=6,
...     temperature=1.0,
... )
>>> response['choices'][0]['text'].strip()
'42'
```

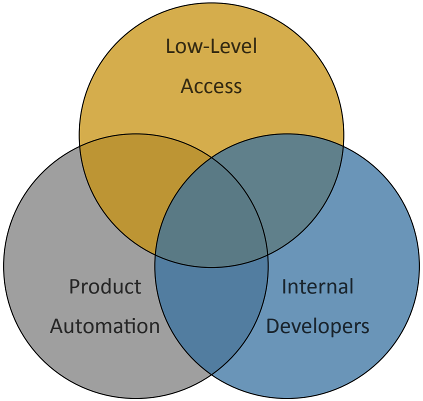
APIs - Problem Space

What do customers want? Let’s consider two kinds of customers who might want APIs:

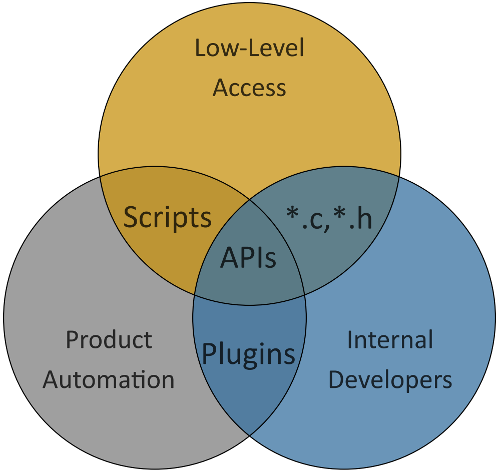
- Those who want to automate repetitive workflows.
- Those who need low-level access to the underlying libraries and components that make up an Ansys product.

What do developers want?

- Well documented low-level interface to foreign libraries to avoid duplicating work.



APIs - Solution Space



APIs - Definition and Examples

APIs - Definition and Examples

API Definition

API stands for “Application Programming Interface” and is a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.

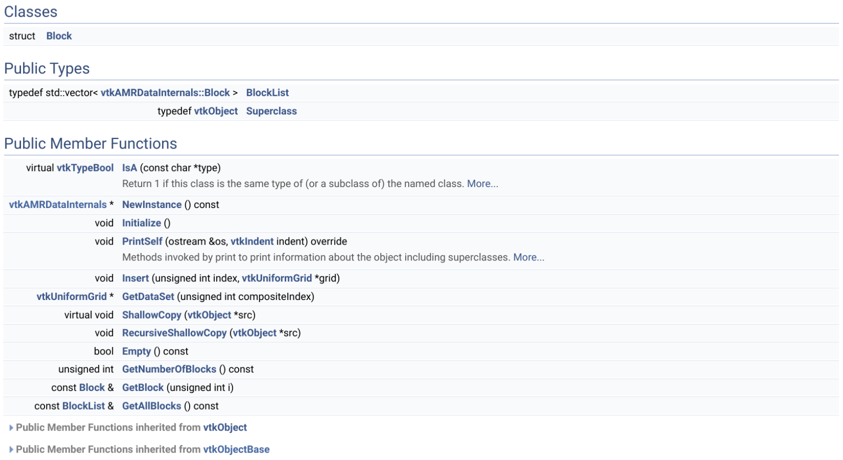
Examples

- A C header file
- Public Python class function
- RESTful HTTP API (POST, GET)

In essence, APIs let the users know how to interact with a given library or network interface without containing the specifics of the implementation.

APIs in the Wild - C++ API

VTK - API Reference



Classes

struct	**Block**

Public Types

typedef std::vector< **vtkAMRDataInternals::Block** >	**BlockList**
typedef **vtkObject**	**Superclass**

Public Member Functions

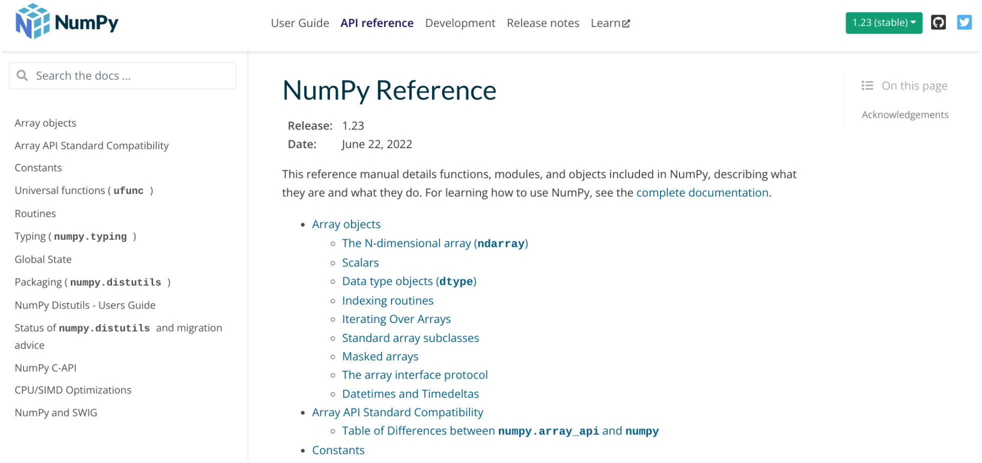
virtual **vtkTypeBool**	**IsA** (const char *type) Return 1 if this class is the same type of (or a subclass of) the named class. More...
vtkAMRDataInternals *	**NewInstance** () const
void	**Initialize** ()
void	**PrintSelf** (ostream &os, **vtkIndent** indent) override Methods invoked by print to print information about the object including superclasses. More...
void	**Insert** (unsigned int index, **vtkUniformGrid** *grid)
vtkUniformGrid *	**GetDataSet** (unsigned int compositeIndex)
virtual void	**ShallowCopy** (**vtkObject** *src)
void	**RecursiveShallowCopy** (**vtkObject** *src)
bool	**Empty** () const
unsigned int	**GetNumberOfBlocks** () const
const **Block** &	**GetBlock** (unsigned int i)
const **BlockList** &	**GetAllBlocks** () const

▸ Public Member Functions inherited from **vtkObject**

▸ Public Member Functions inherited from **vtkObjectBase**

APIs in the Wild - Python API

NumPy - API Reference



Array objects
Array API Standard Compatibility
Constants
Universal functions (**ufunc**)
Routines
Typing (**numpy.typing**)
Global State
Packaging (**numpy.distutils**)
NumPy Distutils - Users Guide
Status of **numpy.distutils** and migration advice
NumPy C-API
CPU/SIMD Optimizations
NumPy and SWIG

NumPy Reference

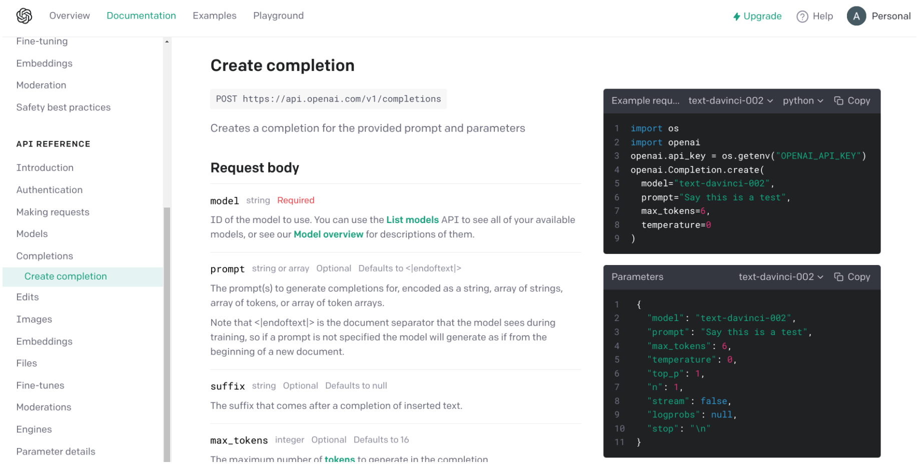
Release: 1.23
Date: June 22, 2022

This reference manual details functions, modules, and objects included in NumPy, describing what they are and what they do. For learning how to use NumPy, see the complete documentation.

- Array objects
  - The N-dimensional array (**ndarray**)
  - Scalars
  - Data type objects (**dtype**)
  - Indexing routines
  - Iterating Over Arrays
  - Standard array subclasses
  - Masked arrays
  - The array interface protocol
  - Datetimes and Timedeltas
- Array API Standard Compatibility
  - Table of Differences between **numpy.array_api** and **numpy**
- Constants

APIs in the Wild - REST API

OpenAI - API Reference



Create completion

```
POST https://api.openai.com/v1/completions
```

Creates a completion for the provided prompt and parameters

Request body

`model` string Required

ID of the model to use. You can use the **List models** API to see all of your available models, or see our **Model overview** for descriptions of them.

`prompt` string or array Optional Defaults to <|endoftext|>

The prompt(s) to generate completions for, encoded as a string, array of strings, array of tokens, or array of token arrays.

Note that <|endoftext|> is the document separator that the model sees during training, so if a prompt is not specified the model will generate as if from the beginning of a new document.

`suffix` string Optional Defaults to null

The suffix that comes after a completion of inserted text.

`max_tokens` integer Optional Defaults to 16

The maximum number of **tokens** to generate in the completion

Example requ... text-davinci-002 python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.Completion.create(
  model="text-davinci-002",
  prompt="Say this is a test",
  max_tokens=6,
  temperature=0
)
```

Parameters text-davinci-002 Copy

```json
{
  "model": "text-davinci-002",
  "prompt": "Say this is a test",
  "max_tokens": 6,
  "temperature": 0,
  "top_p": 1,
  "n": 1,
  "stream": false,
  "logprobs": null,
  "stop": "\n"
}
```

Ansys’s API - Local and Remote

APIs - Definition and Examples

Ansys

Ansys's API - A Holistic Overview

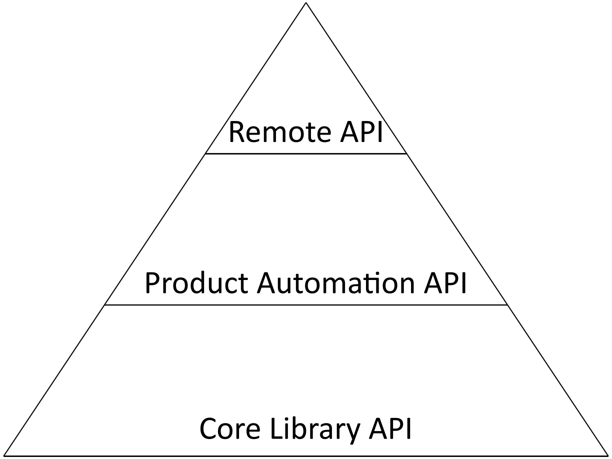
To address the need for product automation and low-level access, we need to expose three types of APIs:

- Core service/library APIs
- Product automation APIs
- Remote product APIs

These three levels of APIs ensure Ansys can enable:

- Inter-product communication without writing to disk
- Product customization and automation
- Compatibility with cloud infrastructure (e.g. Docker)

Ansys's Product APIs - Tech Stack



Network { REST } gRPC

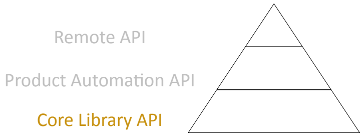
Interpreted python

Compiled

Core Library API - Overview

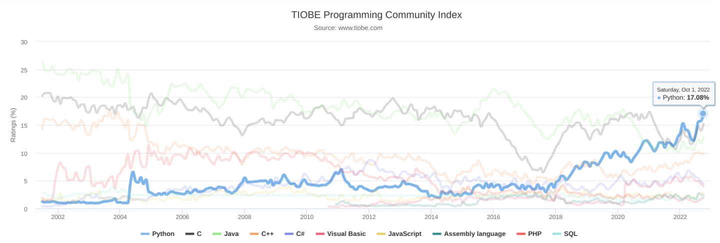
- Primarily directed for developers and (rarely) customers wishing to customize at a very low-level.
- Core Library API should be exposed in the same language as the library source.
- Minimizes code overhead and duplication.

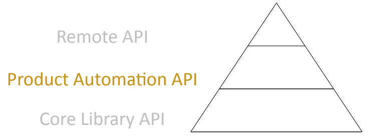
```
c   ---      i       1     numdof   Degrees of freedom per node
c                                   DOF reference numbers are:
c         UX   = 1, UY   = 2, UZ   = 3, ROTX= 4, ROTY= 5, ROTZ= 6, AX   = 7, AY   = 8
c         AZ   = 9, VX   =10, VY   =11, VZ   =12, GFV1=13, GFV2=14, GFV3=15, WARP=16
c         CONC=17, HDSP=18, PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
c         EMF  =25, CURR=26, SP01=27, SP02=28, SP03=29, SP04=30, SP05=31, SP06=32
c         TBOT=33, TE2  =34, TE3  =35, TE4  =36, TE5  =37, TE6  =38, TE7  =39, TE8  =40
c         TE9  =41, TE10=42, TE11=43, TE12=44, TE13=45, TE14=46, TE15=47, TE16=48
c         TE17=49, TE18=50, TE19=51, TE20=52, TE21=53, TE22=54, TE23=55, TE24=56
c         TE25=57, TE26=58, TE27=59, TE28=60, TE29=61, TE30=62, TE31=63, TTOP=64
c                                    (curdof(i),i=1,numdof)

c   NOD      i       1     nnod     Nodal equivalence table. This table equates
c                                   the number used for storage to the actual
c                                   node number
```



Product Customization and Automation - Overview

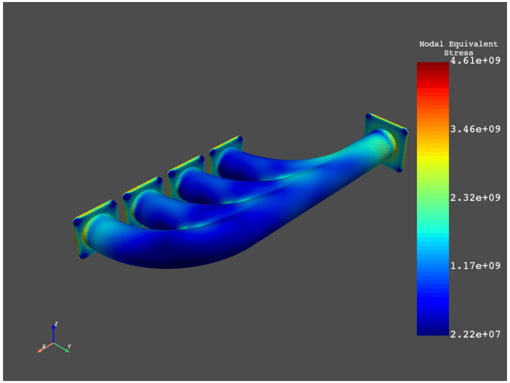
- Directed at method developers who understand the product at the “analyst” level as well as understand basic programming.
- Target language should be a high-level interpreted language like Python or Julia.
- Python is one of the top languages used by method developers due to its wide and deep ecosystem of libraries, frameworks, and tools.





Product Customization and Automation - Example

```python
# Define mesh controls and generate mesh
mapdl.esize(0.0075)
mapdl.vmesh("all")

# Save mesh as VTK object
grid = mapdl.mesh.grid

# Map the imported data to MAPDL grid
inter_grid = grid.interpolate(
    wrapped,
    sharpness=5,
    radius=0.0001,
    strategy="closest_point",
)

# Save node numbers
node_num = inter_grid.point_data["ansys_node_num"]
```

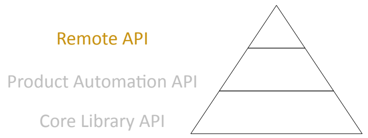


Remote API - Overview

- Consider two types of Remote APIs:
  - Inter-product communication: Use an architecture designed to allow the manipulation and exchange of information between multiple technologies and componentized software components. The gRPC framework works well for this scenario.
  - Web APIs: Use an architecture designed for the interaction of physically separate components. REST is an ideal architectural style given its stateless, cacheable nature.
- The choice of the underlying architecture and technology must be driven by the problem and there is no "one size fits all" solution.

Example REST Call

```
curl -X POST "http://127.0.0.1:5000/Vectors" \
  -H "Content-Type: application/json" \
  -d '{"value":[5, 23, 3, 4]}'
```

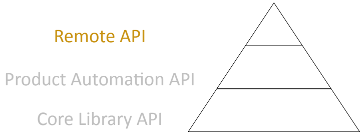


Remote API - Lessons Learned

- Web-facing APIs must reply on an abstraction layer that exposes only what is necessary to accomplish the task.
  - An extended audience for your APIs.
  - APIs can be reused to support new digital products.
  - Normalized, consistent and well-governed API code and documentation.
  - Flexibility on alternative/future coding standards and formats.
- This will require the same sort of refactoring and abstracting as when exposing Ansys's products as GUIs. However, this time we need to consider both human and computers as "first class" users.

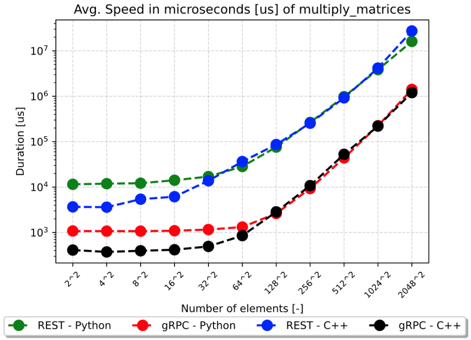
Abstracted API

```
geometry_api = GeometriesApi(api_client)
mesh_operation_api = MeshOperationsApi(api_client)
simulation_api = SimulationsApi(api_client)
```



Remote API - Example

- We have an example for you at API Eigen Example
- Includes:
  - C++ REST Server and Client
  - C++ gRPC Server and Client
  - Python REST Server and Client
  - Python gRPC Server and Client
- Example designed to expose you to the basics of library-orientated remote APIs



Example

```
>>> import ansys.eigen.python.rest.server as rest_server
>>> app = rest_server.create_app()
>>> app.run(”127.0.0.1”, 5000)
```

Ansys