Custom Copilot - Azure OpenAI und Semantic Kernel für Softwareentwickler

Thomas Tomow

CTOO / Co-Founder Xebia MS Germany

Microsoft MVP since 2017 (IoT, AI, Cloud Native)

Thomas Tomow

CTO / COO @ Xebia MS Germany

- Microsoft MVP since 2017
- Developer by Heart over 20 years
- Languages:
  - C(#,++), Python, Delphi, Pascal, VB, Assembler, Logo, Prolog, JavaScrip, TypeScript, ...
- Solution Architect/ Cloud Architect & AI Engineer

Balancing by Hicking & Photography, like alos Karate

Host for Azure Meetup Konstanz & Co-Host Azure Meetup Stuttgart

Azure Meetup KONSTANZ

Azure Meetup STUTTGART

GENERATIVE AI

What is a Model?

A model refers to a specific instance or version of an LLM AI

Available GPT Models

- Ada
  - basic natural language understanding
  - classification, sentiment analysis, summarization, and simple conversation.
- Babbage
  - more complex natural language tasks, such as reasoning, logic, arithmetic, and word analogy.
- Curie
  - advanced natural language tasks, such as text-to-speech, speech-to-text, translation, paraphrasing, and question answering.
- Davinci
  - Handles almost any natural language task,
  - as well as some multimodal tasks, such as image captioning, style transfer, and visual reasoning.
  - It can also generate coherent and creative texts on any topic, with a high level of fluency, consistency, and diversity.

AI Services

Open AI
https://platform.openai.com/docs/models

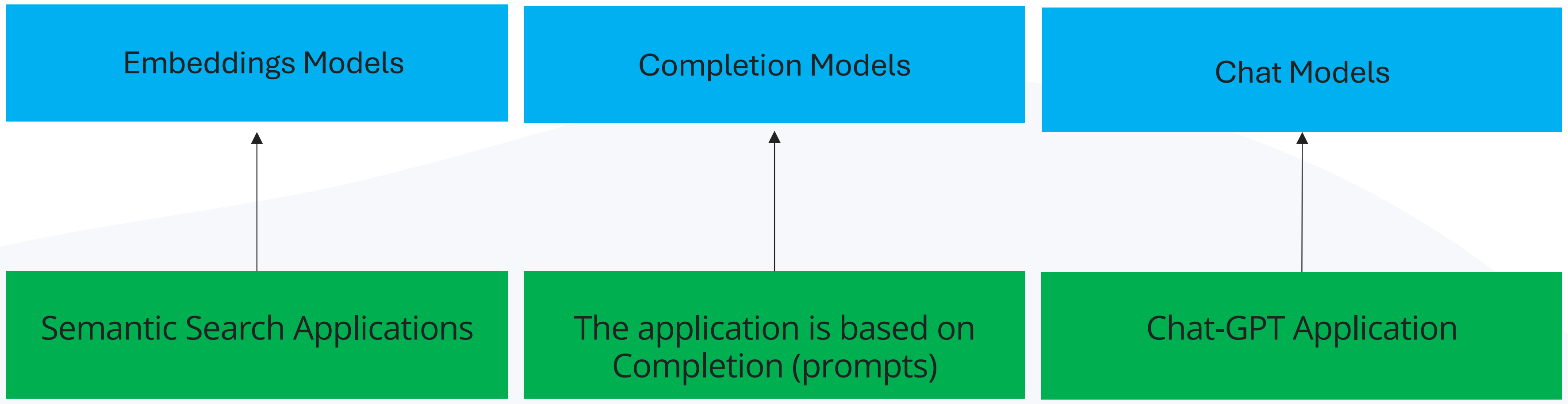
Azure Open AI
https://oai.azure.com/portal/****/models

Request Access to Azure OpenAI Service (microsoft.com)

DEMO

Azure Open AI Playground

Generative Model Types


Embeddings Models
Completion Models
Chat Models
Semantic Search Applications
The application is based on Completion (prompts)
Chat-GPT Application

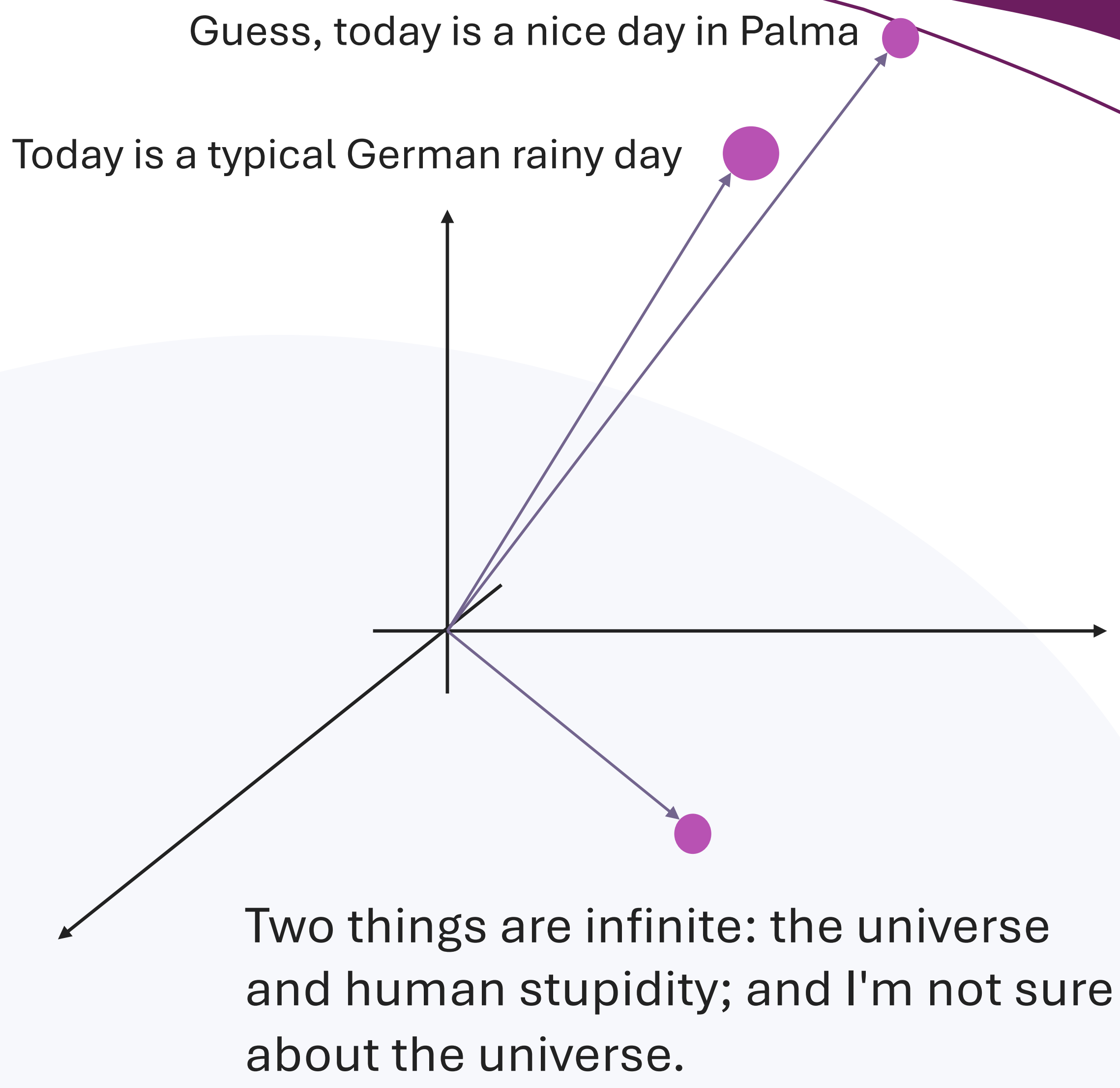
Completion Models

```
Summarize following scientific abstract, using a user-friendly language.

[Abstract]
Sparse representation has attracted much attention from researchers in fields of
signal processing, image processing, computer vision, and pattern recognition.
Sparse representation also has a

good reputation in both theoretical research . . .
```

Chat Models

```
Bot: How can I help you?
User: What's the weather like today?
Bot: Where are you located?
User: I'm in Frankfurt.
Bot: It’s 22 degrees and sunny in Frankfurt today.
User: Thanks! Can you explain scientific abstracts?
Bot: Yes. Please provide abstract.
```

Embedding Models


Guess, today is a nice day in Palma
Today is a typical German rainy day
Two things are infinite: the universe and human stupidity; and I'm not sure about the universe.

Similarity Between Multidimensional Vectors

- Dot Product
- The Norm
- Cosine Similarity

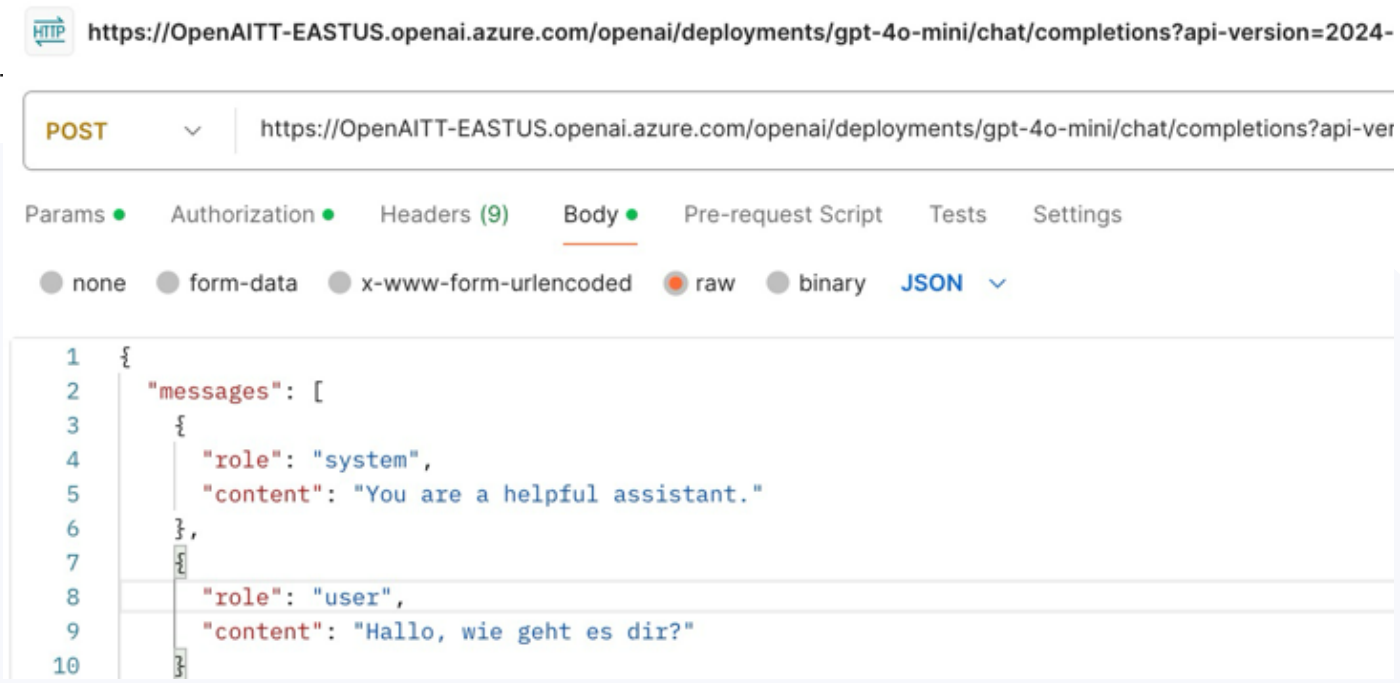
$$A \cdot B = a_1 \cdot b_1 + a_2 \cdot b_2 + \ldots + a_n \cdot b_n$$

$$\|\mathbf{A}\| = \sqrt{a_1^2 + a_2^2 + \ldots + a_n^2}$$

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \|\mathbf{B}\| \cos(\theta)$$

DEMOs

- Consuming AI Service as a REST Endpoint
  - Using Completions
  - Generating Embeddings
- Using AzureOpenAIClien
  - Chat models and more



https://OpenAITT-EASTUS.openai.azure.com/openai/deployments/gpt-4o-mini/chat/completions?api-version=2024-

POST https://OpenAITT-EASTUS.openai.azure.com/openai/deployments/gpt-4o-mini/chat/completions?api-ver

```json
{
  "messages": [
    {
      "role": "system",
      "content": "You are a helpful assistant."
    },
    {
      "role": "user",
      "content": "Hallo, wie geht es dir?"
    }
```

When to use Embeddings?

Semantic Search

Classification

Clustering

Recommendations

. . .

What is a token?

- 1 token ~= 4 chars in English
- 1 token ~= ¾ words
- 100 tokens ~= 75 words
- Byte Pare Encoding (Gage,1994): [Wikipedia]

Tokenizer: https://platform.openai.com/tokenizer

What are tokens and how to count them?

Token Pricing: Pricing (openai.com)

SEMANTIC KERNEL

Software V2

Eine neue Generation von Software

Today

Syntax Driven Languages

“Tomorrow”

AI & Large Language Model Integration

“After tomorrow”

Quantum Computing Software

Semantic Kernel is an open-source SDK that lets you easily combine Generative AI Models syntactic programming languages like C#, Python an JAVA.

Currently supported languages: C#, Python

Currently supported AI Services: OpenAI, Azure OpenAI, and Hugging Face

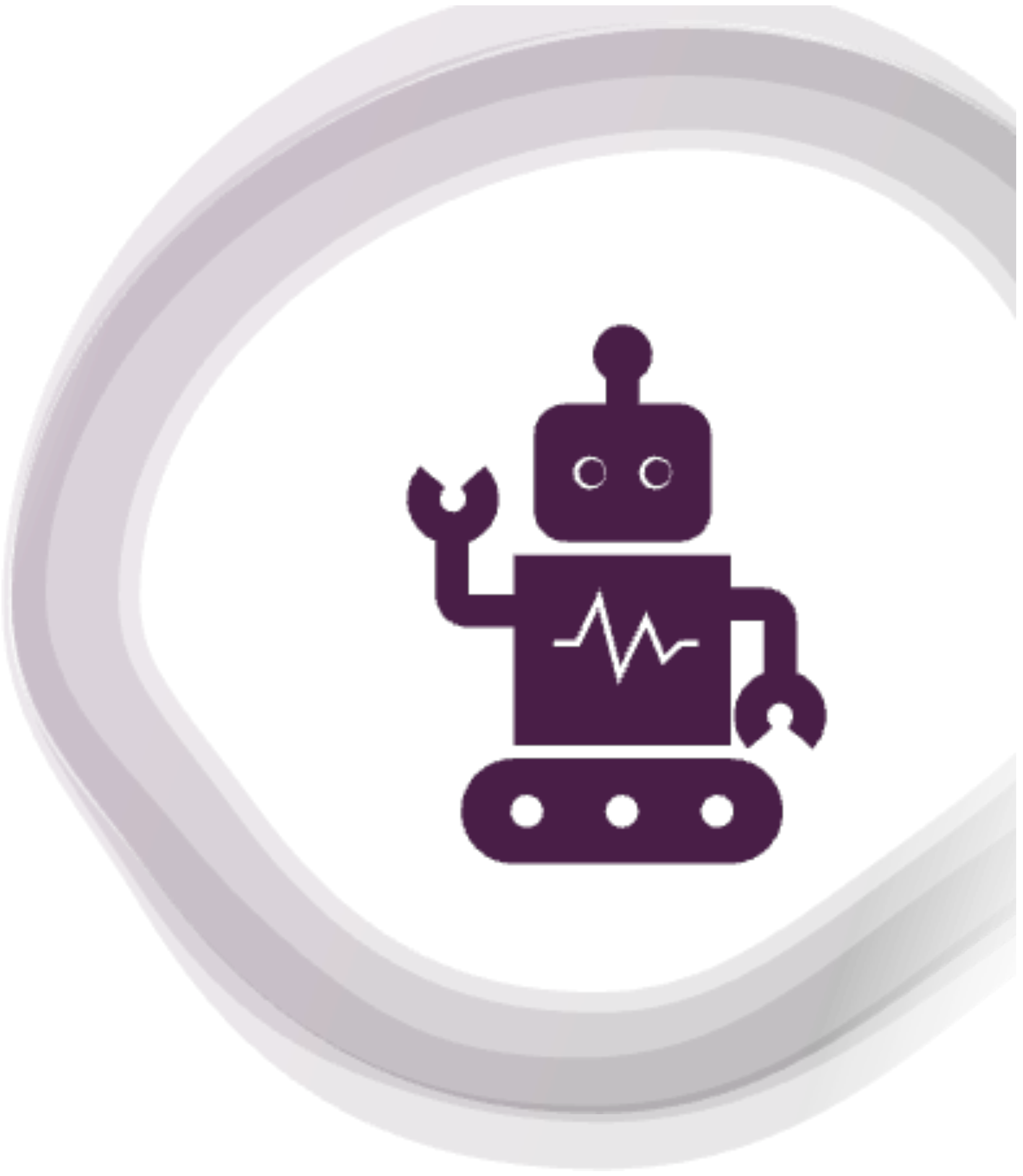
Introduction | 🦜🔗 Langchain

What is an Agent?

- An agent is an AI that can:
  - Answer questions
  - Automate processes for users
  - . . .

What is a Copilot?

- A Copilot is an AI application that consists of one or more agents,
- which orchestrate tasks.



Why to use SK?

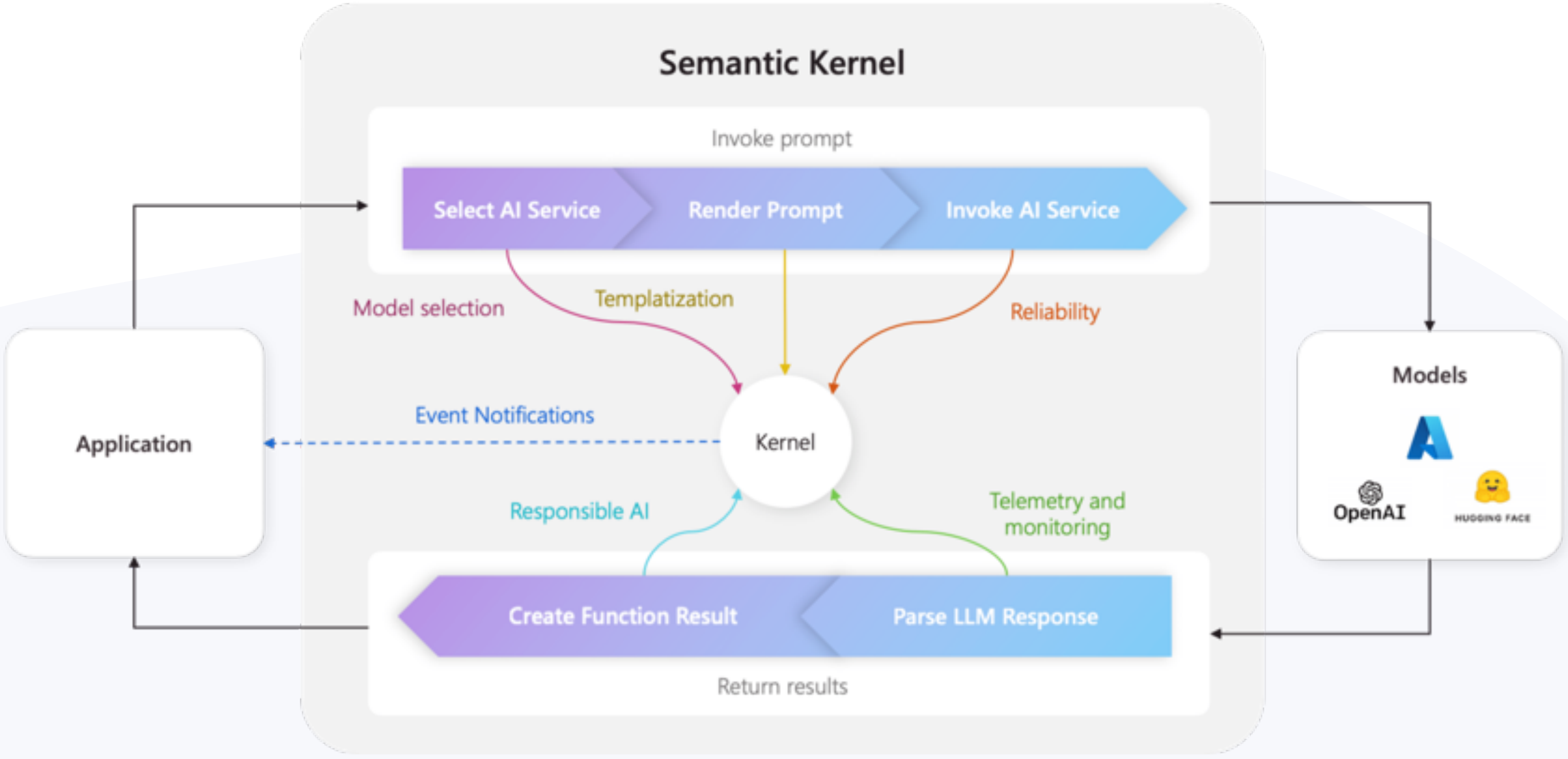
SK is a framework to build agents

You can easily use LLM models for chat, create images and video so on.

Making automated Agents that automate business processes is hard?

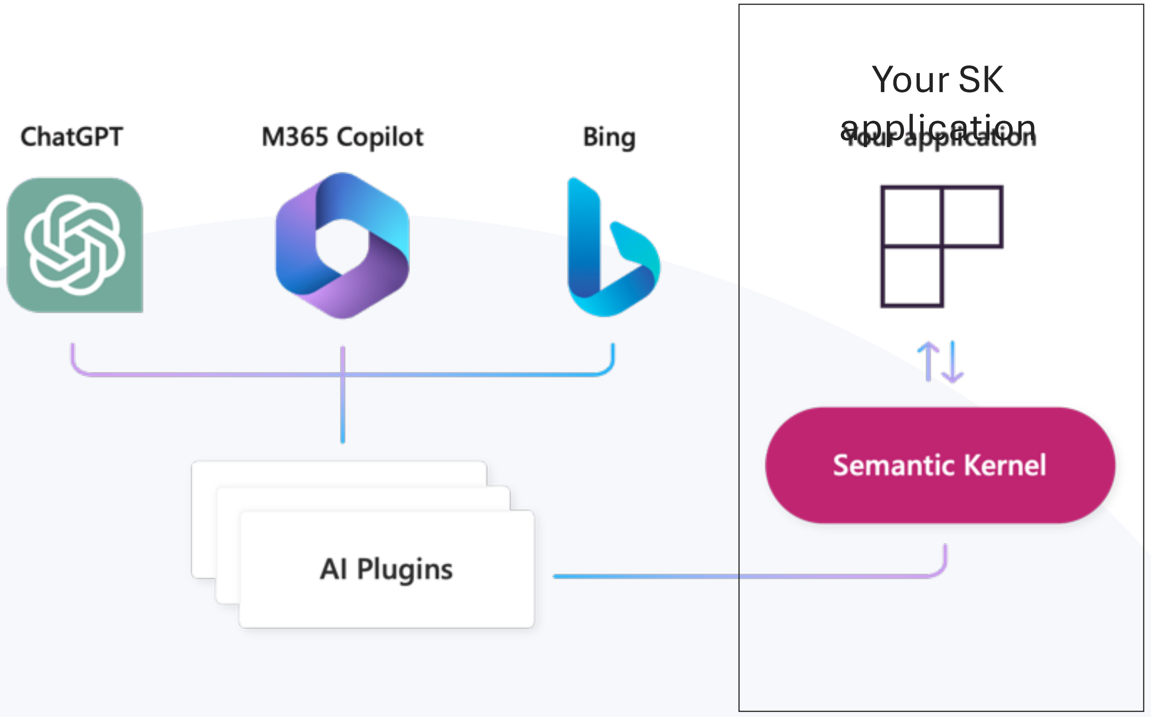
You need a framework for such complex engineering tasks.

Architecture of the “Smart” application



AI Plugins

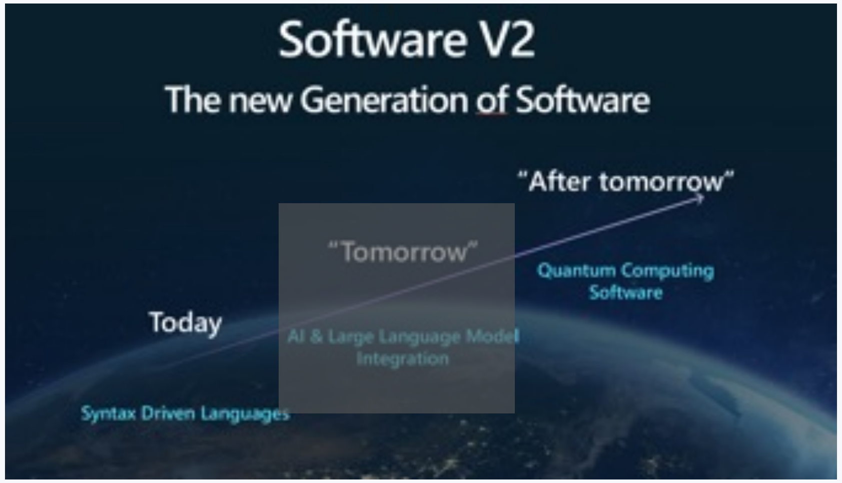
a plugin is a group of functions
that can be exposed to AI applications



OpenAI plugin specification

The Knowledge is inside Plugins

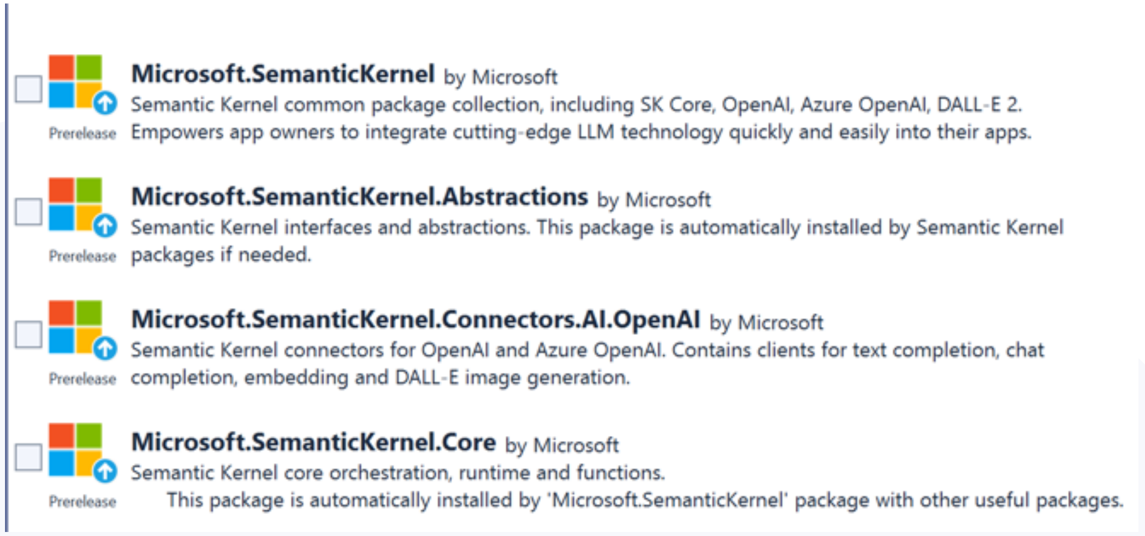
- Native Functions (C#, Python, JAVA, ??)
- Prompts (Semantic Functions)


Software V2
The new Generation of Software
"After tomorrow"
"Tomorrow"
Quantum Computing Software
Today
AI & Large Language Model Integration
Syntax Driven Languages

Deep Dive into Semantic Kernel

1. Overview of the kernel
2. Creating native functions
3. Creating semantic functions
4. Understanding AI plugins
5. Chaining functions together
6. Planer
7. Create and run a ChatGPT plugin

SEMANTIC KERNEL OVERVIEW



Microsoft.SemanticKernel by Microsoft
Semantic Kernel common package collection, including SK Core, OpenAI, Azure OpenAI, DALL-E 2.
Prerelease Empowers app owners to integrate cutting-edge LLM technology quickly and easily into their apps.

Microsoft.SemanticKernel.Abstractions by Microsoft
Semantic Kernel interfaces and abstractions. This package is automatically installed by Semantic Kernel
Prerelease packages if needed.

Microsoft.SemanticKernel.Connectors.AI.OpenAI by Microsoft
Semantic Kernel connectors for OpenAI and Azure OpenAI. Contains clients for text completion, chat
Prerelease completion, embedding and DALL-E image generation.

Microsoft.SemanticKernel.Core by Microsoft
Semantic Kernel core orchestration, runtime and functions.
Prerelease This package is automatically installed by 'Microsoft.SemanticKernel' package with other useful packages.

Semantic Kernel Initialization with OpenAI

```
private static IKernel GetAzureKernel()
 {
    var kernel =  kernel = Kernel.CreateBuilder()
    .AddOpenAIChatCompletion(
  Environment.GetEnvironmentVariable("OPENAI_CHATCOMPLETION_DEPLOYMENT")!,
  Environment.GetEnvironmentVariable("OPENAI_API_KEY")!,
  Environment.GetEnvironmentVariable("OPENAI_ORGID")!)
.Build();}
```

Semantic Kernel Initialization with Azure OpenAI

```
private static IKernel GetKernel()
{
   var kernel =  kernel = Kernel.CreateBuilder()
.AddAzureOpenAIChatCompletion(
   Environment.GetEnvironmentVariable("AZURE_OPENAI_CHATCOMPLETION_DEPLOYMENT")!,
   Environment.GetEnvironmentVariable("AZURE_OPENAI_ENDPOINT")!,
   Environment.GetEnvironmentVariable("AZURE_OPENAI_API_KEY")! )
.Build();  }
```

Two types of functions

Native Functions

Semantic Functions

NATIVE FUNCTIONS

A native function is a function

defined by the code

```
[SKFunction, Description("Gets the UTC current time.")]
public string UtcNow()
{
    return DateTime.UtcNow.ToString();
}
```

SEMANTIC FUNCTIONS

Inline Functions

```
string prompt = @"Bot: How can I help you?
User: {{$input}}
---------------------------------------------
The intent of the user in 5 words or less: ";
```

A semantic function is a function defined by the LLM prompt

```
Plugins
|
|—— Plugin1
    |
    |—— Function11
    |—— Function12

└—— Plugin2
    |
    |—— Function21
    |—— Function22
```

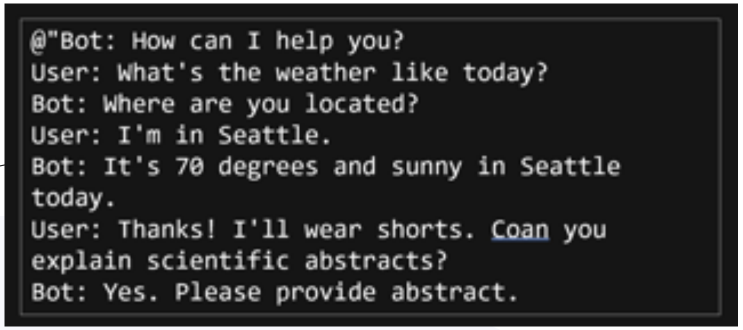
Add Ins

In-file Functions

Preserving context in Conversation History

```
Bot: How can I help you?
User: What's the weather like today?
Bot: Where are you located?
User: I'm in Frankfurt.
Bot: It’s 22 degrees and sunny in Frankfurt today.
User: Thanks! Can you explain scientific abstracts?
Bot: Yes. Please provide abstract.
```

Using Conversation History as variable



```
@"Bot: How can I help you?
User: What's the weather like today?
Bot: Where are you located?
User: I'm in Seattle.
Bot: It's 70 degrees and sunny in Seattle
today.
User: Thanks! I'll wear shorts. Coan you
explain scientific abstracts?
Bot: Yes. Please provide abstract.
```

```
[History]
{{$history}}

---------------------------------------------

Summarize scientific abstract, using a user-friendly language.

[Input]
{{$input}}
```

Using variables (Templatizing), Prompt Template Syntax)

```
[History]
{{$history}}
User: I'm {{$age}} years old kid.
Bot:Ok, explaining to {{$age}} old kid.

---------------------------------------------

Summarize abstract, using a user friendly language. Use language of the
{{$age}} years old kid. Don't use subjects like "we" "our" "us" "your".
If {{$age}} is not on of following options {{$options}} then return an
error message similar to 'Sorry, please use some meaningfull age'.


[Input]
{{$input}}
```

https://learn.microsoft.com/en-us/semantic-kernel/prompt-engineering/prompt-template-syntax

Nested Semantic Functions

Semantic function invokes a semantic function

```
{{SamplePlugin.Translator $input}}

------------------------------------------------

Summarize and compress abstract to 50 words, using a user friendly
language.

[Input]
{{$input}}
```

Nested Native Functions

Semantic function invokes a native function

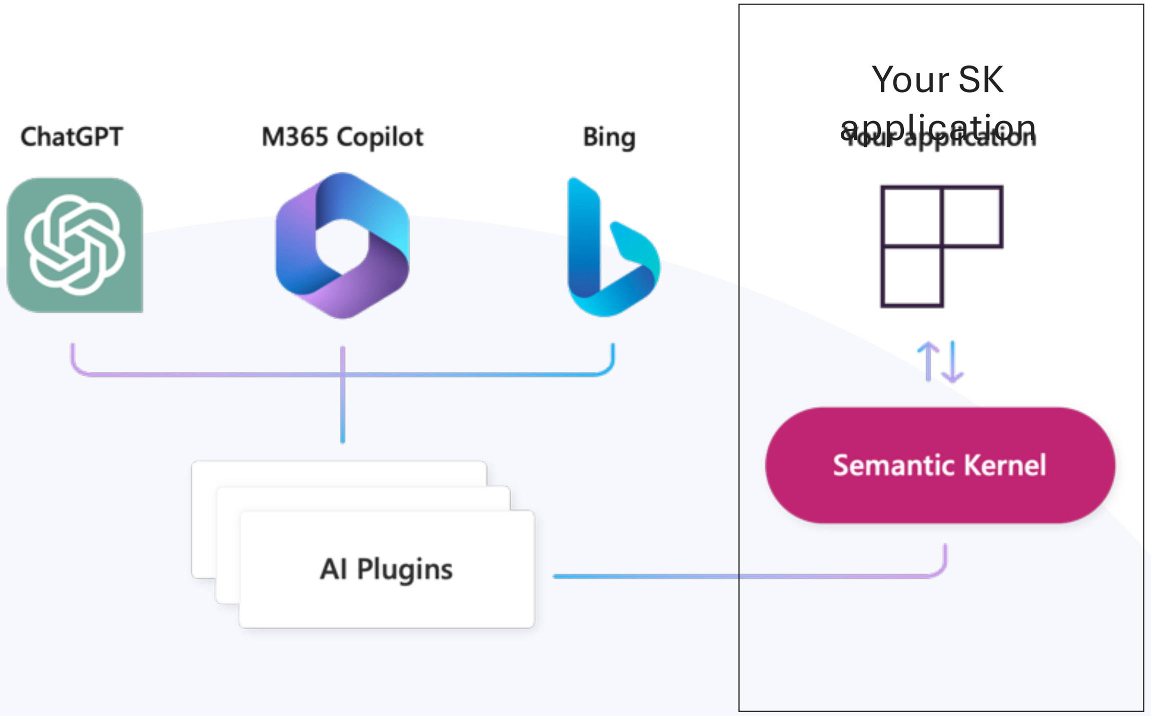
```
{{SamplePlugin.Translator $input}}

--------------------------------------

Summarize the scientific abstract using a user friendly language.
Counter = {{StringPlugin.CharCount $input}}
Provide the result in following format:

[Output]
Counter: number,
Text: translated text

[Input]
{{$input}}

--------------------------------------

[Output]
```

Chaining Functions

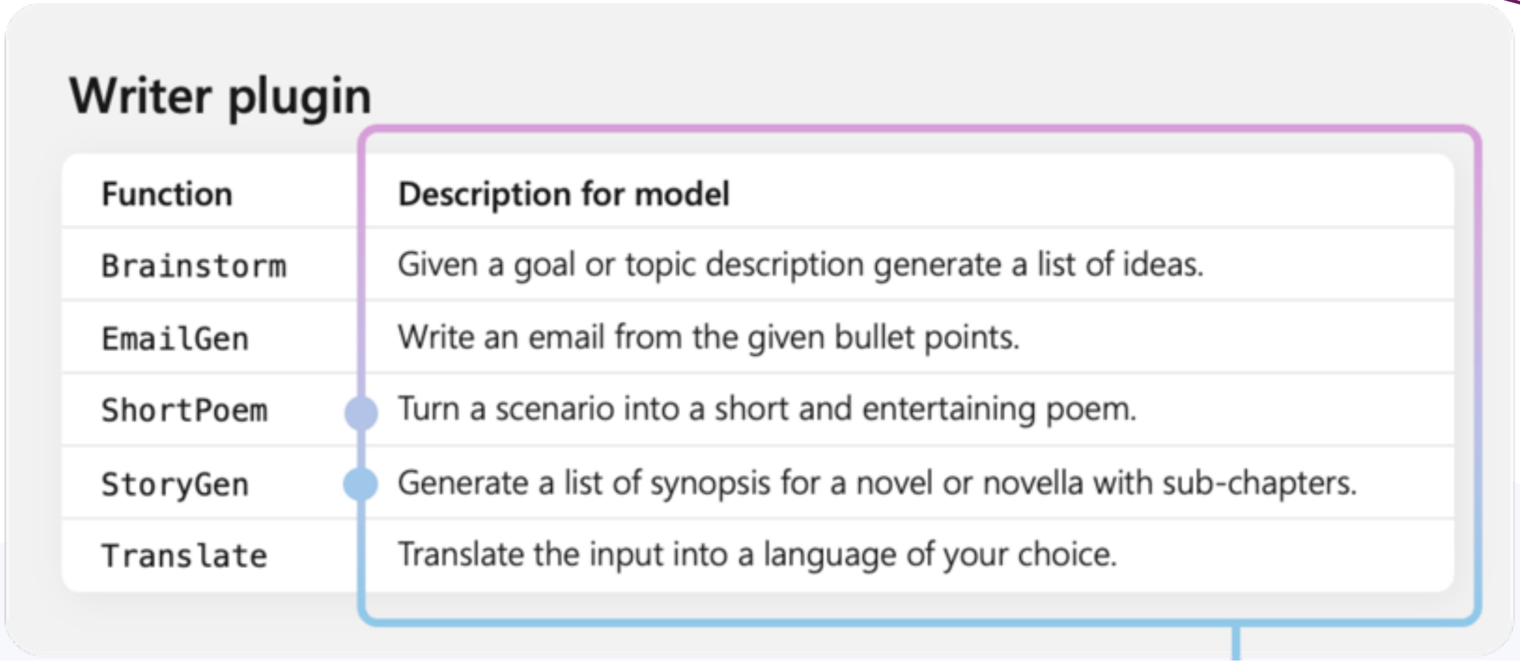
```
string myJokePrompt = """
Tell a short joke about {{$input}}.
""";

string myPoemPrompt = """
Take this "{{$input}}" and convert it to a nursery rhyme.
""";

string myMenuPrompt = """
Make this poem "{{$input}}" influence the three items in a coffee shop menu.
The menu reads in enumerated form:
""";

var myJokeFunction = kernel.CreateSemanticFunction(myJokePrompt, requestSettings: sett);
var myPoemFunction = kernel.CreateSemanticFunction(myPoemPrompt, requestSettings: sett);
var myMenuFunction = kernel.CreateSemanticFunction(myMenuPrompt, requestSettings: sett);

// Get the GetIntent function from the OrchestratorPlugin and run it
var result = await kernel.RunAsync(new ContextVariables("Damir Dobric Microsoft Regional Director"),
            myJokeFunction, myPoemFunction, myMenuFunction);
```

AI Plugins

a plugin is a group of functions
that can be exposed to AI applications



OpenAI plugin specification



Writer plugin

Function	Description for model
`Brainstorm`	Given a goal or topic description generate a list of ideas.
`EmailGen`	Write an email from the given bullet points.
`ShortPoem`	Turn a scenario into a short and entertaining poem.
`StoryGen`	Generate a list of synopsis for a novel or novella with sub-chapters.
`Translate`	Translate the input into a language of your choice.

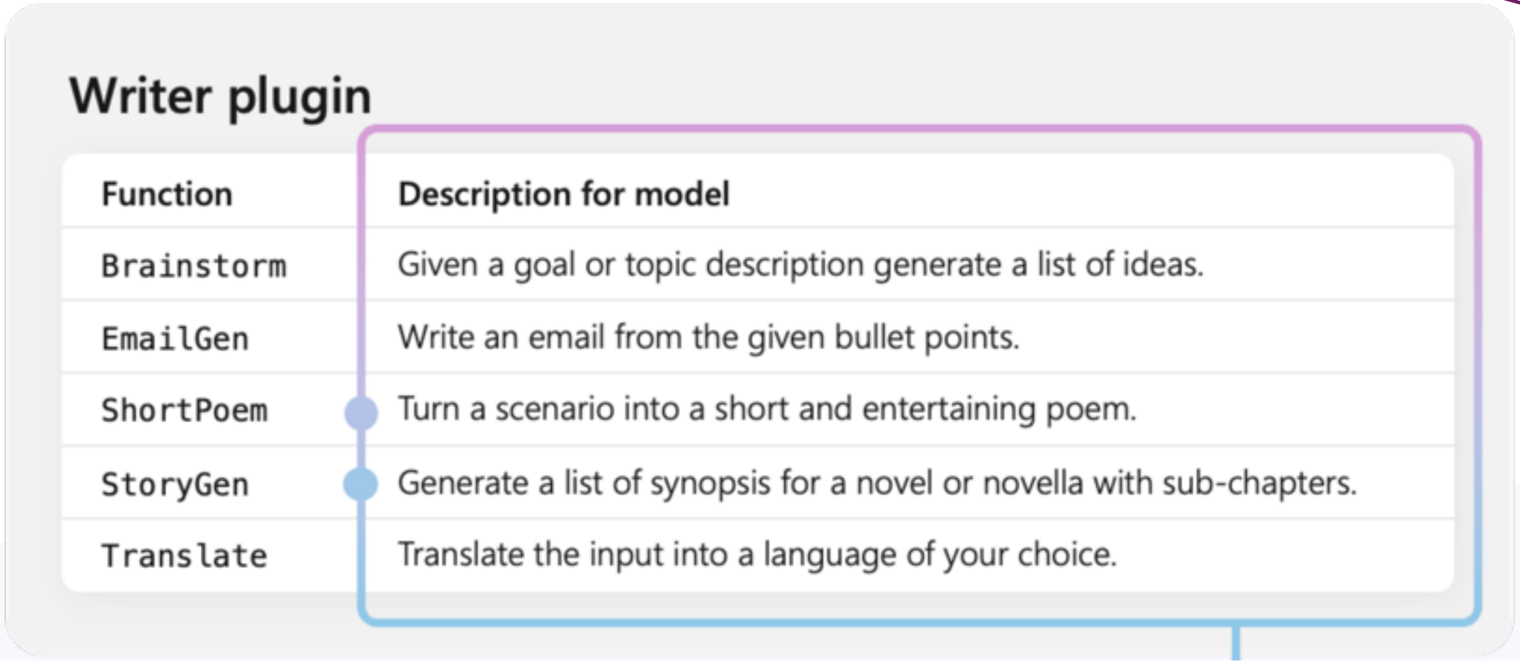
Planers

1. Planner is a function
2. I takes a prompt and returns back a plan on how to accomplish the task described in the prompt.
3. Mix-and-match the plugins.
4. Recombine them into a series of steps that complete a goal.

Converts the prompt into orchestration of functions implemented in Plugins

```
If my investment of 2130.23 dollars increased by 23%,
 how much would I have after I spent $5 on a latte?
```

Planers



Writer plugin

Function	Description for model
`Brainstorm`	Given a goal or topic description generate a list of ideas.
`EmailGen`	Write an email from the given bullet points.
`ShortPoem`	Turn a scenario into a short and entertaining poem.
`StoryGen`	Generate a list of synopsis for a novel or novella with sub-chapters.
`Translate`	Translate the input into a language of your choice.

The plan is a mapping

m: prompt -> {f1,f2,..,fN}

The plan is mapping from prompt to the set of functions

All starts with the Goal

"Summarize an input, translate to klingon, and e-mail to Thomas”

Plan execution

```
kernel.ImportSemanticSkillFromDirectory(folder,
    "SummarizeSkill",
    "WriterSkill");

var plan = await planner.CreatePlanAsync("Summarize an input, translate to klingon, and e-mail to
Thomas");

var input = "Once upon a time, in a faraway kingdom, there lived a kind and just king named Arjun. " +
      "He ruled over his kingdom with happily ever after, with the people of the kingdom remembering
Mira as
       the brave young woman who saved them from the dragon.";
```

- `await ExecutePlanAsync(kernel, plan, input, 5);`

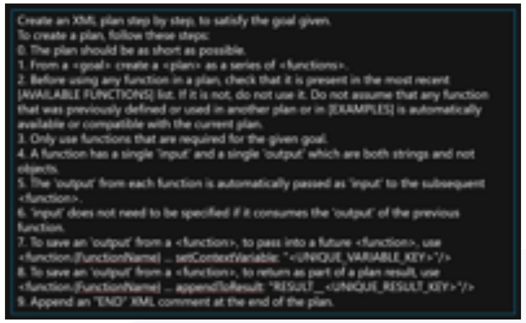
SK creates the plan for a given goal

```
<goal>Summarize the input, then translate to Klingon and email it to Thomas</goal>
<plan>
  <function.WriterSkill.Summarize/>
  <function.LanguageHelpers.TranslateTo translate_to_language="Japanese"
             setContextVariable="TRANSLATED_TEXT" />
  <function.EmailConnector.LookupContactEmail input="Martin"
             setContextVariable="CONTACT_RESULT" />
  <function.EmailConnector.EmailTo input="$TRANSLATED_TEXT"
             recipient="$CONTACT_RESULT"/>
</plan>
```

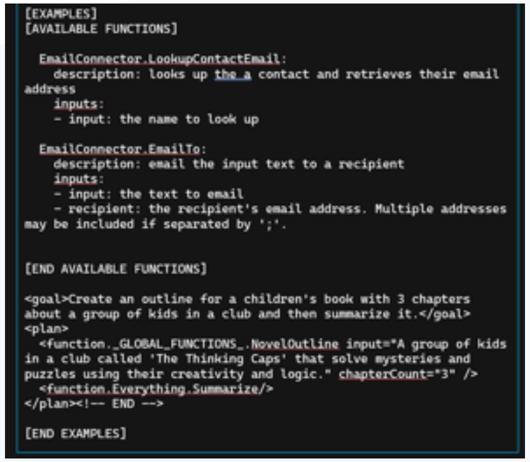
The plan is created from Semantic Function

Create an XML plan step by step, to satisfy the goal given.
To create a plan, follow these steps:
0. The plan should be as short as possible.
1. From a <goal> create a <plan> as a series of <functions>.
2. Before using any function in a plan, check that it is present in the most recent [AVAILABLE FUNCTIONS] list. If it is not, do not use it. Do not assume that any function that was previously defined or used in another plan or in [EXAMPLES] is automatically available or compatible with the current plan.
3. Only use functions that are required for the given goal.
4. A function has a single 'input' and a single 'output' which are both strings and not objects.
5. The 'output' from each function is automatically passed as 'input' to the subsequent <function>.
6. 'input' does not need to be specified if it consumes the 'output' of the previous function.
7. To save an 'output' from a <function>, to pass into a future <function>, use <function.{FunctionName} ... setContextVariable: "<UNIQUE_VARIABLE_KEY>"/>
8. To save an 'output' from a <function>, to return as part of a plan result, use <function.{FunctionName} ... appendToResult: "RESULT__<UNIQUE_RESULT_KEY>"/>
9. Append an "END" XML comment at the end of the plan.

Mapping the Goal to Functions

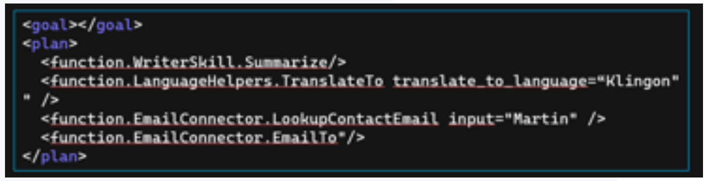


```
Create an XML plan step by step, to satisfy the goal given.
To create a plan, follow these steps:
0. The plan should be as short as possible.
1. From a <goal> create a <plan> as a series of <functions>.
2. Before using any function in a plan, check that it is present in the most recent
[AVAILABLE FUNCTIONS] list. If it is not, do not use it. Do not assume that any function
that was previously defined or used in another plan or in [EXAMPLES] is automatically
available or compatible with the current plan.
3. Only use functions that are required for the given goal.
4. A function has a single 'input' and a single 'output' which are both strings and not
objects.
5. The 'output' from each function is automatically passed as 'input' to the subsequent
<function>.
6. 'input' does not need to be specified if it consumes the 'output' of the previous
function.
7. To save an 'output' from a <function>, to pass into a future <function>, use
<function.{FunctionName}> ... setContextVariable: "<UNIQUE_VARIABLE_KEY>"/>
8. To save an 'output' from a <function>, to return as part of a plan result, use
<function.{FunctionName}> ... appendToResult: "RESULT__<UNIQUE_RESULT_KEY>"/>
9. Append an "END" XML comment at the end of the plan.
```



```
[EXAMPLES]
[AVAILABLE FUNCTIONS]

  EmailConnector.LookupContactEmail:
    description: looks up the a contact and retrieves their email
address
    inputs:
    - input: the name to look up

  EmailConnector.EmailTo:
    description: email the input text to a recipient
    inputs:
    - input: the text to email
    - recipient: the recipient's email address. Multiple addresses
may be included if separated by ';'.


[END AVAILABLE FUNCTIONS]

<goal>Create an outline for a children's book with 3 chapters
about a group of kids in a club and then summarize it.</goal>
<plan>
  <function._GLOBAL_FUNCTIONS_.NovelOutline input="A group of kids
in a club called 'The Thinking Caps' that solve mysteries and
puzzles using their creativity and logic." chapterCount="3" />
  <function.Everything.Summarize/>
</plan><!-- END -->

[END EXAMPLES]
```

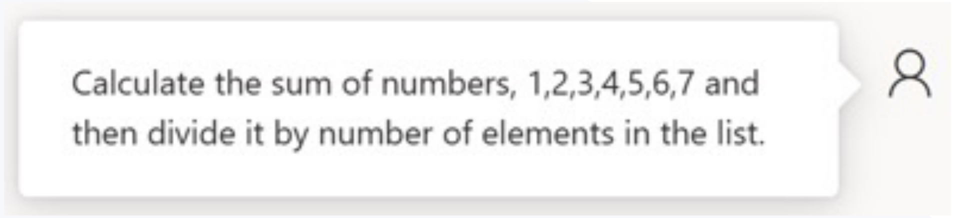
Davinci completion model
GPT4 Model

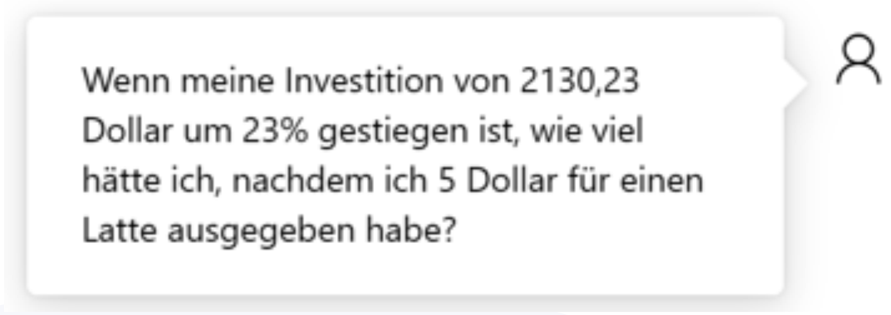


```
<goal></goal>
<plan>
  <function.WriterSkill.Summarize/>
  <function.LanguageHelpers.TranslateTo translate_to_language="Klingon"
" />
  <function.EmailConnector.LookupContactEmail input="Martin" />
  <function.EmailConnector.EmailTo"/>
</plan>
```

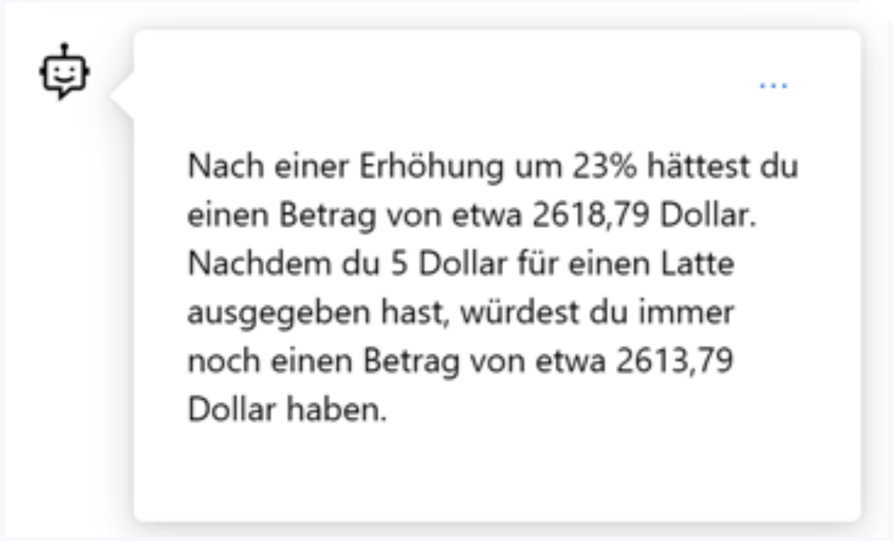
How does the planer work?

```
<goal>Summarize the input, then translate to Klingon and email it to Damir</goal>
<plan>
  <function.WriterSkill.Summarize/>
  <function.LanguageHelpers.TranslateTo translate_to_language=“Klingon"
             setContextVariable="TRANSLATED_TEXT" />
  <function.EmailConnector.LookupContactEmail input="Martin"
             setContextVariable="CONTACT_RESULT" />
  <function.EmailConnector.EmailTo input="$TRANSLATED_TEXT"
             recipient="$CONTACT_RESULT"/>
</plan>
```


Calculate the sum of numbers, 1,2,3,4,5,6,7 and then divide it by number of elements in the list.

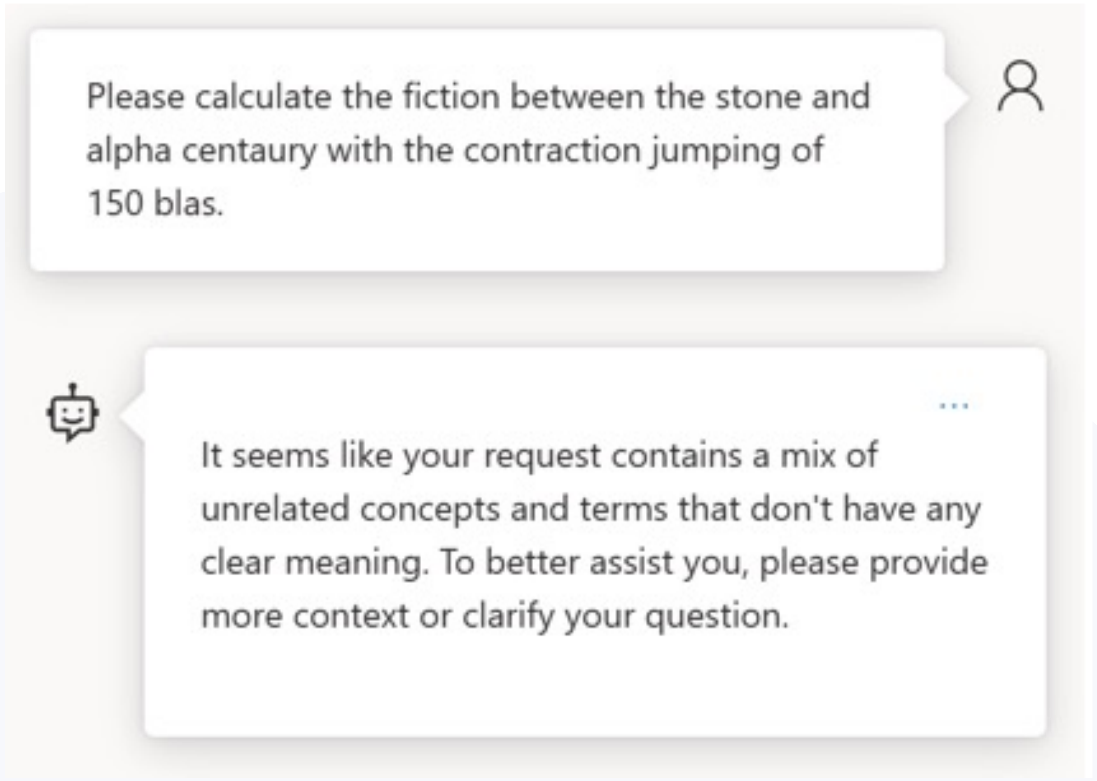


Wenn meine Investition von 2130,23 Dollar um 23% gestiegen ist, wie viel hätte ich, nachdem ich 5 Dollar für einen Latte ausgegeben habe?



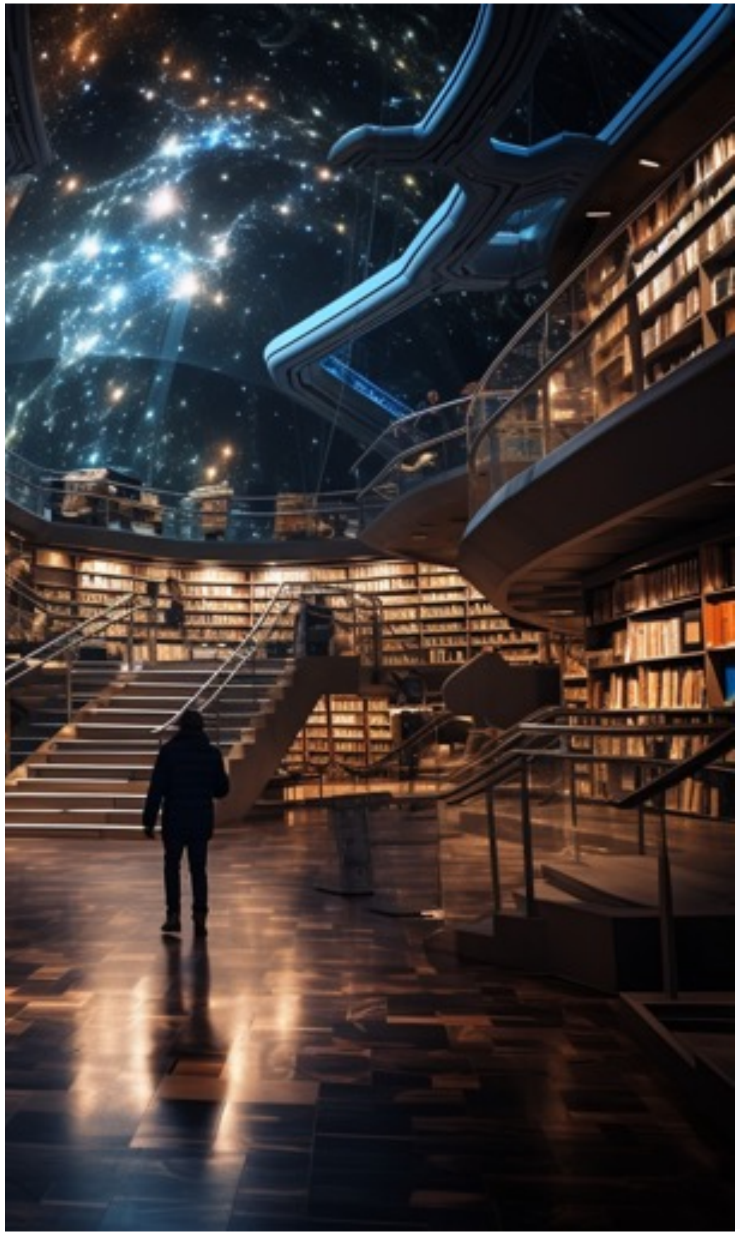
Nach einer Erhöhung um 23% hättest du einen Betrag von etwa 2618,79 Dollar. Nachdem du 5 Dollar für einen Latte ausgegeben hast, würdest du immer noch einen Betrag von etwa 2613,79 Dollar haben.

How about something not solvable?



Please calculate the fiction between the stone and alpha centaury with the contraction jumping of 150 blas.

It seems like your request contains a mix of unrelated concepts and terms that don't have any clear meaning. To better assist you, please provide more context or clarify your question.



References

- Azure/aistudio-copilot-sample at oct-refresh (github.com)
- Microsoft Copilot Studio
- Azure AI Studio
- Semantic Kernel
- AI-Generated 3D Worlds
- Imagine AI Art Generator
- Midjourney
- Azure AI SDK - Azure AI Studio
- BOOK OF NEWS - Microsoft Ignite