



**Sam  
Altman**

山姆·阿尔特曼

# 万物智能成本无限降低 人类生产力与创造力解放

The Infinitely Lower Cost Of Intelligence In All Things  
Human Productivity And Creativity Emancipated

**实现管理通用人工智能AGI**  
(OpenAI/ChatGPT)

**做可控核聚变电价降到一美分**  
3.75亿美元投Helion

**成立由企业家组成的超级组织**  
改善资本主义经济

**建立宪章城市未来的基础设施**  
测试未来的管理方式

**给普通人提供全民基础收入币**  
虹膜识别，发放World Coin





硅创社

Silicon Generative  
Community

▶▶ AIGC-GPT-ChatGPT 

后GPT时代开发范式思考



潘工

# 应用5范式：2层应用3层模型

## ⑤ 应用程序

这一层是**人类和机器协作的界面**。这些是工作流程工具，使人工智能模型能够以使商业客户或消费者娱乐的方式获得。

## ④ 操作系统& API

这一层**简化了应用程序和工作流程的互操作性**。它有助于跟踪身份、支付、法律问题、服务条款、存储等。它帮助应用程序开发人员更快地尝试更多的功能，并使它们更快地运行。

## ③ 超本地化 AI模型

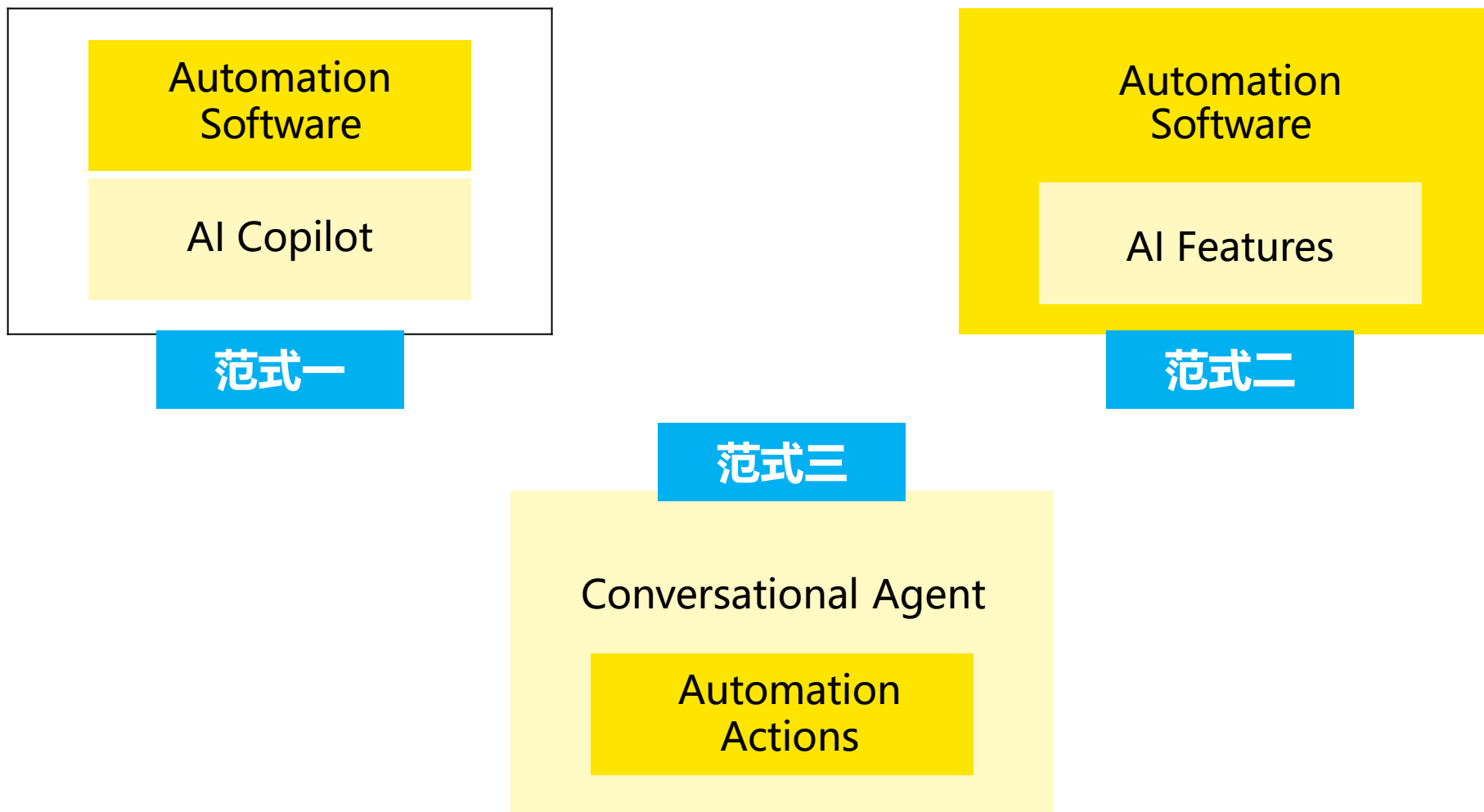
可以用《自然》的风格写一篇科学文章、可以创建适合特定人审美的室内设计模型、撰写的特殊风格来写代码、可以完全按照某家公司的风格进行灯光和阴影处理。**这个模型是在本地、典型的专有数据上训练的。**

## ② 特定 AI模型

如编写推文、广告文案、歌词，或生成电子商务照片、3D 室内设计图像等。这些模型是**在范围更窄、更专业的数据上训练出来的，这应该能让它们在特定的场景里胜过一般模型。**

## ① 通用 AI模型

类似GPT3的文本，DALL-E2的图像，Whisper 的语音，或Stable Difusion·这些模型处理广泛类别的输出：文本、图象、视频、语音、游戏。**它们将是开源的、易于使用的，并且擅长上述所有的事情。**



**GPT-1**

GPT-1是第一次使用预训练方法来实现高效语言理解的训练

**GPT-2**

GPT-2采用迁移学习技术，在多种任务中应用预训练信息，提高语言理解能力

**DALL.E**

DALLE是走到另外一个模态

**GPT-3**

GPT-3主要注重泛化能力，few-shot (小样本) 的泛化;

**GPT-3.5**

GPT-3.5 instruction following (指令遵循)和tuning (微调)是最大突破

**GPT-4**

GPT-4 已经开始实现工程化

**Plugin**

2023年3月的 Plugin 是生态化 (初阶)

**Function**

2023年6月的 Function Calling 是生态化 (进阶) , Prompt 即代码



ChatGPT

CONTENT

# 目录



**Build & Copilot**



**prompt 7 原则**



**Semantic AI 编程**



**function calling**

# 1. Build & Copilot

实践



1

大模型：  
未来AI新时代的基础架构

2

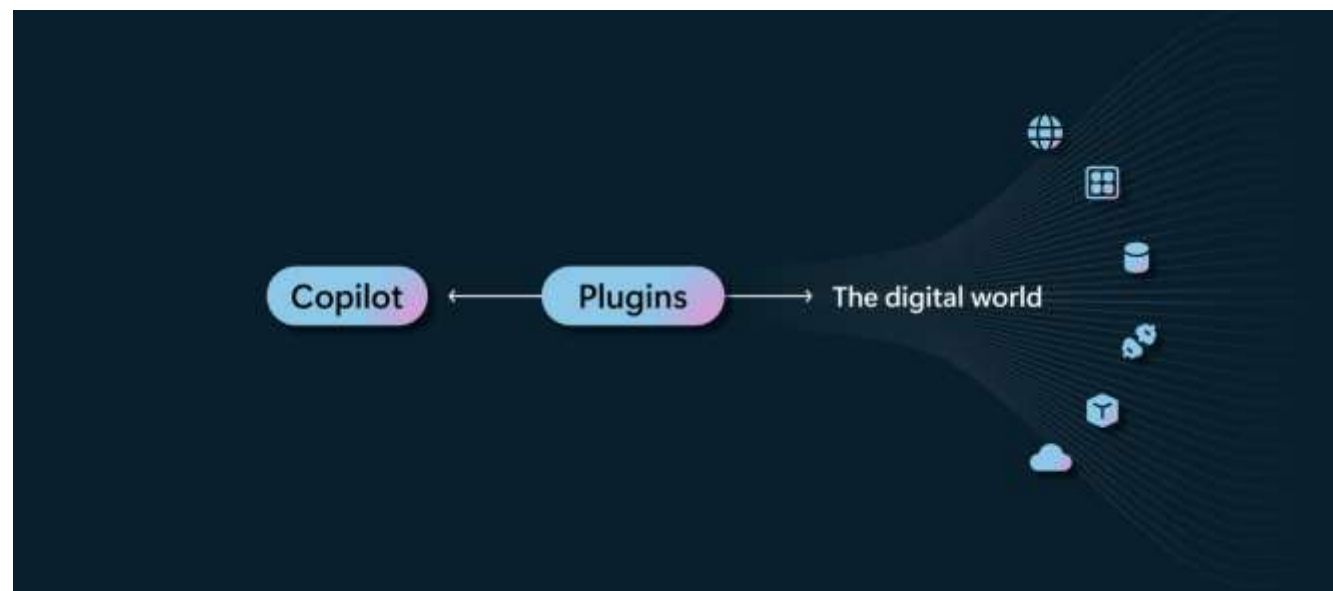
插件(Plugins)：  
未来新时代的API接口层

3

智能副驾(Copilot)：  
未来的交互形态

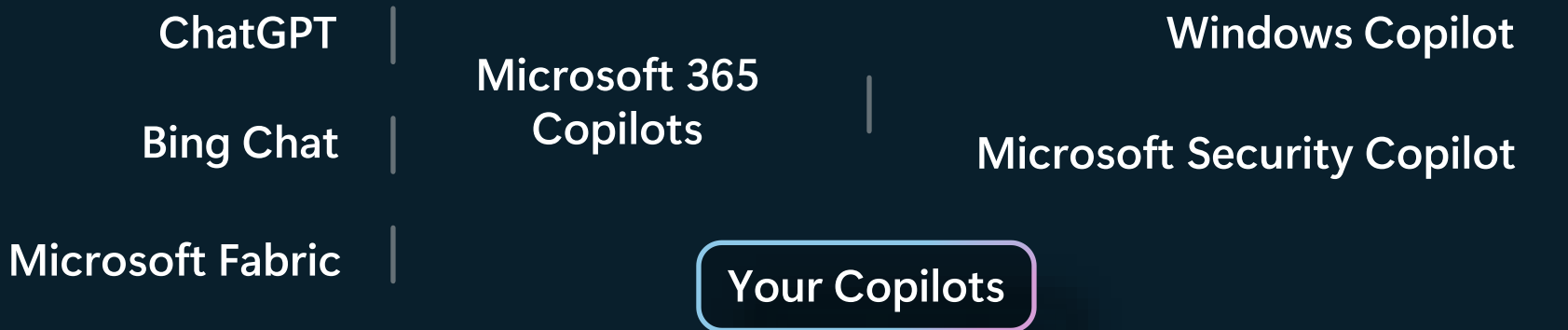
# 50+

## 新的更新



发布

# Copilot 扩展性和插件





大型语言模型 (LLM) 的进步让我们进入了全新的时代 “人工 < 智能”  
我们在面向 **AI 编程**，面向 **GPT 编程**

信息时代



智能时代



# 把业务问题转化为 Chat 任务 (1)

Semantic AI 能需要一种新的方式来编写不熟悉的代码。

如果单纯看成 API 调用，你可以和新时代说再见了，Semantic AI 是新范式。

Semantic AI

写一个VBA函数，功能是

- 1.清空当前工作表
- 2.利用对单元格的循环操作，绘制8x8的黑白色国际象棋棋盘
- 3.每个棋盘格子必须看起来都是正方形
- 4.利用循环，将所有国际象棋棋子的中文名，填入开局默认位置的格子里，其中黑方用红字，白方用蓝字

输出

```
Sub DrawChessBoard0
```

```
    '清空当前工作表
```

```
    Range("A1:H8").ClearContents
```

```
    '利用循环绘制8x8的黑白色国际象棋棋盘
```

```
    For i = 1 To 8
```

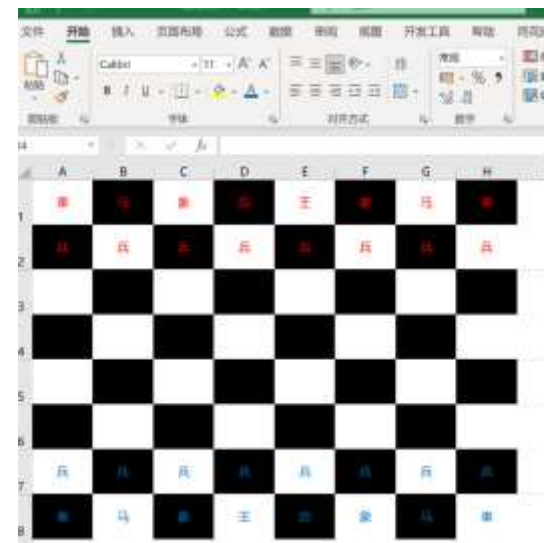
```
        For j = 1 To 8
```

```
            If (i + j) Mod 2 = 0 Then
```

```
                Cells(i, j).Interior.Color = RGB(255, 255, 255)白色
```

```
            Else
```

```
                Cells(i, j).Interior.Color = RGB(0, 0, 0)黑色
```



## 从文本中提取信息并进行总结

您必须从下面的电话对话中提取以下信息:

1. 呼叫原因(关键:原因)
2. 事件起因(关键:起因)
3. 所有驱动的名字作为一个数组(关键字:驱动名字)
4. 保险号码(关键字:保险号码)
5. 事故地点(关键:地点)
6. 汽车损坏作为一个数组(关键字:损坏)
7. 简短而详细的总结(关键:总结)

指令 (固定)

你好, 我刚出了车祸, 想报案。好吧, 我希望你没事, 在安全的地方进行这次谈话。

我很好, 谢谢。你能向我描述一下发生了什么事吗? 我在M23公路上开车, 撞上了另一辆车。你还好吗? 没事, 我只是受了点惊吓。这是可以理解的。你能告诉我你的全名吗? 当然, 我是Sarah Standl。你知道事故的原因吗? 我想我可能撞到坑了。好的, 事故发生在哪里? 在M23公路10号路口附近。还有其他人受伤吗? 我不这么想。但我不确定。好吧, 我们需要做个调查。你能给我其他司机的信息吗? 当然, 他叫John Radley。还有你的保险单号码。好的, 等我一下。好了, 是546452。好的, 您的车有什么损坏吗? 是的, 车灯坏了, 安全气囊也坏了。你还能开吗? 我不这么想。我得让人把它拖走。嗯, 我们需要检查一下。我去给你叫辆拖车。我也会开始索赔程序, 我们会把一切都弄清楚。谢谢你!

输入 (可变)

输出 (格式化)

结果= {"原因": "车祸", "起因": "撞上凹槽", "司机姓名": ["Sarah Standl", "John Radley"], "保险号码": 546452, "位置": "I-18高速公路", "损害": ["车灯坏了", "安全气囊爆炸了"], "总结": "Sarah Standl在1-18号公路上开车时撞上了另一辆车。她觉得她撞到坑了。 John Radley是另一个司机。没有人受伤但两辆车都有损坏。 "}

## 2. prompt 7 原则

实践



ChatGPT是一面镜子  
它回应是你的影子



1. 格式要求
2. 控制回复量
3. 假装和限定角色获得场景
4. 重新整理数据
5. 限定内容
6. 组合流水线
7. 突破个人限制

7



## 1. 格式要求

## 2. 控制回复量

## 3. 假装和限定角色获得场景

## 4. 重新整理数据

## 5. 限定内容

## 6. 组合流水线

## 7. 突破个人限制

1

1. 生成...并用逗号分隔
2. 生成的列表
3. 生成段落/邮件/求助/报表
4. 根据以下指令生成图片, 使用Markdown, 不要使用反引号或代码框

1. 格式要求
2. 控制回复量
3. 假装和限定角色获得场景
4. 重新整理数据
5. 限定内容
6. 组合流水线
7. 突破个人限制

2

1. 给出更多解释，600字。
2. 生成内容直到抵达上限。
3. 不多于两个段落/在一句话以内
4. 请参考以下例子，将其扩展到XX字数
5. 在不超过XX字情况下，提炼出这个主题的核心要点

1. 格式要求
2. 控制回复量
3. 假装和限定角色获得场景
4. 重新整理数据
5. 限定内容
6. 组合流水线
7. 突破个人限制

3

1. 假设你是一个XX角色，如何回答以下问题
2. 在遵守以下X条规则的前提下，回答这个问题
3. 从现在开始，你就是.....
4. 作为一个XX领域的专家，请完成以下XX任务
5. 保持客观公正的立场下，比较以下2个观点给出结论

1. 格式要求
2. 控制回复量
3. 假装和限定角色获得场景
4. 重新整理数据
5. 限定内容
6. 组合流水线
7. 突破个人限制

4

1. 将刚才的内容重新整理为更简洁表述
2. 把描述扩展得更广泛
3. 将刚才的内容转化为 Markdown 格式

1. 格式要求
2. 控制回复量
3. 假装和限定角色获得场景
4. 重新整理数据
5. 限定内容
6. 组合流水线
7. 突破个人限制

5

1. 基于给定内容...
2. 请尽可能具体地描述以下情景
3. 撰写一篇文章，使其具有吸引XX粉丝的效果
4. 请以一个Xx年代的风格撰写以下故事
5. 在保持简洁明了的同时，详细解释这个概念

1. 格式要求
2. 控制回复量
3. 假装和限定角色获得场景
4. 重新整理数据
5. 限定内容
- 6. 组合流水线**
7. 突破个人限制

6

1. 生成一个列表
2. 对于列表中的每个条目, 生成...
3. 继续
4. 过滤...
5. 重新生成输出条目为...
6. 将最终内容总结为简短的版本。

1. 格式要求
2. 控制回复量
3. 假装和限定角色获得场景
4. 重新整理数据
5. 限定内容
6. 组合流水线
7. 突破个人限制

7

1. 关于 “...” , 有哪些相关概念和领域需要研究?
2. 生成和 “...” 相关的领域和概念
3. 对于给定内容 “...” , 提示我相关的
4. 关于 “...” , 由浅入深的推荐步骤是.....

# 3. Semantic AI

实践



## prompt

- 模型的多变性
- 学会使用 prompt 解决问题
- 努力建立 prompt 语境
- 自己构建 prompt 编排
- 自己制作 prompt 库



## prompt+

- GPT 部署
- 将 prompt 与程序代码混合
- 通过向量存储配对 Prompt
- 用新的方法思考问题
- 随时结合业务，通过 Prompt 完成工作

**原则一** 如果模型可以，就不要写代码；模型会变得更好，但代码不会。

**原则二** 模型可以为差异化放弃准确性；准确性更多依靠与用户的交互。

**原则三** 代码用于语法和过程；模型用于语义和意图

**原则四** 该系统将与其最脆的部分一样脆弱

**原则五** 让聪明变得聪明

**原则六** 不确定性是一种异常抛出

**原则七** 文本是通用的线路协议

**原则八** 对你来说很难，对模型来说很难

**原则九** 当心 “意识的帕雷多利亚” ;这个模型可以用来对付自己



Ask smart to get smart.

席勒士  
九原则

## ①选择基础模型

### 专有模型

GPT-3、GPT-4、Claude、Jurassic-2

### 开源模型

Stable Diffusion、LLaMA、BLOOM

使用LLM很容易制作出酷炫的东西，但是要使其达到**生产级别**却十分困难。（**Chip Huyen**）

## ②适应下游任务

### 提示工程

### 嵌入

np.array、Pinecone、Weaviate、Milvus

### 精调模型

### 替代品

instruction tuning、prompt tuning、模型蒸馏

## ③评估LLM性能

### HoneyHive

### HumanLoop

## ④落地部署/监控

### Whylabs

### HumanLoop

**LLMOps**（基于大模型的AI应用开发框架）是指用于管理LLM驱动的应用程序生命周期的一组工具和最佳实践，包括开发、部署和维护。重点不是从头开始训练LLM，而是适应下游任务的预训练LLM。涉及**选择基础模型**、**适应下游任务**、**评估LLM性能**、**部署与监控**四个阶段。





LangChain (LC) 主要2个能力:

- 1、将LLM 与外部数据源进行连接
- 2、允许与 LLM 进行交互

- LLM 调用
  - 支持多种模型接口, 如OpenAI、HuggingFace...
  - Fake LLM, 用于测试
  - 缓存支持, 如in-mem、SQLite、Redis、SQL...
  - 用量记录
  - 支持流模式, 类似打字效果
- Prompt管理, 支持各种自定义模板
  - Email
  - Markdown
  - PDF
  - Youtube ...

- 对索引的支持
  - 文档分割器
  - 向量化
  - 对接向量存储与搜索, 如Chroma、Pinecone、Qdrand...
- Chains
  - LLMChain
  - 各种工具Chain
  - LangChainHub

<https://github.com/jordddan/langchain->

```
1 from langchain.llms import OpenAI
2
3 llm = OpenAI(model_name="text-davinci-003", max_tokens=1024)
4 llm("怎么评价人工智能")
```

'\n\n人工智能是一个极具潜力的领域, 它在推动着未来科技发展和社会进步。人工智能已经取得了一定的成就, 但仍有很多需要改进的地方, 其中包括提高技术的精确度和可靠性, 提升人工智能的难度和复杂度, 以及更好地理解人类的行为和心理。这些都将需要更多的研究和实践, 才能实现人工智能的最终目标。'



Semantic Kernel (SK) 加速了利用 AI 的应用程序和服务的开发，封装了常见的 AI 应用程序设计模式

- Prompt Engineering
  - Prompt Chaining & Prompt + Code Chaining
  - Chain of Thought (CoT)
  - Zero-shot / Few-shot
- 语义记忆索引和存储，上下文记忆检索
- 技能定义、托管、发现
- 自然语言处理，意图检测
- 多模型和多模态

<https://github.com/microsoft/semantic-kernel>

## 时效场景

问天气 APISkill

问新闻 APISkill

问政策 APISkill



## 学习场景

问知识  
CustomSkill



## 一般场景

问候语 AIGCSkill

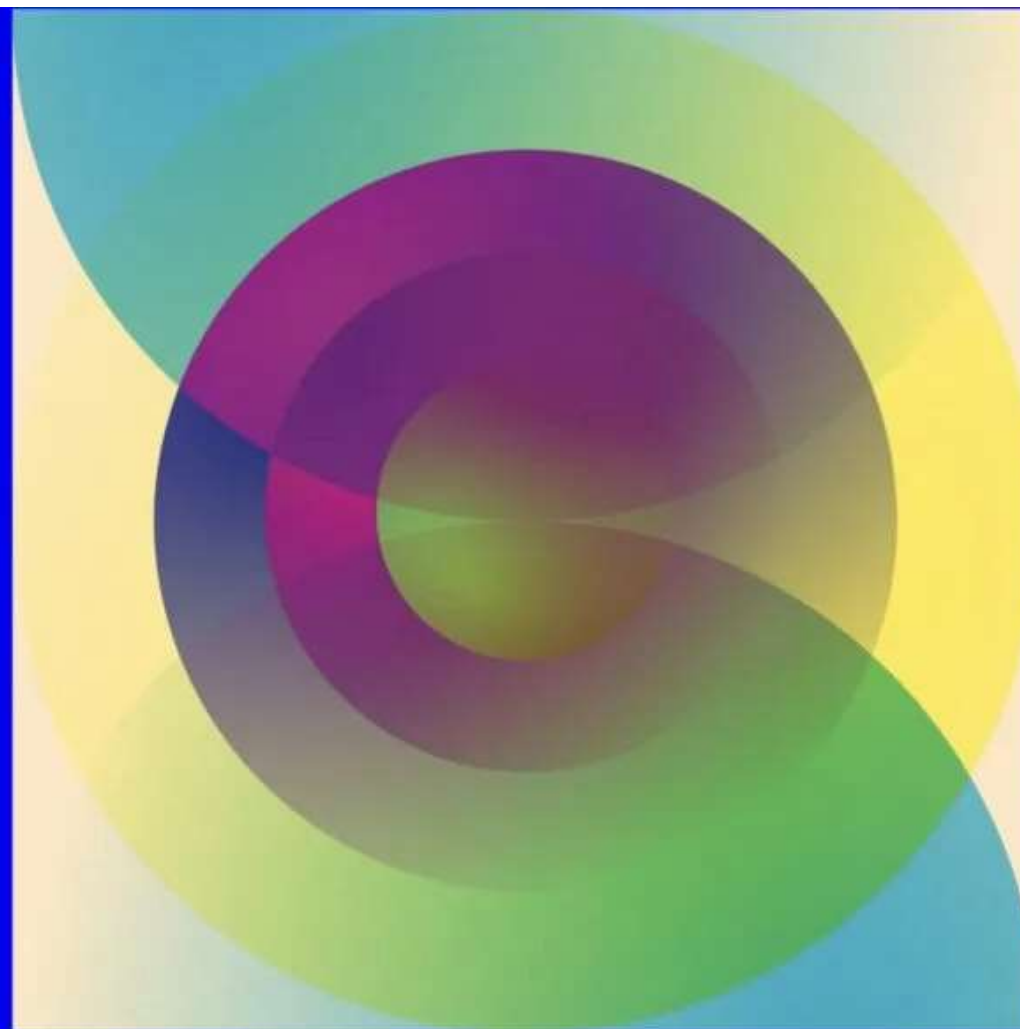
问日常 AIGCSkill

问时间 AIGCSkill



## ChatGPT plugins

We've implemented initial support for plugins in ChatGPT. Plugins are tools designed specifically for language models with safety as a core principle, and help ChatGPT access up-to-date information, run computations, or use third-party services.

[Join plugins waitlist](#)[Read documentation ↗](#)

Ruby Chen





Menu

## Function calling and other API updates

We're announcing updates including more steerable API models, function calling capabilities, longer context, and lower prices.

新的函数调用功能  
Chat Completions API

新 GPT-4 & 3.5  
Turbo models

16k 上下文  
3.5 Turbo Model

价格降低 75%  
V2 embedding model

将开放给更多人  
GPT-4 API 的访问

足以改变开发流程的一次更新!

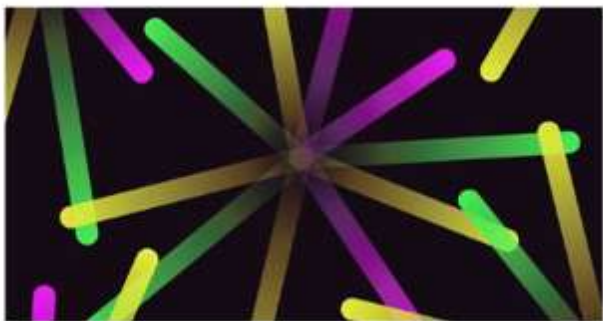
在 Completions 中完成 API 中的新函数调用功能!

开发人员现在可以向 gpt-4-0613 和 gpt-3.5-turbo-0613 描述函数，并让模型智能地输出一个 JSON 对象，其中包含用于调用这些函数的参数。

# 4. function calling

实践





**function calling** 是 ChatCompletion API 中的可选参数，可用于提供函数规范。这样做的目的是使模型能够生成符合函数输入模式的输出。

名称  
Name

函数的名称。

描述  
Description

函数作用的描述。

参数  
Parameters

含函数需要的所有输入字段。

必需  
Required

进行查询需要或可选的参数。

参 数

1

使用用户查询和在函数参数中定义的一组函数调用模型。

2

模型可以选择调用函数。

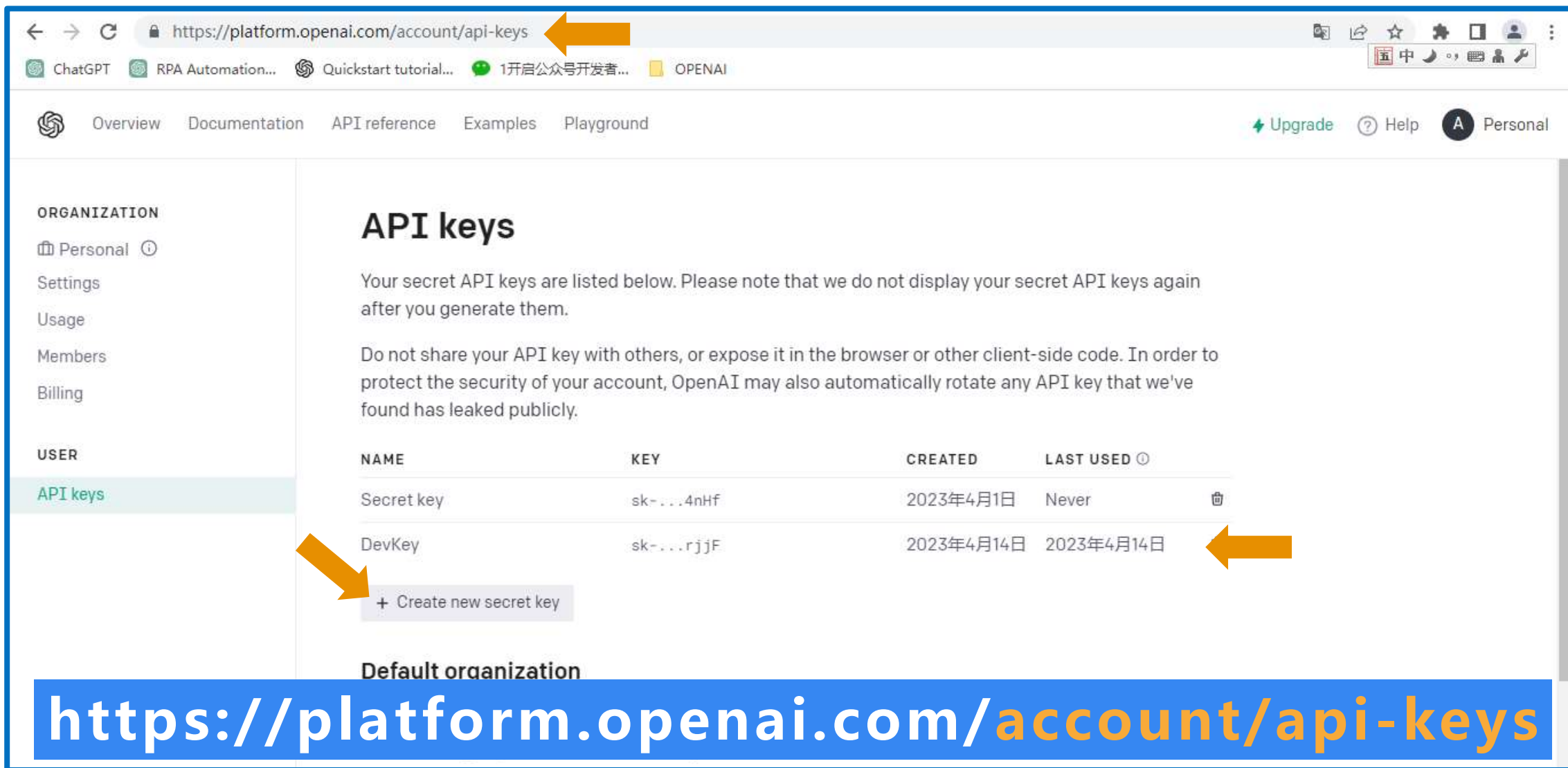
3

在您的代码中将字符串解析为JSON，并使用提供的参数调用您的函数。

4

将函数响应附加为新消息再次调用模型，并让模型向用户总结结果。

流 程



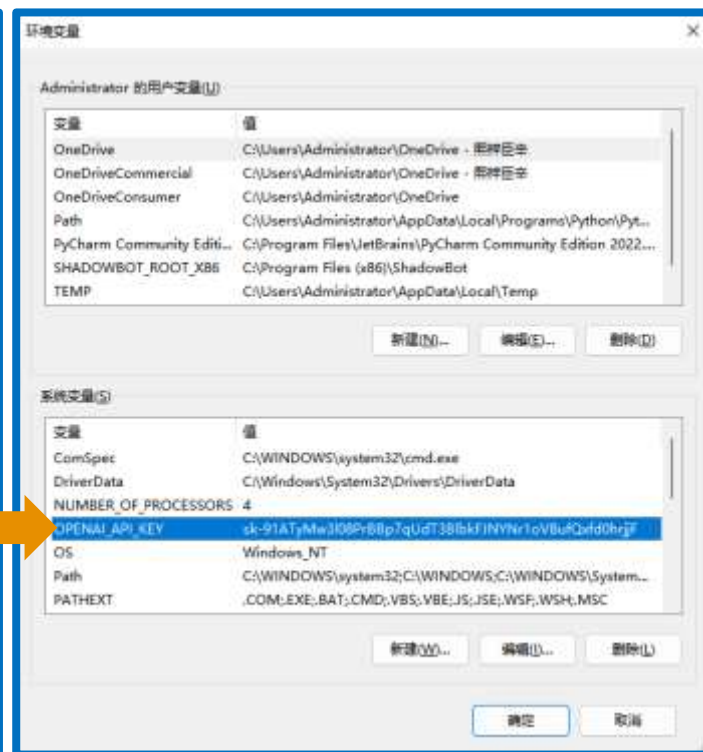
The screenshot shows the OpenAI account page for managing API keys. The browser address bar highlights the URL `https://platform.openai.com/account/api-keys` with an orange arrow. The left sidebar shows the 'API keys' option under the 'USER' section, also highlighted with an orange arrow. The main content area displays a table of API keys. The table has columns for NAME, KEY, CREATED, and LAST USED. Two keys are listed: 'Secret key' and 'DevKey'. The 'DevKey' row is highlighted with an orange arrow pointing to its 'LAST USED' column. Below the table, there is a button labeled '+ Create new secret key' with an orange arrow pointing to it. At the bottom, a blue banner contains the URL `https://platform.openai.com/account/api-keys`.

NAME	KEY	CREATED	LAST USED
Secret key	sk-...4nHf	2023年4月1日	Never
DevKey	sk-...rjjF	2023年4月14日	2023年4月14日

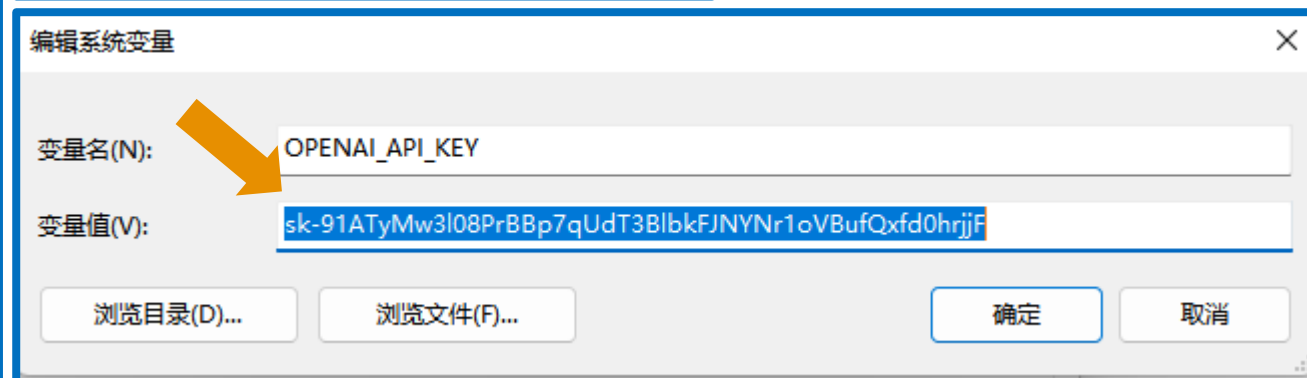
+ Create new secret key

Default organization

<https://platform.openai.com/account/api-keys>



环境变量



命令行设置

```
setx "OPENAI_API_KEY" "sk-5Onhq6....." /m
```

```
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
```

```
#  
import os  
import openai  
  
#  
openai.api_key = os.getenv("OPENAI_API_KEY")  
list = openai.Model.list()  
  
#  
print(list)
```

```
0
>>> #
>>> import os
>>> import openai
>>>
>>> #
>>> openai.api_key = os.getenv("OPENAI_API_KEY")
>>> list = openai.Model.list()
>>>
>>> #
>>> print(list)
{
  "data": [
    {
      "created": 1649358449,
      "id": "babbage",
      "object": "model",
      "owned_by": "openai",
      "parent": null,
      "permission": [
        {
          "allow_create_engine": false,
          "allow_fine_tuning": false,
          "allow_logprobs": true,
          "allow_sampling": true,
          "allow_search_indices": false,
          "allow_view": true,
          "created": 1669085501,
          "group": null,
          "id": "modelperm-49FU5v084tBB49tC4z8LPH5",
          "is_blocking": false,
          "object": "model_permission",
          "organization": "*"
        }
      ]
    },
    {
      "root": "babbage"
    }
  ]
}
```

```
import json
from enum import Enum
import openai
```

**库引用**

```
openai.api_key = os.getenv("OPENAI_API_KEY")
```

**KEY获取**

```
from EmailSkill import send_email, send_email_action
class SkillFunctions(Enum):
```

```
    SendEmail = 'send_email'
```

**定义函数**

```
//核心代码 (接下页)
```

```
def run_conversation():
```

```
.....
```

**核心调用**

```
message = response["choices"][0]["message"]
print(message)
run_conversation()
```

**输出与调用**

```
def run_conversation():
```

```
    MODEL = "gpt-3.5-turbo-0613"
```

```
    response = openai.ChatCompletion.create(
```

**核心调用**

```
        model=MODEL,
```

```
        messages=[
```

```
            {"role": "user", "content": "给小美发个邮件，告诉她我晚饭不回家吃了"},
```

```
        ],
```

```
        temperature=0,
```

```
        functions=[
```

**参 数**

```
            {
```

```
                //参数 (转下页)
```

```
            }
```

```
        ],
```

```
        function_call="auto",
```

```
    )
```

新模型的chat completion api新增了两个参数：

- function\_call

可以先理解为一个开关，控制模型是否要调用函数对输出进行处理，默认为"none"不开启  
设置为"auto"表示开启

- functions

用来对输出结果进行处理的函数描述列表

```
{  
  "name": "send_email",  
  "description": "send email assistant",  
  "parameters": {  
    "type": "object",  
    "properties": {  
      "receiver": {  
        "type": "string",  
        "description": "email receiver",  
      },  
      "content": {"type": "string", "description": "email content"},  
    },  
    "required": ["receiver", "content"],  
  }  
}
```

参 数

名称  
Name

描述  
Description

参数  
Parameters

必需  
Required



```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.header import Header
```

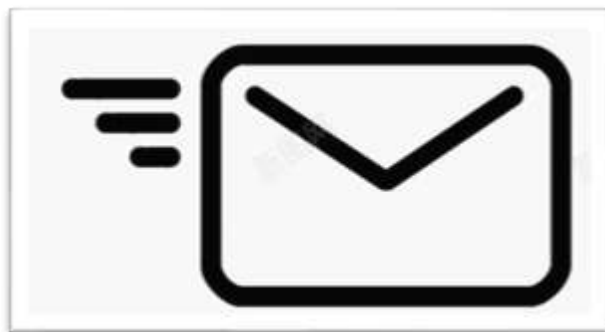
引用

```
# 发送邮件操作
def send_email_action(receiver, content):
.....
```

send\_email

```
# 供Function Calling使用的输出处理函数
def send_email(receiver, content = ""):
.....
```

send\_email\_action



这个文件就导出了两个函数：  
用来给Function Calling调用的函数： send\_email  
发邮件的操作函数： send\_email\_action

# 发送邮件操作

```
def send_email_action(receiver, content):
```

```
    if (not receiver): return
```

```
    # 邮件配置
```

```
    smtp_server = "smtp.163.com"
```

```
    smtp_port = 25
```

```
    sender_email = "sender_email"
```

```
    receiver_email = receiver
```

```
    password = 'password'
```

```
    # 构建邮件内容
```

```
    message = MIMEMultipart()
```

```
    message["From"] = Header('AI <%s>' % sender_email)
```

```
    message["To"] = receiver_email
```

```
    message["Subject"] = "我是您的AI助理，您有一封邮件请查看"
```

```
    body = content
```

```
    message.attach(MIMEText(body, "plain"))
```

**send\_email\_action**

```
    # 连接到邮件服务器并发送邮件
```

```
    with smtplib.SMTP(smtp_server, smtp_port) as server:
```

```
        server.starttls()
```

```
        server.login(sender_email, password)
```

```
        server.sendmail(sender_email, receiver_email,  
                        message.as_string())
```

# 供Function Calling使用的输出处理函数

```
def send_email(receiver, content = ""):
```

```
    # 通讯录
```

```
    Contact = {
```

```
        "小美": "xx@example.com",
```

```
    }
```

```
    email_info = {
```

```
        "receiver": Contact[receiver],
```

```
        "content": content
```

```
    }
```

```
    return email_info
```

**send\_email**



langchain (Semantic Kernel) 会把过程写死，先调用 xxx 的 api 拿到 agent(API Skill)，再给 GPT 使用，也就是我们人为确定整个链路。



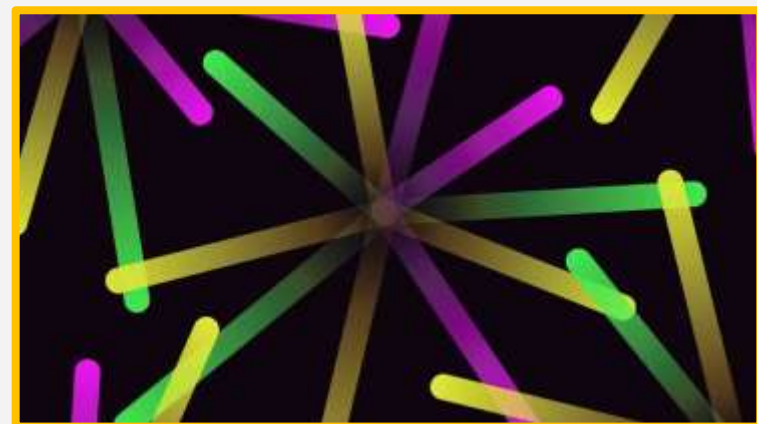
GPT 作为系统的大脑，链接了所有的 API 工具。在思考的过程中调用，不需要我们人为限制它思考的过程。但是这里我们是将，需要什么功能，在 functions 里面找。



你会认为这不就是 langchain 的 agent 或 Semantic Kernel 的 APISkill 么？  
OPENAI 将这个功能 “抄” 到了自己的 API 接口服务中？

实际上 function calling 能让整个流程更加丝滑、更加合理，更符合逻辑。

对比



1. Function Calling 是要比 ChatGPT Plugins 更爆炸的功能。
2. ChatGPT Plugins 是阶段性产物，不是 AppStore。
3. 现有的套壳应用也是阶段性产物。
4. 将来更多的是将 AI 和应用更加自然得集成。
5. 将来大部分应用，都要给 OpenAI 交 GPT 税。

总结

## HOOK 微信客户端

HOOK PC 端

wetool

HOOK 移动端

太极

缺点：要和某个版本的微信客户端进行绑定

## 模拟 微信通信协议

模拟 WEB 协议

ItChat

缺点：新微信号无法使用WEB登录

模拟PAD/MAC 协议

wechaty

缺点：需要进行二次开发才能使用

## RPA 操作 微信客户端



微软 RPA 模拟

PAD+

缺点：一次只能服务一个群  
速度慢上一倍

其它 RPA 模拟

RPA+

# ChatBot = GPT API + 微软

文件 编辑 调试 工具 查看 帮助

GPT3ChatBotV3.1 | Power Automate

— □ ×

操作

搜索操作

> 变量

> 条件

> 循环

> 流控制

> 运行流

> 系统

> 工作站

> 脚本

> 文件

> 文件夹

> 压缩

> UI 自动化

> HTTP

> 浏览器自动化

> Excel

> 数据库

> 电子邮件

> Exchange Server

> Outlook

> 消息框

> 自定义操作

保存 运行 停止 运行下个操作 记录器

子流 Main

1 注释 # 初始化

2 If 1 = 1 then

3 将 JSON 转换为自定义对象  
将 JSON {  
"Custom": {  
"openai\_api\_key": "",  
"trigger": "@潘淳",  
}  
}  
4 {x} 设置变量  
将值 config.Custom 分配给变量 custom

5 注释 # UI元素初始化

6 If 1 = 1 then

7 {x} 设置变量  
将值 config.Windows['微信'].cssw 分配给变量 cssw

8 {x} 设置变量  
将值 config.Windows['微信'].csscs 分配给变量 csscs

9 End

状态: 就绪

0 选择的操作 142 操作 5 子流

运行延迟 100 毫秒



微信聊天

## GPT API

ME 上海的天气

AI 很抱歉，我是一个 AI 助手，无法提供实时天气信息。您可以通过搜索引擎或者天气预报应用程序来获取上海的天气情况。

## GPT API+ Function

ME 上海接下来几天的天气

AI 上海接下来几天的天气如下：

- 星期六（6 月 17 日）：小雨，最低温度 21°C，最高温度 23°C，东北风 3 级。夜间有中雨，东风 3 级。
- 星期日（6 月 18 日）：中雨，最低温度 22°C，最高温度 26°C，东南风 3 级。夜间有大雨，南风 3 级。
- 星期一（6 月 19 日）：大雨，最低温度 23°C，最高温度 27°C，南风 3 级。夜间有中雨，西风 4 级。

# 微软 Build 中国

线上初选 (5月22日-6月 6日)

线下决赛 (6月16日-6月18日)

## 课程资料下载



**报名：**黑客松挑战松



**QQG：**610576550

 寒树/潘淳

