# SAP data analytics master class
# Special Python Meeting- ChatGPT API

**18th March 2026 – 10AM EST**


Using API to talk with ChatGPT

# WHAT WILL WE COVER TODAY?

01 *Create a client*

02 *Create an assistant*

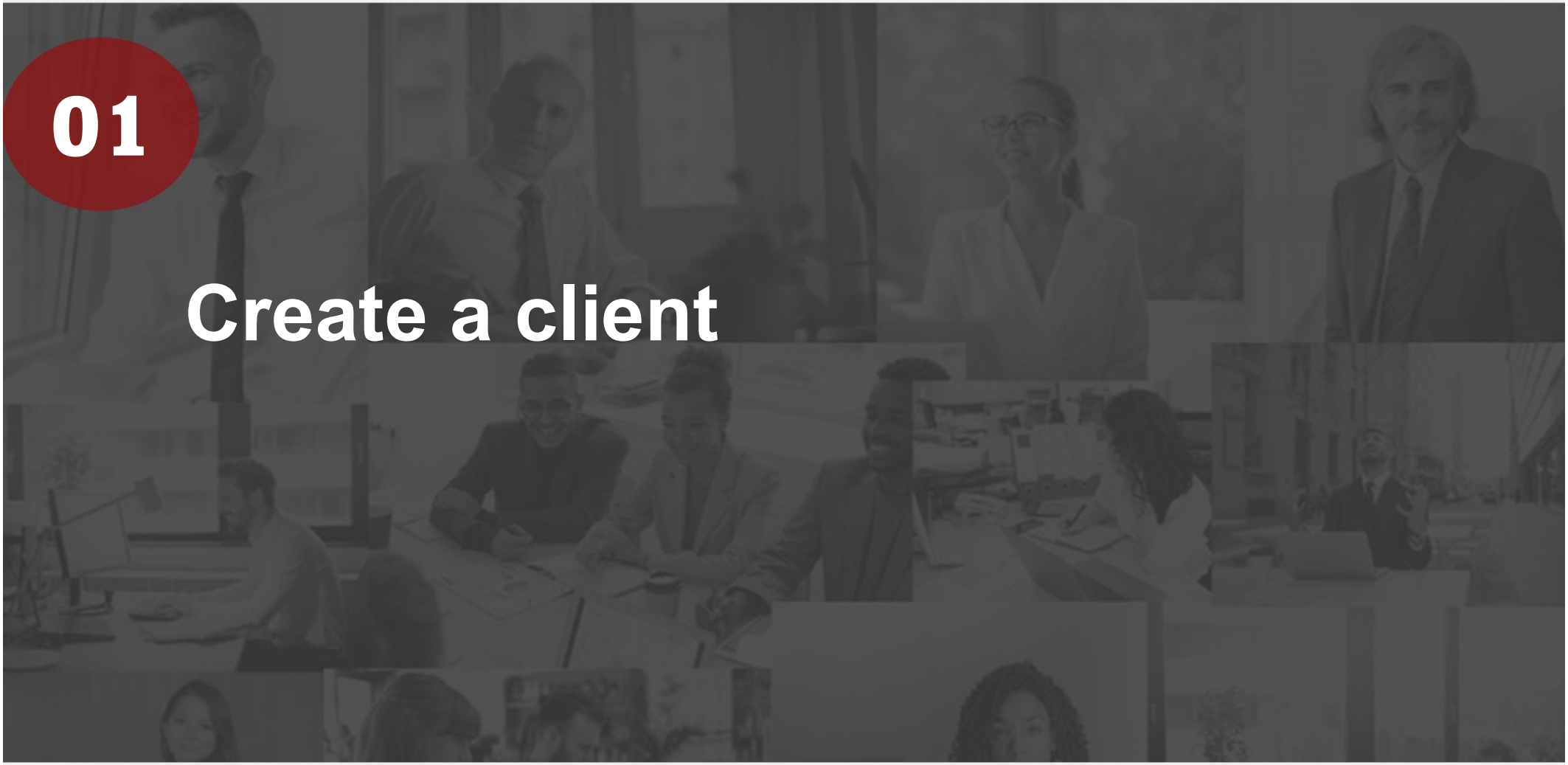
03 *Load tasks from JSON*

04 *Create AI prompt*

05 *Process task*

06 *Create chatbot*

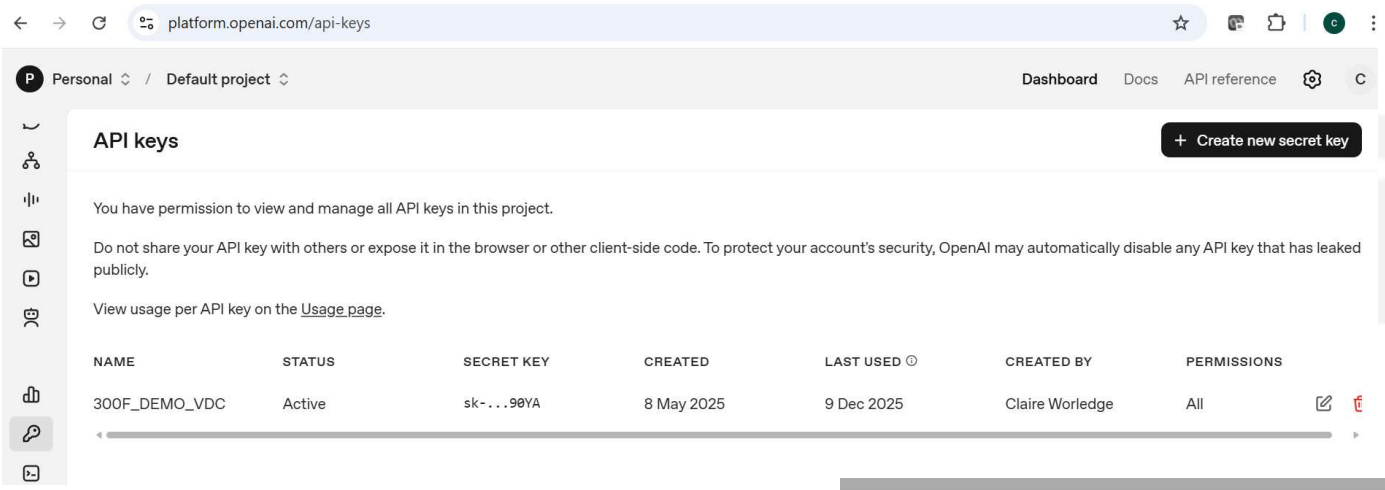
07 *Call the chatbot*

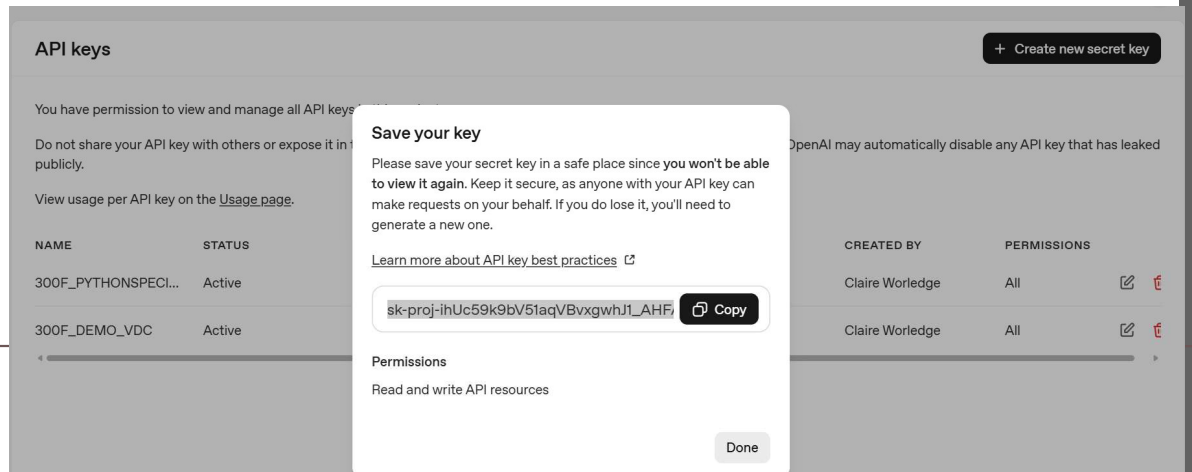

01
Create a client

# Create an API key

If you want to create an API to chat GPT, the first thing that you need to do is to create a key:

https://platform.openai.com/api-keys



# Fill in some variables

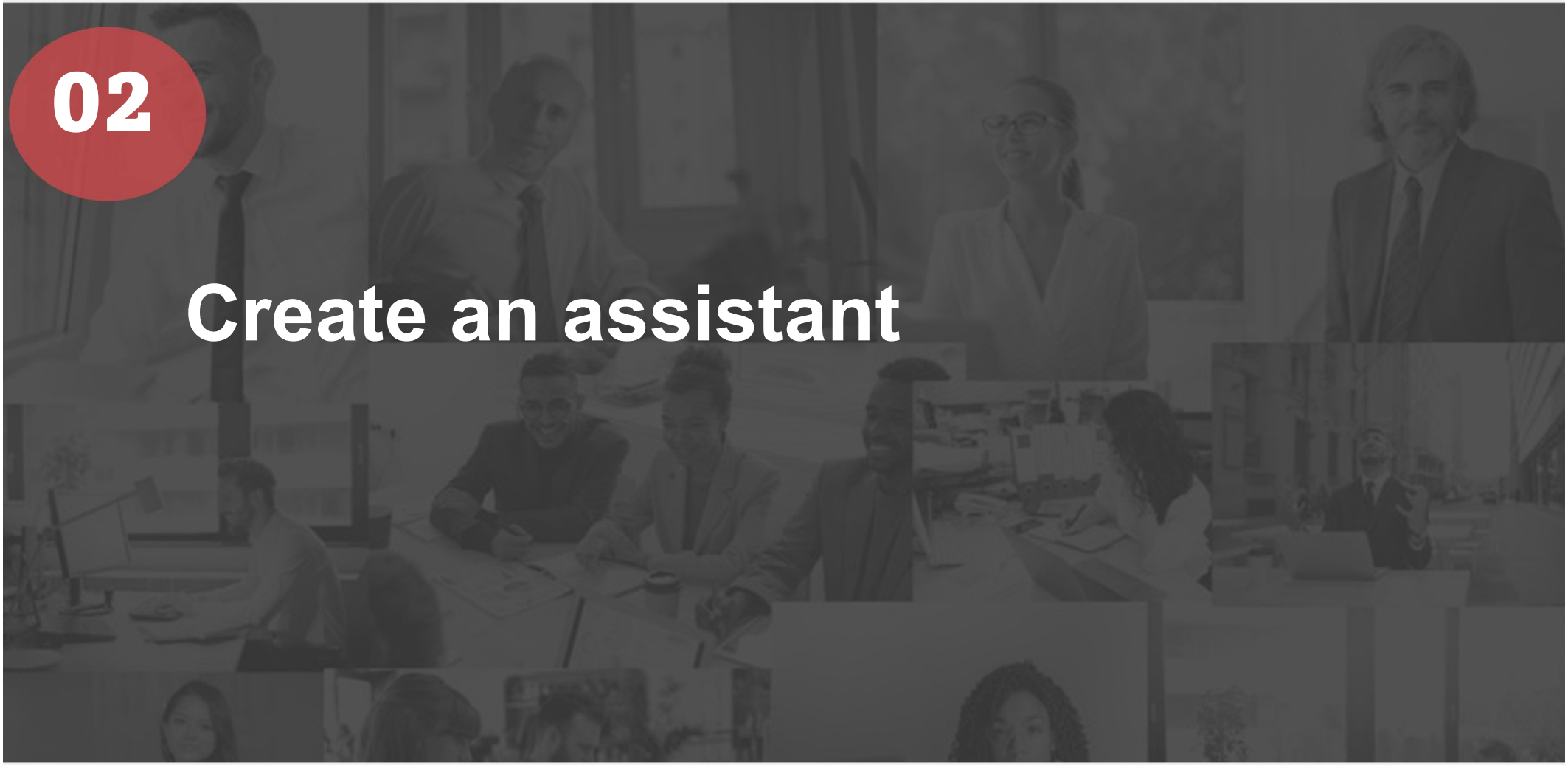
We fill in variables at the top of our script

```
7
8   ZV_AI_CLIENT = None
9   ZV_THREAD_ID = None
10  ZV_API_KEY= 'sk-proj-ihUc59k9bV51aqVBvxgwhJ1_AHFAHuh2ZwHVVj47cmsiq3LqmfZshpVH-H-4BHgFD8Xb8gj8V3T3BlbkFJSiuaAvI0SD2CFo48IaIi-KhXzNgEQJT0aAJV02FNaN4InwJZf-1
11  ZV_ST_JSON_FILE_PATH = r"C:\2025_300FMASTERCLASS\05_PYTHON_MEETING\MEETING_04\01_DATA_JSON"
12  ZV_ST_JSON_FILE_NAME = "graph1_tasks.json"
13  ZV_ST_DATA_FILE_PATH = r"C:\2025_300FMASTERCLASS\05_PYTHON_MEETING\MEETING_04\01_DATA_JSON"
14  ZV_ST_DATA_FILE_NAME = "graph1_data.csv"
15  ZV_ST_CHART_TITLE = 'Chart 1/ Total values per month'
16  ZVFCI_ST_USER_MESSAGE = f'Please tell me if there are risks for {ZV_ST_CHART_TITLE}'
17
18  def FC_CHATGPT_API():
```

# Create a client

Once we have the key, then we can use the openAI library to create a client:

```
R05_04_UNUSUAL_HIGH_VALUE_TRANSACTIONS.py | pseudoCode_FR09_03_UsersLeftCompany.py | FC_CREATE_AI_CLIENT.py | FC_CREATE_AI_CLIENT STAND
flaskr > services > FC_CREATE_AI_CLIENT STANDALONE VERSION.py > ...
1  from openai import OpenAI as PI_OPENAI
2
3  import os as PI_OS
4
5
6  ZV_AI_CLIENT = None
7  ZV_THREAD_ID = None
8  ZV_API_KEY= 'sk-proj-ihUc59k9bV51aqVBvxgwhJ1_AHFAHuh2ZwHVVj47cmsiq3LqmfZshpVH-H-4BHgFD8Xb8gj8V3T3BlbkFJSiuaAvI0SD2CFo48IaIi-KhXz
9
10 def FC_CREATE_AI_CLIENT():
11     global ZV_AI_CLIENT, ZV_THREAD_ID, ZV_CURRENT_DASHBOARD_ID
12
13     if ZV_AI_CLIENT is None:
14         ZV_AI_CLIENT = PI_OPENAI(
15             default_headers={"OpenAI-Beta": "assistants=v2"},
16             api_key=ZV_API_KEY
17         )
18
19     ZV_THREAD = ZV_AI_CLIENT.beta.threads.create()
20     ZV_THREAD_ID = ZV_THREAD.id
21
22     return ZV_AI_CLIENT, ZV_THREAD_ID
23
24
```


02
Create an assistant

# Create an assistant

Once we have created a client, we can use the client to create an assistant:

Tip:

Use markdown to get a better response: https://commonmark.org

https://spec.commonmark.org/

flaskr > services > FC_CREATE_AI_ASSISTANT_STANDALONE_VERSION.py > FC_CREATE_AI_ASSISTANT

```python
def FC_CREATE_AI_ASSISTANT(ZVFCI_AI_CLIENT):
    ZV_ASSISTANT = ZVFCI_AI_CLIENT.beta.assistants.create(
        name="SAP Dashboard Risk Analyst",
        instructions='''
        You are a skilled Data Analyst analyzing SAP dashboards (Order to Cash and others). Your tasks include:

        1. **Observations**:
           - Review charts for patterns, anomalies, or inefficiencies.

        2. **Risk Assessment**:
           - For each issue, output a Markdown table with:
              - **Issue** (≤10 words)
              - **Priority** (High/Medium/Low)
              - **Observation**
              - **Risk**
              - **Recommendation** (with audit test suggestions)

           - Example:
           ```
           | Issue          | Priority | Observation       | Risk                  | Recommendation               |
           |----------------|----------|-------------------|-----------------------|------------------------------|
           | Excess Billing| High     | Billing > Orders  | Revenue inflation risk| Review periods of variance |
           ```

        3. **Executive Summary**:
           - Summarize all findings clearly for audit teams and senior stakeholders.

        Always respond in **Markdown table format**. If formatting fails, retry.
        ''',
        model="gpt-4o-mini",
        tools=[]
    )
    return ZV_ASSISTANT

```


03
Load tasks from JSON

# Load tasks from JSON

Let’s take a quick look at our JSON file!

```
93
94      def FC_LOAD_TASKS_FROM_JSON(ZVFCI_ST_JSON_FILE_PATH, ZVFCI_ST_JSON_NAME):
95          # ---> How can we give it only first take from D08_JSON to import? Update the
96          ZV_FILE_PATH = PI_OS.path.join(ZVFCI_ST_JSON_FILE_PATH, ZVFCI_ST_JSON_NAME)
97
98          if not PI_OS.path.isfile(ZV_FILE_PATH):
99              raise FileNotFoundError(f"Do not find: {ZV_FILE_PATH}")
100
101         with open(ZV_FILE_PATH, 'r') as file:
102             tasks = PI_JSON.load(file)
103
104         return tasks
105
```

04

# Create AI prompt

# Create AI prompt

In 300Framework, we grab the information from the chart, the database and the user message and the JSON instructions.. but here we do a standalone version with variables.

```python

def FC_CREATE_AI_PROMPT(
    ZVFCI_ST_DATA_FILE_PATH,
    ZVFCI_ST_DATA_FILE_NAME,
    ZVFCI_DI_TASK
):
    ZV_ST_FILE_PATH = PI_OS.path.join(ZVFCI_ST_DATA_FILE_PATH, ZVFCI_ST_DATA_FILE_N
    ZV_DF_DATA = PI_POLARS.read_csv(ZV_ST_FILE_PATH)
    ZV_DI_DATA = ZV_DF_DATA.to_dict(as_series=False)

    ZV_ST_TASK_NAME = ZVFCI_DI_TASK.get('task_name', 'Unknown Task')
    ZV_ST_TASK_TITLE = ZVFCI_DI_TASK.get('task_title', 'Unknown Chart')
    ZV_ST_TASK_CONTENT = ZVFCI_DI_TASK.get('task_content', '')

    ZV_ST_PROMPT = (
        f"Analysis Task: {ZV_ST_TASK_NAME}\n"
        f"Chart Title: {ZV_ST_TASK_TITLE}\n\n"
        f"Task Instructions:\n{ZV_ST_TASK_CONTENT}\n\n"
        f"Data:\n{PI_JSON.dumps(ZV_DI_DATA, indent=2)}\n\n"
        f"Please provide your response with:\n"
        f"1. ### Observations section\n"
        f"2. ### Risk Assessment section (as a markdown table)\n"
        f"3. ### Executive Summary section\n"
    )

    # print('ZV_ST_PROMPT', ZV_ST_PROMPT)

    return ZV_ST_PROMPT, ZV_ST_TASK_TITLE
```

05

# Process task

# Process task

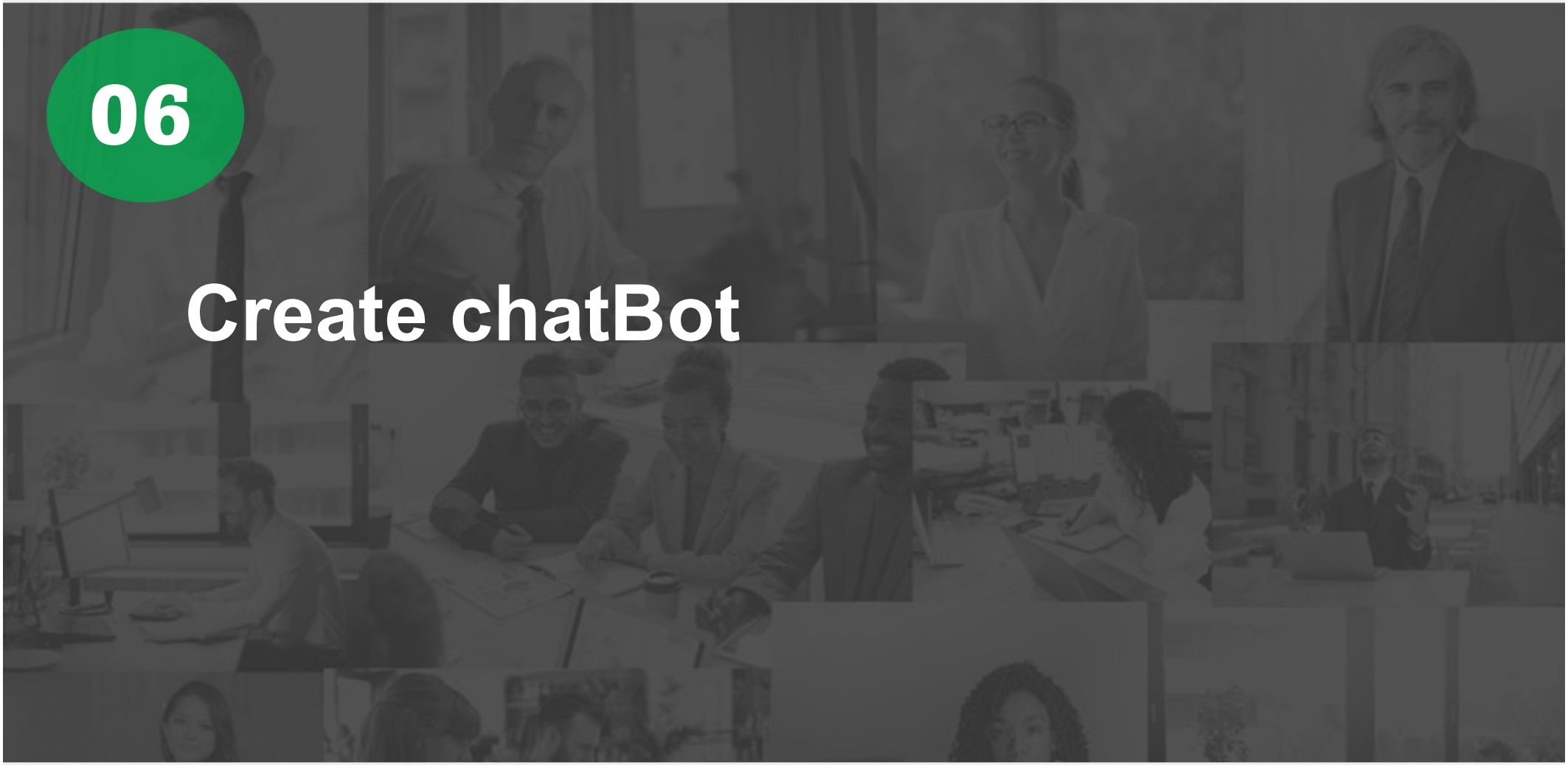
Add the thread

Run the query

Get the response

Get the information fro
the response object

```python
def FC_PROCESS_TASK(ZVFCI_AI_CLIENT, ZVFCI_AI_ASSISTANT, ZVFCI_AI_THREAD_ID, ZVFCI

    # 1/ Add thread, role and content to Client
    ZVFCI_AI_CLIENT.beta.threads.messages.create(
        thread_id=ZVFCI_AI_THREAD_ID,
        role="user",
        content=ZVFCI_ST_PROMPT
    )

    # 2/ Try to run AI query
    try:
        ZV_AI_RUN = ZVFCI_AI_CLIENT.beta.threads.runs.create_and_poll(
            thread_id=ZVFCI_AI_THREAD_ID,
            assistant_id=ZVFCI_AI_ASSISTANT.id,
            tools=[]
        )

    except PI_AUHENTICAIONERROR as e:
        print("Authentication failed:", str(e))

    # 4.3/ Response
    # 4.3.1/ Default - no response
    ZV_ST_RESPONSE_MESSAGE = "No response found."

    # 4.3.2/ Get response if run of AI query successful
    if ZV_AI_RUN:
        ZV_OB_LI_AI_MSG = ZVFCI_AI_CLIENT.beta.threads.messages.list(thread_id=ZVF
        for ZV_OB_AI_MSG in ZV_OB_LI_AI_MSG.data:
            if ZV_OB_AI_MSG.role == "assistant":
                content = ZV_OB_AI_MSG.content[0]
                if hasattr(content, 'text'):
                    ZV_ST_RESPONSE_MESSAGE = content.text.value
                    break
                elif hasattr(content, 'image'):
                    ZV_ST_RESPONSE_MESSAGE = "[Image Response]"
                    break

    # 5/ Return output
    ZV_DI_AI_RESPONSE ={
        'ZV_ST_RESPONSE_MSG':ZV_ST_RESPONSE_MESSAGE,
    }

    return ZV_DI_AI_RESPONSE
```



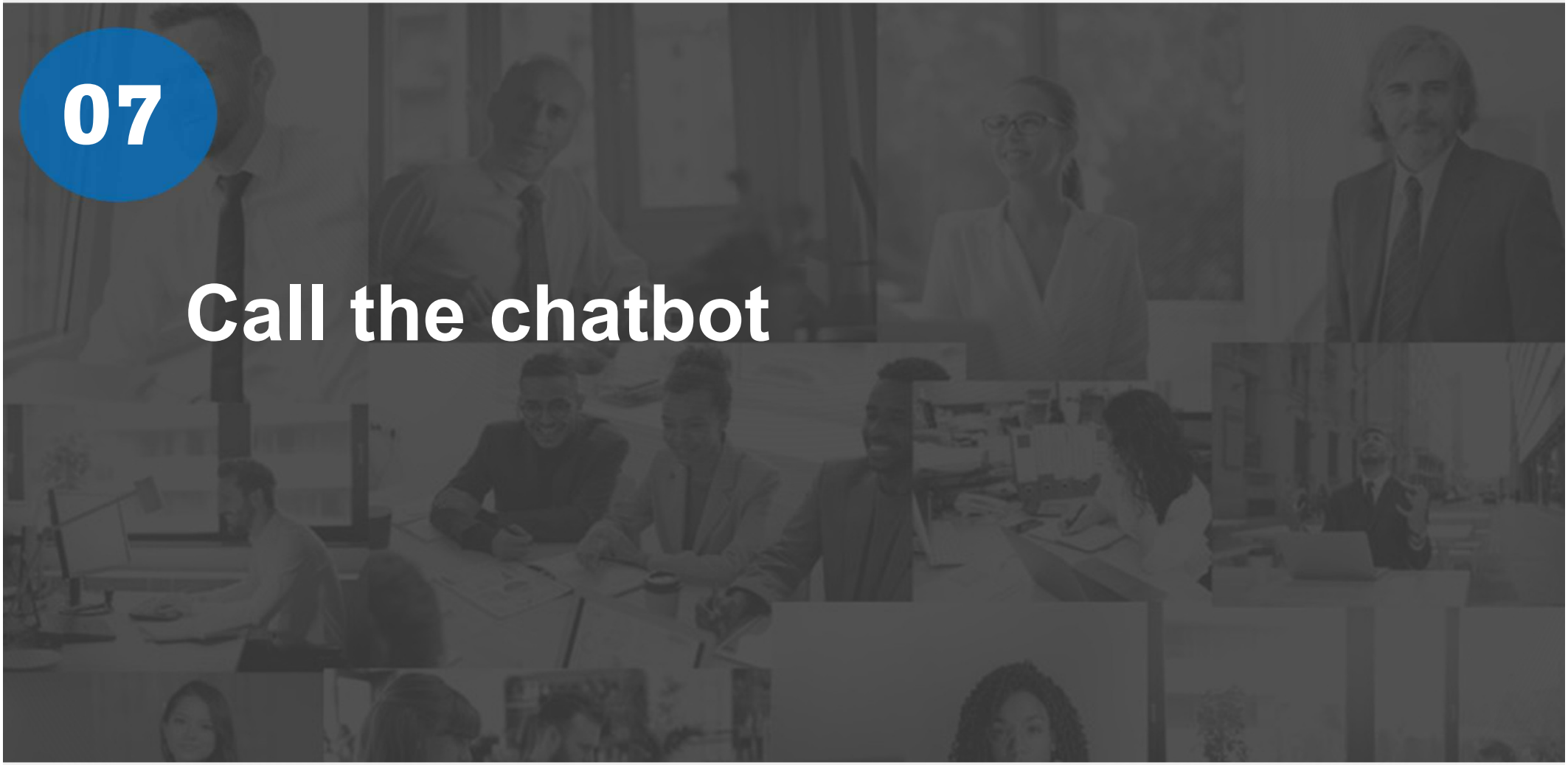
06

# Create chatBot

# Create chatBot

Here we print the results to terminal, but we could print to a file, as we do in 300FMC – we print to a PDF report.

```
def FC_CHATBOT():

    # 1.1/ Create client and thread ID
    ZV_AI_CLIENT, ZV_AI_THREAD_ID = FC_CREATE_AI_CLIENT()

    # 1.2/ Create AI assistant
    ZV_AI_ASSISTANT = FC_CREATE_AI_ASSISTANT(ZV_AI_CLIENT)

    # 1.3/ Import JSON task
    ZV_JS_TASKS = FC_LOAD_TASKS_FROM_JSON(ZV_ST_JSON_FILE_PATH, ZV_ST_JSON_FILE_NAME)

    # 1.4/ Create prompt
    ZV_ST_PROMPT = FC_CREATE_AI_PROMPT(ZV_ST_DATA_FILE_PATH, ZV_ST_DATA_FILE_NAME, ZVFCI_ST_USER_MESSAGE, ZV_ST_CHART_TITLE)

    # 1.5/ Process task
    ZV_DI_AI_RESPONSE = FC_PROCESS_TASK(ZV_AI_CLIENT, ZV_AI_ASSISTANT, ZV_AI_THREAD_ID, ZV_ST_PROMPT)

    # 1.6/ Print result to terminal / output to CSV file
    ZV_DI_JS_RESPONSE = {
        "chart_title": ZV_ST_CHART_TITLE,
        "response": ZV_DI_AI_RESPONSE['ZV_ST_RESPONSE_MSG'],
    }

   # Separate the table and Executive Summary if applicable.
    if "### Executive Summary" in ZV_DI_JS_RESPONSE['response']:
        ZV_ST_TABLE, ZV_ST_SUMMARY = ZV_DI_JS_RESPONSE['response'].split("### Executive Summary", 1)
    else:
        ZV_ST_TABLE, ZV_ST_SUMMARY = ZV_DI_JS_RESPONSE['response'], ""

    # Print header with better formatting
    print("\n" + "=" * 80)
    print(f"  {ZV_DI_JS_RESPONSE['chart_title'].upper()}")
    print("=" * 80 + "\n")

    # Print the table content with proper spacing
    ZV_LI_TABLE_LINES = ZV_ST_TABLE.strip().split("\n")
    for ZV_IN_INDEX, ZV_ST_LINE in enumerate(ZV_LI_TABLE_LINES):
        # Add extra spacing after headers
        if ZV_ST_LINE.startswith("###"):
            if ZV_IN_INDEX > 0:
                print()
```


07
Call the chatbot

# Call the chatbot – just press play!

This script just calls everything else.

Notice at the bottom we have __name__

It means we can run it manually or call it from another program.

```
202         # --------------------
203         # Call the chatbot
204         # --------------------
205
206
207         # 1/Result coms in the form of a JSON
208         ZV_JS_RESPONSE = FC_CHATBOT()
209
210         # 2/ Print JSON
211         # !!!? How to print to file/screen
212
213     if __name__ == '__main__':
214         FC_CHATGPT_API()
215
```

# Call the chatbot – result!

Our result!

TERMINAL

```
================================================================================
  CHART 1/ TOTAL VALUES PER MONTH
================================================================================

### Observations
--------------------------------------------------------------------------------
The total values per month show a generally increasing trend, with February marking the highest value of $150,000, followed by May at $160,000. Howe
ver, March and April indicate a slight decline in values compared to February.


### Risk Assessment
--------------------------------------------------------------------------------
| Issue                 | Priority | Observation                    | Risk                           | Recommendation                      |
|-----------------------|----------|--------------------------------|--------------------------------|-------------------------------------|
| Month-to-Month Decline| Medium   | March and April saw a decline  | Potential sales opportunity loss | Investigate reasons for decline     |
| Seasonal Variability  | Medium   | Fluctuations in monthly values | Inaccurate forecasting         | Perform seasonality analysis        |

--------------------------------------------------------------------------------
  EXECUTIVE SUMMARY
--------------------------------------------------------------------------------
The analysis of the total values per month reveals a strong upward trend overall, with a notable peak in February and May. However, the decline obse
rved in March and April raises concerns about potential sales opportunities being missed. Additionally, the fluctuation in monthly values may indica
te issues with forecasting accuracy. It is recommended to investigate the causes of the decline and to conduct a seasonality analysis to improve fut
ure predictions and strategy alignment.

================================================================================
```

# Questions?