# Six ideas for turning AI into an analytical operating system

From writing a better prompt to running a system that compounds what it learns.

01

### Machines now speak human

Natural language becomes the interface between analysts and code.

02

### Prompting is really specification

A good prompt reads like a brief, not a question — the CLEAR framework.

03

### Anything repeated becomes a skill

Once a workflow works, save it. Don't reinvent it every time.

04

### Memory turns prompts into a learning system

Standards and lessons compound instead of resetting to zero.

05

### Continuous agents run the workflow for you

Bounded, persistent jobs — not open-ended autonomy.

06

### Multimodal RAG gives the model an institutional memory

Grounded in your own text, tables and charts, not just training data.

# 1. Machines now speak human

AI moves market analysis from using software to building your own analytical system.

1940s-1950s
### Machine code

```
01001010
01001111
01000001
...
```

Speak directly to the hardware.

→

1950s-1970s
### Assembly

```
MOV A, B
ADD A, 1
JMP LOOP
...
```

A thinner, more human syntax.

→

1980s-2000s
### Object-oriented

```
class Portfolio:
def
rebalance(self):
...
...
```

Structure for growing complexity.

→

2000s-2020s
### Python

```
import pandas as pd
df =
pd.read_csv(...)
df.groupby(...)
...
```

Productivity for the many, not the few.

→

2020s +
### Natural language

*"Clean this data, merge it with wages, and test these three hypotheses."*

Just tell it what you want.

*For decades, humans learned to speak like machines. Now machines speak like us.*

## Four implications

01 **The cost of programming collapses**
Natural language lowers the barrier from years of training to a simple prompt.

02 **Coding moves to the edge of the organisation**
Analysts can build tools directly, without waiting on a developer.

03 **Analysts work directly with data, automation and models**
From question to code to result, in one conversational workflow.

04 **Move beyond Excel into Python, big data and ML**
Natural language makes far more powerful tools accessible to many more people.

## A simple example

From a question, all the way down to the metal.

1 **Prompt** (natural language)

*"Create a chart of global central bank balance sheets and give me the key takeaways."*

2 **Generated code** (Python)

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('cb_balance_sheets.csv')
df.plot(x='year', y=['Fed','ECB','BoJ','BoE'])
plt.legend(); plt.show()
```

3 **Compiles down to** (assembly — simplified)

```
MOV  EAX, csv_ptr
CALL _read_csv
CALL _groupby
CALL _plot
RET
```

4 **Ultimately runs as** (machine code)

```
01001010
01001111
01000001
...
```

**Key takeaway**

- The interface moved up a level. (Python really compiles to bytecode, not assembly — simplified here for the visual.)

# 2. Prompting is really specification

A good prompt reads like a brief to an analyst — built with a simple framework: CLEAR.

C · CONTEXT

### Give the background

```
Who it's for.
Why you need it.
Any constraints.
```

Background, not just the ask.

→

L · LENGTH & FORMAT

### How long, what shape

```
150 words.
Table, 4 columns:
Asset | Dir. | Why
```

Bullets, a table, JSON — the shape you want.

→

E · EXAMPLES

### Show good (and bad)

```
GOOD: tight, one view
per bullet, cited.
BAD: hedges, no view.
```

Often teaches more than instructions alone.

→

A · REASONING

### Ask it to think first

```
Think step-by-step,
then give the final
answer only.
```

For anything genuinely complex.

→

R · ROLE

### Assign a persona

*"As a senior macro strategist..."*

If it changes how the answer should sound.

*The workflow is not prompt → answer. It is prompt → output → evaluate → refine → retest.*

## Four implications

**01** **CLEAR forces you to specify the job properly**
Context, Length & format, Examples, Ask for reasoning, Role — not "summarise this."

**02** **Evaluation is the real skill, not the question**
Score outputs against your own ground truth, then refine the prompt.

**03** **The loop is iterative by design**
You keep retesting until it reliably meets your analytical standard.

**04** **This is much closer to training a junior analyst**
Than to asking a search engine a one-off question.

## A worked example

A Fed-decision prompt, structured with CLEAR.

1 **Prompt** (CLEAR prompt)

> *"As a senior macro strategist, synthesize what's new in today's Fed meeting for professional investors. Think through the reaction function, then return 3 bullets plus a table for 2Y, 10Y, USD and S&P."*

2 **First pass vs. your standard** (evaluation)

You summarise the same 3 articles yourself. Compare: too verbose? Missed the reaction function? Confused fact with interpretation?

3 **Result** (implications table)

| Asset | Direction | Why |
|---|---|---|
| 2Y yield | ↑ | Cuts pushed out |
| 10Y yield | → | Term premium offsets |
| USD | ↑ | Narrower rate-cut gap |
| S&P 500 | ↓ | Higher-for-longer repricing |

# 3. Anything repeated becomes a skill

Once a workflow works, you shouldn't have to reinvent it every single time.

SPEECHES
**Central banks**

```
Track hawkish → dovish
language shifts across
successive speeches.
```

Same lens, applied every time.

→

EARNINGS
**Earnings calls**

```
Extract catalysts and
changes in forward
guidance automatically.
```

A repeatable extraction task.

→

DATA
**Economic releases**

```
Compare against
expectations, revisions
and trend each time.
```

The comparison never changes.

→

WRITING
**Your house style**

*Sentence length, structure, chart style, signal vs. noise — learned once.*

Fed by hundreds of past notes.

*Anything you ask the model to do more than once deserves to become a skill.*

## Four implications

01 **Writing becomes a durable skill**
Feed it your back-catalogue once; style is inferred, not re-explained.

02 **Visualisation becomes a durable skill**
Tufte-style density and annotation rules apply to every chart, by default.

03 **Any recurring analytical task qualifies**
Speeches, earnings, positioning, morning notes, code review.

04 **You end up with a library of specialists**
Not one generalist assistant re-taught from scratch each time.

## Before and after a skill

The same prompt, with your standards already built in.

1 **Prompt** (natural language)

*"Write today's morning note on the Fed decision."*

2 **Skill applied** (persistent rules)

`tight language` `chart: Tufte density` `fact vs. interpretation` `no unnecessary words`

3 **Result** (on-brand, first time)

GENERIC PROMPT

*"The Federal Reserve today decided to maintain interest rates, citing various economic factors…"*

WITH YOUR SKILL

*"The Fed held rates. What changed: the reaction function, not the decision itself."*

# 4. Memory turns prompts into a learning system

Without memory, every interaction starts from zero. With it, standards compound.

STEP 1
### Work

```
Run a real analytical
task with the model.
```

The starting point of every cycle.

→

STEP 2
### Feedback

```
Compare the output
against your own standard.
```

Where the gap becomes visible.

→

STEP 3
### Learning

```
What mistakes repeated?
What instruction helped?
```

The weekly review question.

→

STEP 4
### Memory

- *use tight language*
- *facts vs. interpretation*
- *ask what changes the view*

Lessons carried into the next prompt.

*Without memory, every interaction starts from zero.*

## Four implications

01 **Memory should hold standards, not just facts**
Tight language, fact vs. interpretation, your Python and charting conventions.

02 **Run a weekly self-evaluation**
What mistakes repeated? What instruction actually improved results?

03 **Turn lessons into standing instructions**
Nudges, not hard rules — the loop compounds: work → feedback → learning → memory → better work.

04 **The system adapts to you**
Not the other way round — it gradually learns how you analyse markets.

## The weekly review

How a mistake becomes guidance for next time.

1 **What happened** (this week)

18 prompts run across speeches, earnings and positioning notes.

2 **What we learned** (self-evaluation)

The model kept blending facts with interpretation in FOMC summaries — flagged three times.

3 **What got saved** (standing instruction)

```
aim to label: FACT vs. INTERPRETATION
```

```
ask: what would change this conclusion?
```

```
favour existing Python + charting
conventions
```

# 5. Continuous agents run the workflow for you

Bounded, persistent jobs with clear inputs, outputs and evaluation — not open-ended autonomy.

06:00
### Observe

```
Scan overnight speeches,
IMF/BIS research, news.
```

No one has to type the prompt.

→

06:05
### Retrieve & compare

```
Pull yesterday's view.
Diff against today's.
```

What actually changed.

→

06:10
### Update & flag

```
Revise relevant themes.
Flag contradictions.
```

Where the evidence disagrees with you.

→

06:15
### Report & note

*Publish the brief.*
*Note the lesson for next time.*

Bounded, persistent, evaluated.

*I don't think of this as letting an AI run autonomously forever — it's a bounded, persistent job.*

## Four implications

01 **Give it a standing job, not a single prompt**
Bounded and persistent, with clear inputs, outputs and evaluation.

02 **Markets work is inherently repetitive**
What changed, why, does it matter, does it alter the view.

03 **Guardrails matter more than autonomy**
Scheduled and bounded — far less exposure to open-ended risk, though never zero.

04 **The loop runs on a schedule**
Observe, retrieve, analyse, compare, update, report, note — daily.

## A standing job

The morning brief, running itself.

1 **Trigger** (standing job, not a prompt)

Every weekday morning, 06:00 — on a schedule, not typed by hand.

2 **Runs** (the loop)

```
observe → retrieve → analyse
compare → update → report
note it for next time
```

3 **Output** (this morning's brief)

*"Contradiction flagged: BoJ language softened overnight vs. our hawkish base case."*

# 6. Multimodal RAG gives the model an institutional memory

The model stops relying only on training data — and starts operating over your own text, tables and charts.

INGEST
### Research library

`IMF, Fed, BIS papers, sell-side, your own notes.`

Your proprietary universe, not the open web.

→

STRUCTURE
### Parse & chunk

`Break into passages. Attach metadata, tables, charts.`

Text, tables and charts all become retrievable.

→

RETRIEVE
### Relevant material

`Pull the passages, tables or charts that actually answer the question.`

Not just what training data recalls.

→

STRUCTURE OUT
### Text becomes data

*Hawkish/dovish shifts.*
*Theme frequency.*
*AI-capex sentiment.*

Thousands of documents, one dataset.

*Once you structure the text properly, thousands of documents become a dataset.*

## Four implications

01 **Don't let the model rely only on training data**
Build a library of IMF, Fed, BIS and your own historical notes.

02 **Retrieval isn't limited to text**
Charts and tables are retrieved directly, not just captioned as text — this is multimodal RAG.

03 **Text becomes data**
Hawkish/dovish shifts, theme frequency, sentiment on AI capex.

04 **AI moves from helping you read research to being research infrastructure**
A daily process that asks what's new, and what's changed.

## A worked example

Grounding a historical-parallel question in your own library.

1 **Prompt** (natural language)

> *"What historical episodes resemble today's mix of loose fiscal policy, tight monetary policy and rising term premium?"*

2 **Retrieved** (not recalled)

`2 BIS papers` `1 IMF working paper` `3 Fed speeches`

3 **Result** (grounded synthesis)

Closest parallel: US, late 1960s — fiscal expansion met a tightening Fed, and term premium rose as the market priced persistence.

# Six things to try this week

Same six ideas. Here's what to actually do with each one.

01

### Machines now speak human

→ Pick one Excel task and ask a model to do it in plain English instead.

02

### Prompting is really specification

→ Rewrite your next prompt using CLEAR: Context, Length & format, Examples, Ask for reasoning, Role.

03

### Anything repeated becomes a skill

→ Turn your most-repeated prompt into a saved skill or template.

04

### Memory turns prompts into a learning system

→ Run one weekly review: what mistake repeated, what fixed it, save it.

05

### Continuous agents run the workflow for you

→ Set up one standing job — even a simple daily scheduled prompt.

06

### Multimodal RAG gives the model an institutional memory

→ Start a small research library and ask one grounded question against it.

*AI stops being a chatbot and starts becoming an analytical operating system.*