JBSWIKI AI LEARNING SERIES · SESSION 01 · WEEK 1

Introduction to Generative AI and Large Language Models

A correct mental model of what an LLM is, how it writes, and where it fails

Vivek Janakiraman
JBSWiki · YouTube series first, instructor-led later

Scenario data: BlueFern Travel (fictional) · Microsoft Foundry (formerly Azure AI Foundry)

WHAT WE BUILD OVER 17 SESSIONS

- **S01 · Concepts and vocabulary — today**
- **S02–S04 · Toolkit, Foundry Tools, Foundry setup**
- **S05–S07 · Prompts, embeddings, RAG**
- **S08–S11 · Agents, MCP, A2A**
- **S12–S16 · Security, evaluation, cost**
- **S17 · Capstone: the Fern assistant**

SESSION PURPOSE

The right mental model before touching Azure

By the end you can explain an LLM to a colleague — correctly, and without the hype.

1. **Understand**
   what AI, machine learning, deep learning, generative AI and LLMs are
2. **See**
   how a model turns tokens into an answer, and why it sometimes invents facts
3. **Judge**
   which BlueFern Travel tasks suit generative AI and which do not

THE BLUEFERN STORY

BlueFern Travel answers the same questions all day: baggage, refunds, changes, insurance.

Managers want an assistant, Fern, that answers from policy and escalates to humans.

Before we build Fern we must know what the engine inside can and cannot do.

Every example in this series uses fictional BlueFern data only.

LEARNING OBJECTIVES

Five things you will be able to do afterwards

1 **Define**

AI, machine learning, deep learning, generative AI and LLM in a nested diagram

2 **Explain**

tokens, next-token prediction, context window, reasoning effort and verbosity

3 **Describe**

why models hallucinate and why they are not a source of truth

4 **List**

five good and five poor business use cases for generative AI

5 **Recognise**

the Microsoft Foundry model catalog and model versus deployment

How we will check

A five-question knowledge check near the end, plus a take-home worksheet.

PREREQUISITE KNOWLEDGE

You need curiosity, a browser and nothing else

Required

- Comfortable using a web browser
- Have used any chat assistant once
- Basic English reading

Helpful, not required

- Any programming exposure
- Heard of Azure or cloud services
- Worked with a support or policy team

Not needed today

- An Azure subscription
- Python installed (Session 2 shows how)
- Any mathematics or statistics

If you have asked a chatbot a question and wondered how it answered, you are ready.

Presenter has a Foundry project ready for the demo; Session 4 shows you how to create one.

AGENDA

72 minutes: 40% theory, 60% demonstration



Introduction	Purpose, objectives, prerequisites, scope	**4 min**
Theory	Concepts, tokens, hallucination, catalog, use cases	**30 min**
Demonstration	Tokeniser, playground, reasoning effort, context window	**30 min**
Knowledge check	Five questions and answers	**4 min**
Summary	Recap, assignment, references, next session	**4 min**

SCOPE

What is in, out, assumed and postponed

In scope

- Core concepts: AI, ML, deep learning, generative AI, LLM
- Tokens, context window, reasoning effort, verbosity
- Hallucination, capabilities and limits
- A first look at the Foundry chat playground

Out of scope

- Creating Azure resources (Session 4)
- Prompt techniques (Session 5)
- Embeddings and vector search (Session 6)
- Fine-tuning (Session 15)

Assumed

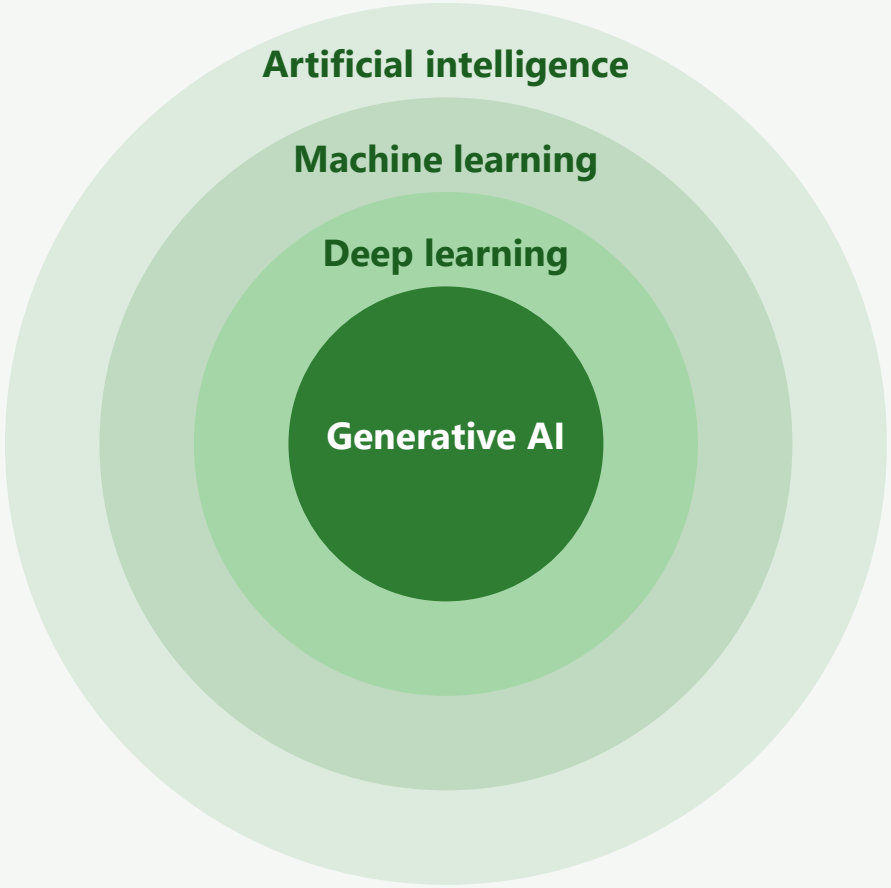
- Nothing. This is the first session.
- No account, no tools, no code

Postponed

- Everything hands-on with Azure
- Your own Foundry project (Session 4)
- Running code against a model (Session 4 and 5)

CORE CONCEPT 1

AI, ML, deep learning and generative AI nest



Artificial intelligence
software that performs tasks that normally need human judgement

Machine learning
the software learns patterns from data instead of hand-written rules

Deep learning
machine learning with many-layered neural networks

Generative AI
models that create new content: text, images, audio, code

Large language model (LLM)
a generative model trained on text; multimodal ones also read images or audio

LLM
a generative model for text

Discriminative
Classifies what exists: is this email spam?

Generative
Creates something new: write the reply email.

CORE CONCEPT 2

How a language model is built and answers

1 Training

Read huge amounts of text and learn which words tend to follow which

2 Transformer

A network design whose attention step decides which earlier words matter most

3 Prediction

Given your prompt, score every possible next token and pick one

4 Loop

Append the token, predict again, stop at an end token or the limit

What next-token prediction looks like



Green: your prompt. Teal: tokens the model has generated so far. Amber: the token being chosen now.

No understanding is required for this to work; the pattern of language is enough. Keep that in mind for hallucination.

CORE CONCEPT 3

Tokens: the unit of input, output and cost

A token is a piece of a word

Blue	Fern	Travel	refund	policy

"BlueFern Travel refund policy" is 5 tokens, not 4 words (verify with count_tokens.py)

~4
English characters per token

750
words in about 1,000 tokens

more
tokens for Hindi or Tamil than English

Rule of thumb only. Numbers, code and non-Latin scripts use more tokens per character, which means higher cost.

Context window

The maximum number of tokens one request can hold, counting input, reasoning and output together.



One shared budget; whatever input uses, output cannot.

- Long policy pasted in means less room to answer
- Reasoning tokens count too, and they are billed as output
- Exceed it and the request fails or is cut off

CORE CONCEPT 4

Three dials you control on a reasoning model

`reasoning effort`

minimal · low · medium · high

How much thinking before the answer. More effort means deeper answers, more time and more cost.

Like asking a colleague for a quick guess or a considered opinion.

`verbosity`

low · medium · high

How long the visible answer is. Low gives the essentials, high gives detail and examples.

Like asking for a headline or the full report.

`max_output_tokens`

a number, for example 400

The budget for reasoning plus visible output in one response. Hit it and the answer is incomplete.

Like a word limit on an exam answer.

LEGACY temperature and top_p controlled randomness on older models (GPT-4.1, GPT-4o). GPT-5 family reasoning models do not accept them.

CORE CONCEPT 5

Hallucination: fluent, confident and wrong

Asked with no company data

“What is BlueFern Travel’s baggage allowance?”
“BlueFern allows 23 kg checked and 7 kg cabin baggage on all fares.”

Sounds right. The model has never seen a BlueFern policy.

Why it happens

- Pattern completion: a plausible sentence is the goal, not truth
- No grounding: it was never given your documents
- Training data has a cut-off date and contains errors

Other limits to remember

Knowledge cut-off
nothing after the training date without tools

No real-time data
weather, prices, seat availability need a tool

Arithmetic slips
long calculations can be wrong; use code

Bias
training text carries human biases

Non-determinism
the same prompt can give different answers

How we fix it later
Grounding with RAG (S07) · guardrails (S13) · evaluation (S14)

TERMINOLOGY

Twelve words you will hear in every session

AI

software doing tasks that need human-like judgement

ML

learning patterns from data, not rules

LLM

large language model; generates text token by token

Token

a piece of a word; the unit of cost and limits

Context window

max tokens per request: input + reasoning + output

Prompt

everything you send: instructions plus the question

Response

what the model returns (older name: completion)

Reasoning effort

how much the model thinks before answering

Hallucination

fluent but false output

Grounding

giving the model your facts to answer from

Multimodal

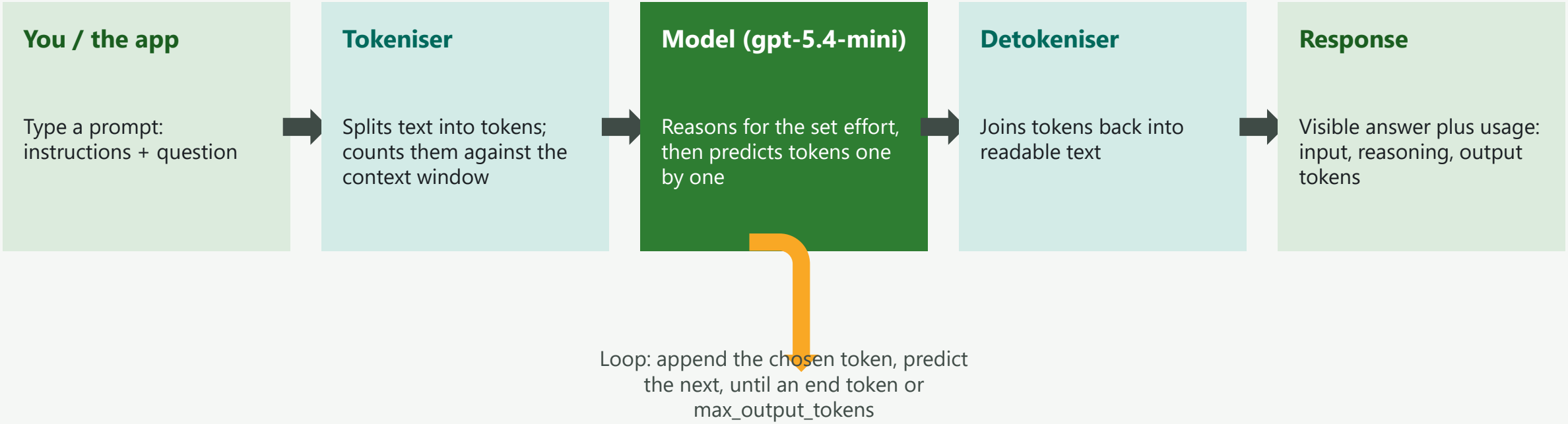
takes text plus images or audio as input

Inference

running a trained model to get an answer

HOW A REQUEST FLOWS

From your words to the model’s answer and back



Everything is text in, text out
Images and audio are converted into tokens too

The model keeps no memory
Each request must carry the whole conversation

Usage is returned every time
input_tokens, reasoning_tokens, output_tokens drive cost

MICROSOFT FOUNDRY MODEL CATALOG

Model is the recipe; deployment is your kitchen

Discover › Models (Nov 2026)

Azure OpenAI in Foundry Models
gpt-5.4-mini (our default) · gpt-5.6-sol / terra / luna (newest) · gpt-5.1 · text-embedding-3-small · image and audio models

Partner and community
Meta Llama · Mistral · DeepSeek · Cohere · xAI Grok · Microsoft Phi and MAI

History
GPT-4.1 family and gpt-4o retire 2027 · o-series reasoning models retire 2026-11-19

LEGACY Always check the catalog on the day; names and dates change.

Model: gpt-5.4-mini
Published by the provider; the same for everyone



Deployment: gpt-5.4-mini
YOUR named instance inside your Foundry project
Type Global Standard · tokens-per-minute quota · a guardrail

Your code calls the deployment name, never the model directly.
We name deployments after the model to reduce confusion.

REAL-WORLD USE CASES AT BLUEFERN

Where generative AI earns its place

Summarise	Draft	Classify	Extract	Translate	Code	Reason	Converse

The capability map: eight verbs a language model is good at, with multimodal input where needed.

1

Support drafting

Draft a reply to Asha Rao about a changed flight; an agent reviews and sends

Draft · Converse

2

Policy summarisation

Turn the 6-page refund policy into a one-screen summary for new staff

Summarise

3

Review sentiment

Classify 2,000 hotel reviews as positive, neutral or negative

Classify

4

Itinerary suggestions

Propose a 3-day Kochi plan from a traveller's preferences

Reason · Draft

5

Code assistance

Explain the bookings.json schema and write a filter script

Code

Highlighted: the first thing Fern will do, from Session 5 onward.

APPROPRIATE VS INAPPROPRIATE USE

Good fits suggest; bad fits decide alone

Appropriate

✓Drafting replies a human reviews before sending

✓Summarising long policies for staff

✓Classifying and routing incoming messages

✓Suggesting options the traveller chooses from

✓Explaining code and documentation to developers

Inappropriate

✗Final legal or medical decisions

✗Giving customers facts no one has verified

✗Processing personal data without consent

✗Approving refunds with no human sign-off

✗Anything where a confident wrong answer causes harm

Test: if the model is wrong, who notices before a customer is affected? If nobody, it is a bad fit.

DEMO ARCHITECTURE

Seeing an LLM think: two paths, nothing created

Path A · Your laptop

count_tokens.py
Python 3.10+

→

tiktoken
o200k_base

→

Token counts
printed in the terminal

Path B · Foundry (prepared beforehand)

Chat playground
Build › Playgrounds

→

Deployment
gpt-5.4-mini

→

Model catalog
Discover › Models

What we will observe

1. **Tokens**
   non-English costs more
2. **Hallucination**
   a confident invented policy
3. **Effort and verbosity**
   minimal vs high, low vs high
4. **Context window**
   long input, small budget
5. **System message**
   Fern refuses off-topic

Foundry resource, project and deployment already exist (East US 2). Session 4 shows how to create them. Region check: pick a region where gpt-5.4-mini is listed.

REGION CHECK

DEMO 1 · COUNT TOKENS LOCALLY

Four chars per token holds only for English

```python
# Session01_Demo_Code/count_tokens.py
import tiktoken
enc = tiktoken.get_encoding("o200k_base")

text = "BlueFern Travel refund policy"
ids = enc.encode(text)
print(len(ids), [enc.decode([i]) for i in ids])
# -> 5 ['Blue', 'Fern', ' Travel', ' refund', ' policy']
```

Run: `python count_tokens.py`

- Runs offline; no Azure, no key, no cost
- Same encoding family used by current OpenAI models (verify)
- Pass your own text as an argument to experiment

Expected output (illustrative counts)

Sample	Chars	Tokens
Short phrase	29	5
Long sentence	218	~45
Hindi sentence	40	~20
Tamil sentence	33	~25
Numbers	61	~30

What to notice

Hindi and Tamil need roughly one token per two characters. Same meaning, two to three times the cost of English.

Same question, different effort and verbosity

Chat playground · no system message, no data

"What is BlueFern Travel's baggage allowance?"
Watch for: a specific, confident answer about a company that does not exist, or an honest "I don't have that information".

Either outcome teaches the lesson: never ship unverified facts.

Experiment: "Explain how an airline sets baggage fees"

Run	Reasoning effort	Verbosity	Observe
1	`minimal`	`low`	fast, short, surface-level
2	`high`	`low`	slower, short but deeper
3	`minimal`	`high`	fast, long, padded
4	`high`	`high`	slowest, longest, most complete

Compare the response time and the output-token count shown in the playground after each run.

Effort buys depth
and costs reasoning tokens plus seconds

Verbosity buys length
not accuracy; long is not better

Pick per task
low effort to classify, high for multi-step

DEMO 3 · LIMITS, INSTRUCTIONS AND THE CATALOG

Push the window, set the persona, browse models

1 Context window

- Paste the full baggage policy, ask one question
- Note the input-token count in the playground
- Set max output tokens to 20 and ask again
- Answer stops mid-sentence: the budget is shared

2 System message

- "You are Fern, BlueFern Travel's assistant. Answer only travel questions."
- Ask: "How do I cook biryani?"
- Expect a polite refusal and a travel offer
- This is Session 5's starting point

3 Model catalog

- Discover › Models › gpt-5.4-mini card
- Read context length, retirement date, regions
- Open gpt-5.6 family and text-embedding-3-small
- Model card vs your deployment: two different things

If the playground truncates instead of erroring, say so and show the token counter. If the portal is unavailable, use the prepared screenshots and the Python demo only.

COMMON MISTAKES

Six habits beginners bring, and the correction

Treating the model as a search engine

It predicts text; it does not look things up. Ground it or verify.

Trusting a confident tone

Fluency is not accuracy. Check every fact that reaches a customer.

Ignoring tokens

Long prompts and chats quietly multiply cost and hit the window.

Expecting memory

Each request starts blank; your app must resend the history.

Using one setting for everything

Low effort for classification, high for multi-step reasoning.

Confusing model and deployment

Code calls your deployment name, with its own quota and region.

SECURITY CONSIDERATIONS

A playground is a cloud service, not a notepad

Data you paste

- Never paste real customer or personal data
- Use the fictional BlueFern set only
- Prompts travel to the service and are logged for abuse monitoring

Who can call the model

- Sign in with Microsoft Entra ID; keys are not used in this series
- Foundry User is the minimum role for learners
- Least privilege: one role per need

What can go wrong

- Prompt injection: text that tells the model to ignore its rules
- Data exfiltration through pasted or retrieved content
- Both are covered properly in Sessions 12 and 13

Rule for today: fictional data in, Entra ID sign-in, no keys in code, no secrets in prompts.

RESPONSIBLE AI CONSIDERATIONS

Three principles, applied from the first prompt

Transparency

Tell people they are talking to an AI, and what it cannot do.

At BlueFern
Fern's first message says it is an AI assistant and offers a human.

Fairness

Training data carries bias; test answers across languages and groups.

At BlueFern
The same refund question in English and Hindi must get the same answer.

Human oversight

A person stays accountable for decisions that affect customers.

At BlueFern
Priya, the support lead, approves refunds; Fern only drafts.

Microsoft publishes six Responsible AI principles; we cover all six, plus guardrails and content safety, in Session 13. Reference: Responsible use of AI overview on Microsoft Learn.

COST CONSIDERATIONS

You pay per token: input, reasoning and output

```
cost = (input_tokens  / 1,000,000) x input_price
     + (output_tokens / 1,000,000) x output_price

# output_tokens includes reasoning tokens
# prices: Azure OpenAI pricing page (never hard-code)
```

- Every playground message is a billed request
- Output tokens cost more than input tokens; reasoning counts as output
- High effort can multiply the bill for the same question
- Keep max_output_tokens sensible; set verbosity low for tests
- Pasting a long policy on every turn is the hidden cost

Where today's demo costs money

count_tokens.py
nothing; runs offline

Playground question
a few hundred tokens

High-effort run
hundreds of reasoning tokens extra

Long policy pasted
thousands of input tokens per turn

Browsing the catalog
nothing

Check current prices on the pricing page before quoting any number.

TROUBLESHOOTING

What you may see today and what it means

Symptom	Likely cause	What to do
ModuleNotFoundError: tiktoken	Package not installed in the active environment	pip install tiktoken in the same terminal; check the venv
Encoding download fails	No internet or proxy blocks the first-run download	Check connectivity; retry; encodings are cached afterwards
Model not in the playground list	No deployment in this project, or wrong region	Deploy gpt-5.4-mini (Session 4); check region availability
Answer cut off mid-sentence	max_output_tokens too small for reasoning plus output	Raise the budget or lower reasoning effort
Context length error	Input plus requested output exceeds the window	Shorten the input; send only the relevant paragraphs
429 Too Many Requests	Tokens-per-minute quota exhausted	Wait, lower the rate, or raise quota under Operate › Quotas
Content filtered	Guardrail blocked the prompt or answer	Rephrase; review the guardrail (Session 13)

Portal labels can change; if a menu name differs, verify on the day and use the search box in the portal.

KNOWLEDGE CHECK

Five questions; answers on the next slide

1. What is a token, and roughly how many English characters make one?
2. What does reasoning effort control, and what does it cost you?
3. Why can an LLM state a false fact with complete confidence?
4. What is the difference between a model and a deployment?
5. Name one task where a prebuilt Foundry Tool beats an LLM.

ANSWERS

Check yourself against these

1	**Token**	A piece of a word, the unit the model reads and bills. About 4 English characters; fewer for Hindi or Tamil.
2	**Reasoning effort**	How much the model thinks before answering. More effort means deeper answers, more reasoning tokens, more time.
3	**Confident falsehoods**	It predicts plausible text from patterns; it has no grounding and no notion of truth unless you supply facts.
4	**Model vs deployment**	The model is the provider's published recipe; the deployment is your named instance with quota, region and guardrail.
5	**Prebuilt tool wins**	Speech-to-text, translation, or reading a passport with Document Intelligence: exact, repeatable, cheaper.

SUMMARY

What to remember from Session 1

1. **Nested family**

   AI ⊃ ML ⊃ deep learning ⊃ generative AI; an LLM generates text token by token

2. **Tokens rule everything**

   input, reasoning and output share one context window and one bill

3. **Three dials**

   reasoning effort for depth, verbosity for length, max_output_tokens for budget

4. **Hallucination is normal**

   fluent and wrong; fix with grounding, guardrails and evaluation later

5. **Model vs deployment**

   the recipe versus your named, quota-bound instance in Foundry

6. **Fit test**

   if the model is wrong, who notices before a customer is affected?

LEARNER ASSIGNMENT

About ninety minutes, no Azure account needed

1. **Install Python 3.10 or later**
   Session 2 walks through it in detail; any recent version works

2. **pip install tiktoken, run count_tokens.py**
   Then count three sentences of your own, one in your mother tongue

3. **Complete use_case_worksheet.md**
   Classify 10 BlueFern requests as suitable or not, with one reason each

4. **Read “What is Microsoft Foundry”**
   Ten minutes on Microsoft Learn; note the three portal sections

5. **Optional: create an Azure account**
   Pay-as-you-go or sponsorship; the free tier may not include AI services

Files: v2/01_Introduction_to_Generative_AI/Session01_Demo_Code/ · Bring your worksheet answers to Session 2.

OFFICIAL REFERENCES

All links are Microsoft Learn or official pages

What is Microsoft Foundry	`learn.microsoft.com/azure/foundry/what-is-foundry`
Foundry Models sold by Azure	`learn.microsoft.com/azure/foundry/foundry-models/concepts/models-sold-directly-by-azure`
Model region availability	`learn.microsoft.com/azure/foundry/foundry-models/concepts/models-sold-directly-by-azure-region-availability`
Quotas, limits and tokens	`learn.microsoft.com/azure/foundry/openai/quotas-limits`
Responses API (reasoning, verbosity)	`learn.microsoft.com/azure/foundry/openai/how-to/responses`
Responsible use of AI overview	`learn.microsoft.com/azure/foundry/responsible-use-of-ai-overview`
Learn: Fundamentals of Generative AI	`learn.microsoft.com/training/modules/fundamentals-generative-ai/`
AI-900 certification	`learn.microsoft.com/credentials/certifications/azure-ai-fundamentals/`

Pricing: azure.microsoft.com/pricing/details/cognitive-services/openai-service/ · Links checked Nov 2026; names change, search Learn if one moves.

NEXT SESSION · SESSION 02 · WEEK 1

Developer Toolkit for AI Projects

Python, VS Code, Azure CLI, Git and the BlueFern data set — everything your laptop needs before we touch Foundry.

- Install and verify Python 3.10+, VS Code, Azure CLI, Git
- Create the bluefern-data folder: policies, bookings, questions
- Call your first REST API from Python
- Set up .env and a virtual environment the right way

Bring to Session 2

Your worksheet answers
Your token-count ratio for your language
A laptop with admin rights to install tools

DEMO RUNBOOK · 1 OF 3

Before you start

Item	Value / check
Foundry resource + project	jbs-ai-foundry-<initials>01 / fern-project, region East US 2 (created as in Session 4)
Deployment	gpt-5.4-mini, Global Standard, default guardrail; visible under Build › Models + deployments
Role	Foundry User on the project (Foundry Project Manager if you must deploy)
Local tools	Python 3.10+, terminal open in Session01_Demo_Code, pip install -r requirements.txt
Files	count_tokens.py · baggage_policy.md (fictional, for the context-window paste)
Browser	Signed in at https://ai.azure.com, project open, Build › Playgrounds › Chat loaded
Quota	Operate › Quotas: tokens-per-minute not exhausted; one test message sent successfully

REGION CHECK gpt-5.4-mini must be listed for your region in Discover › Models; otherwise pick Sweden Central.

DEMO RUNBOOK · 2 OF 3

Run it

1	**Terminal:**	`python count_tokens.py`	read the table; point at Hindi and Tamil rows
2	**Terminal:**	`python count_tokens.py "BlueFern Travel refund policy"`	show the five pieces
3	**Portal:**	`Build › Playgrounds › Chat › Deployment = gpt-5.4-mini`	clear system message
4	**Ask:**	`“What is BlueFern Travel’s baggage allowance?”`	discuss the invented or declined answer
5	**Set:**	`Reasoning effort = minimal, Verbosity = low`	ask “Explain how an airline sets baggage fees”; note time and tokens
6	**Set:**	`Reasoning effort = high, Verbosity = high`	same question; compare time, length, output tokens
7	**Paste:**	`baggage_policy.md, ask about sports equipment`	show the input-token count
8	**Set:**	`Max output tokens = 20, ask again`	answer stops mid-sentence; restore the default
9	**System message:**	`“You are Fern, BlueFern Travel’s assistant. Answer only travel questions.”`	ask “How do I cook biryani?”
10	**Portal:**	`Discover › Models › gpt-5.4-mini, then gpt-5.6 family, text-embedding-3-small`	model card vs deployment

Expected result and fallback

What you should see

- Token table printed; Hindi and Tamil rows show about 2 chars per token
- Baggage question: invented specifics or an honest “I don’t know”
- High effort: slower, more output tokens, deeper answer
- Policy paste: input count in the thousands
- Max output tokens 20: answer truncated mid-sentence
- Fern system message: polite refusal of the cooking question

If something fails

- Portal down: use the prepared screenshots; run the Python demo only
- No gpt-5.4-mini quota: deploy gpt-5.1 or gpt-5.4-nano instead
- Model missing in region: switch to Sweden Central or East US 2
- No truncation error: say so; show the token counter instead
- 429 quota error: wait a minute, then resend one message
- tiktoken install fails: show the token strip on slide 9