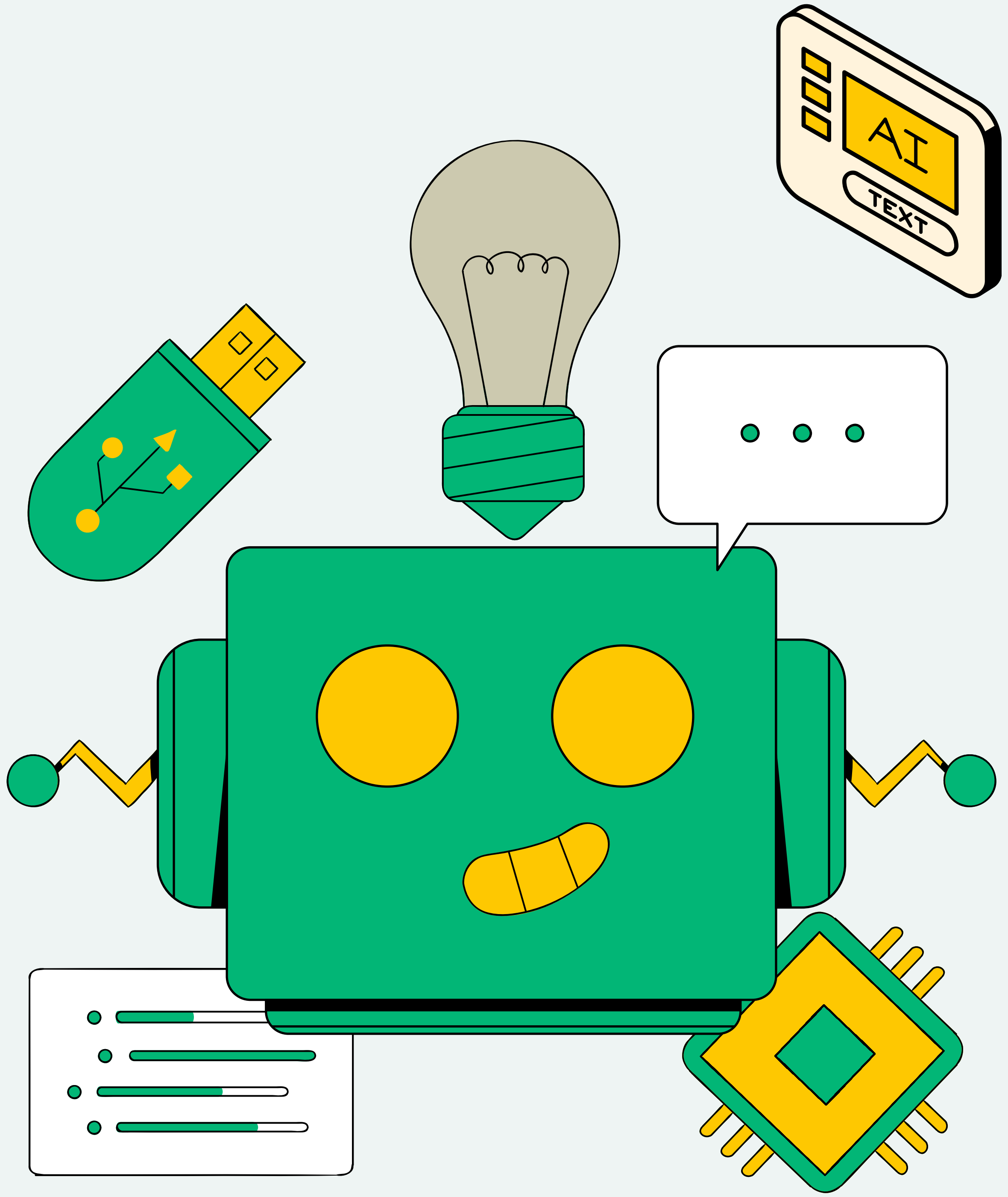
ENTROPIC

ISCB-AFRICA ASBCB CONFERENCE ON BIOINFORMATICS

14–17 April 2025 Cape Town, South Africa

IP5: BUILDING AGENTIC WORKFLOWS IN BIOINFORMATICS



CREATED BY: DIONIZIJE FA

Funded by the European Union
NextGenerationEU

PRESENTERS

DIONIZIJE FA

MATEO ČUPIĆ

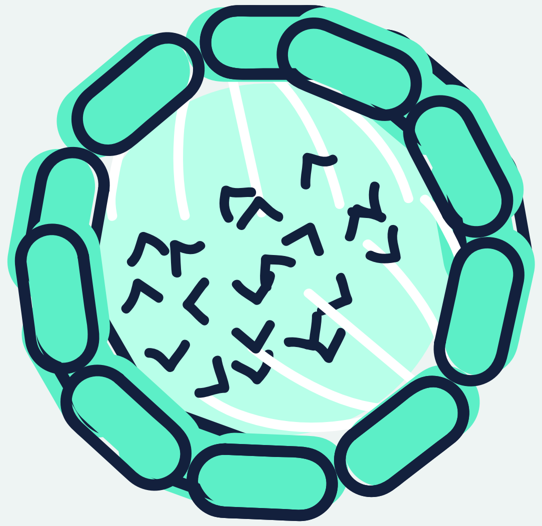
BRUNO PANDŽA

PRESENTATION OUTLINE

1. Bioinformatics introduction
2. AI Overview
3. Large Language Models
4. LLM Agents and Agentic Workflows
5. **Bioinformatics agents – Hands on**
6. Advanced techniques

BIOINFORMATICS

How do we investigate biology on computers and is there something we can automate?



FROM EXPERIMENT TO A FASTA FILE

DNA EXTRACTION

- Isolate DNA from cells using chemical and enzymatic methods.
- Purify the genetic material from proteins, lipids, and other contaminants.

SEQUENCING

- Utilize sequencing-by-synthesis to convert DNA into signals.
- Incorporate fluorescently labeled or modified nucleotides that emit distinct signals.

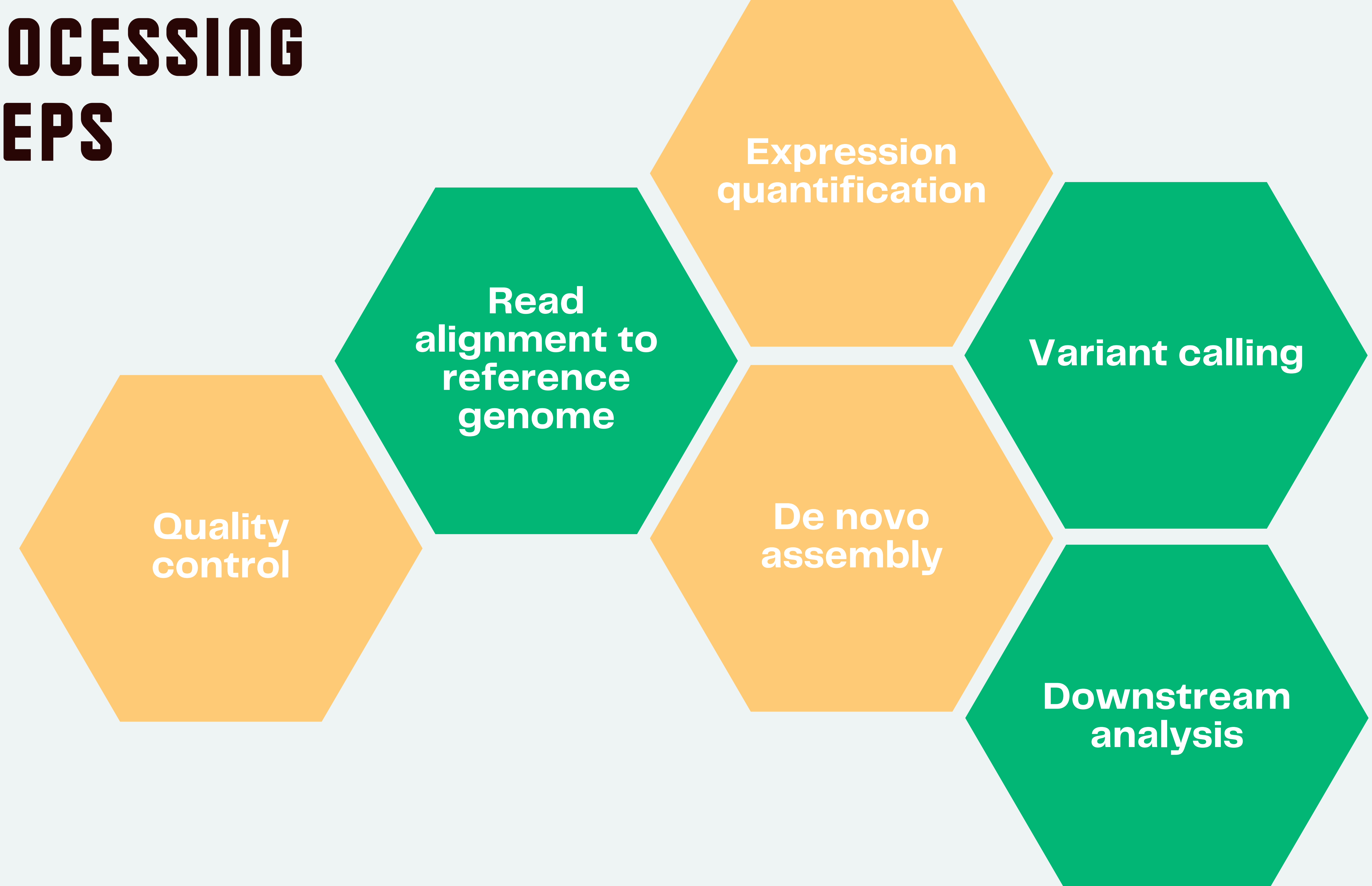
BASE CALLING

- Detect emitted signals as nucleotides are added during synthesis.
- Algorithms translate these signals into A, T, G, C

ANALYSIS

- Process raw data into digital sequences.
- Store high-quality sequences in FASTA format for downstream bioinformatics analyses.

PROCESSING STEPS



Quality control

Read alignment to reference genome

Expression quantification

De novo assembly

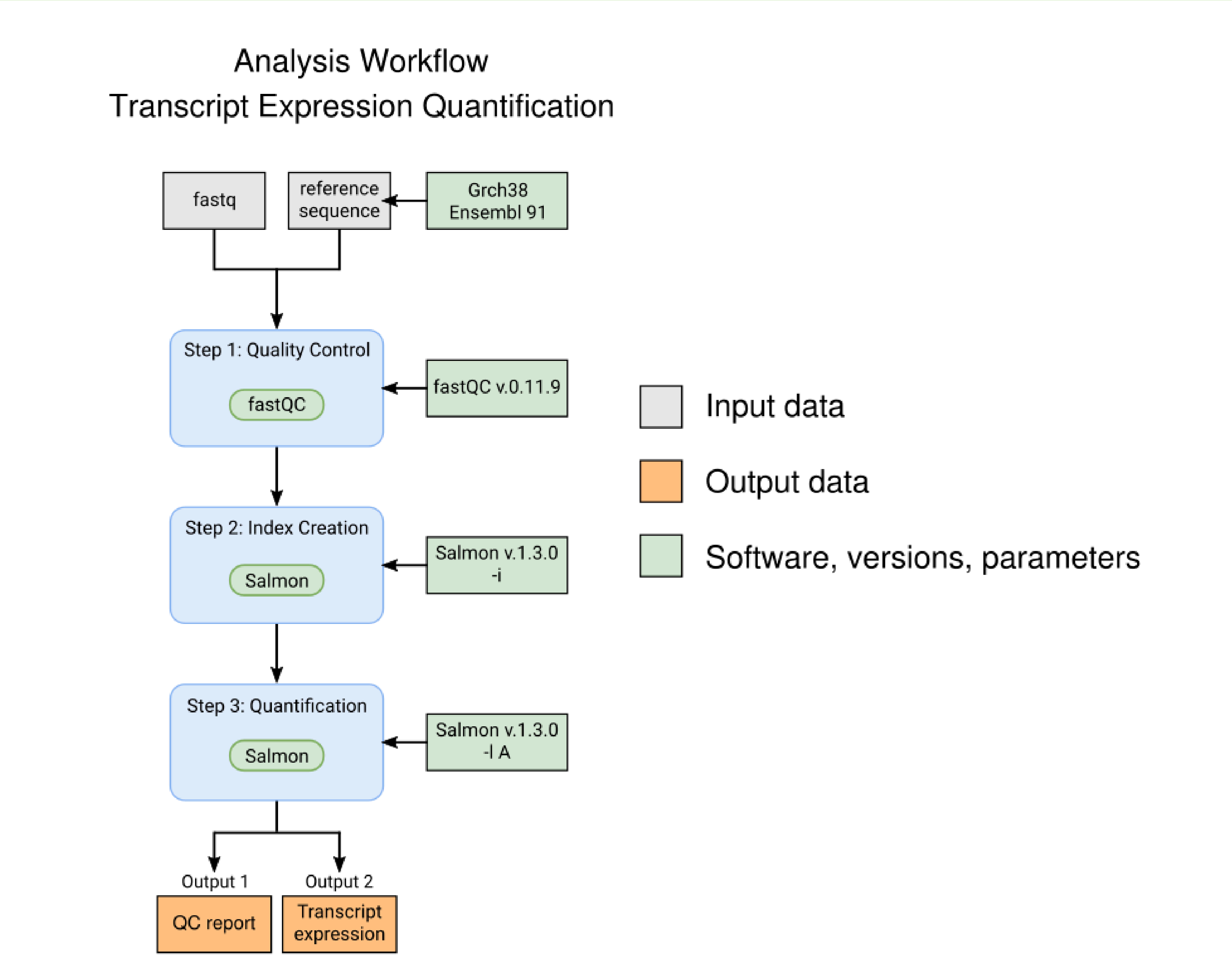
Variant calling

Downstream analysis

BOTTLENECKS IN BIOINFORMATICS

- Manual hand-offs between tools
- Pipeline configuration and debugging
- Resource and scheduling issues
- Expertise bottleneck
- Scaling and updating workflows
- Focus on software rather than interpretation

TODAY'S TASK TO AUTOMATE

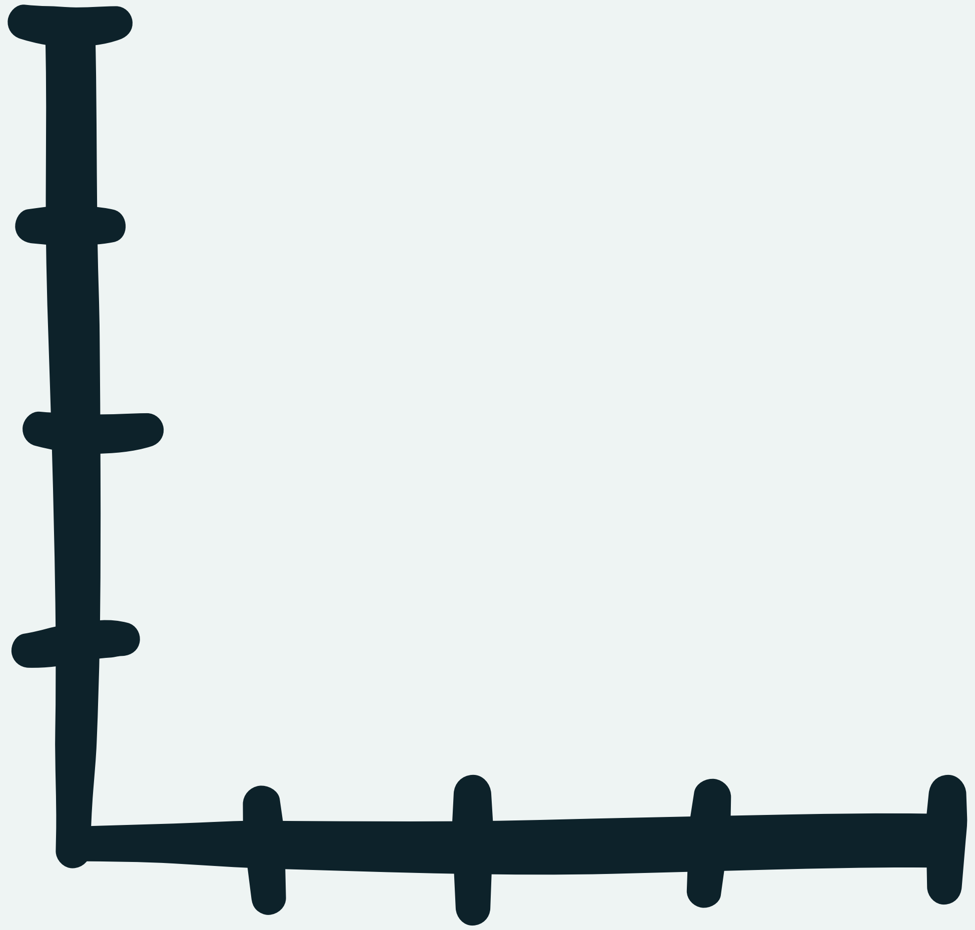


Analysis Workflow
Transcript Expression Quantification

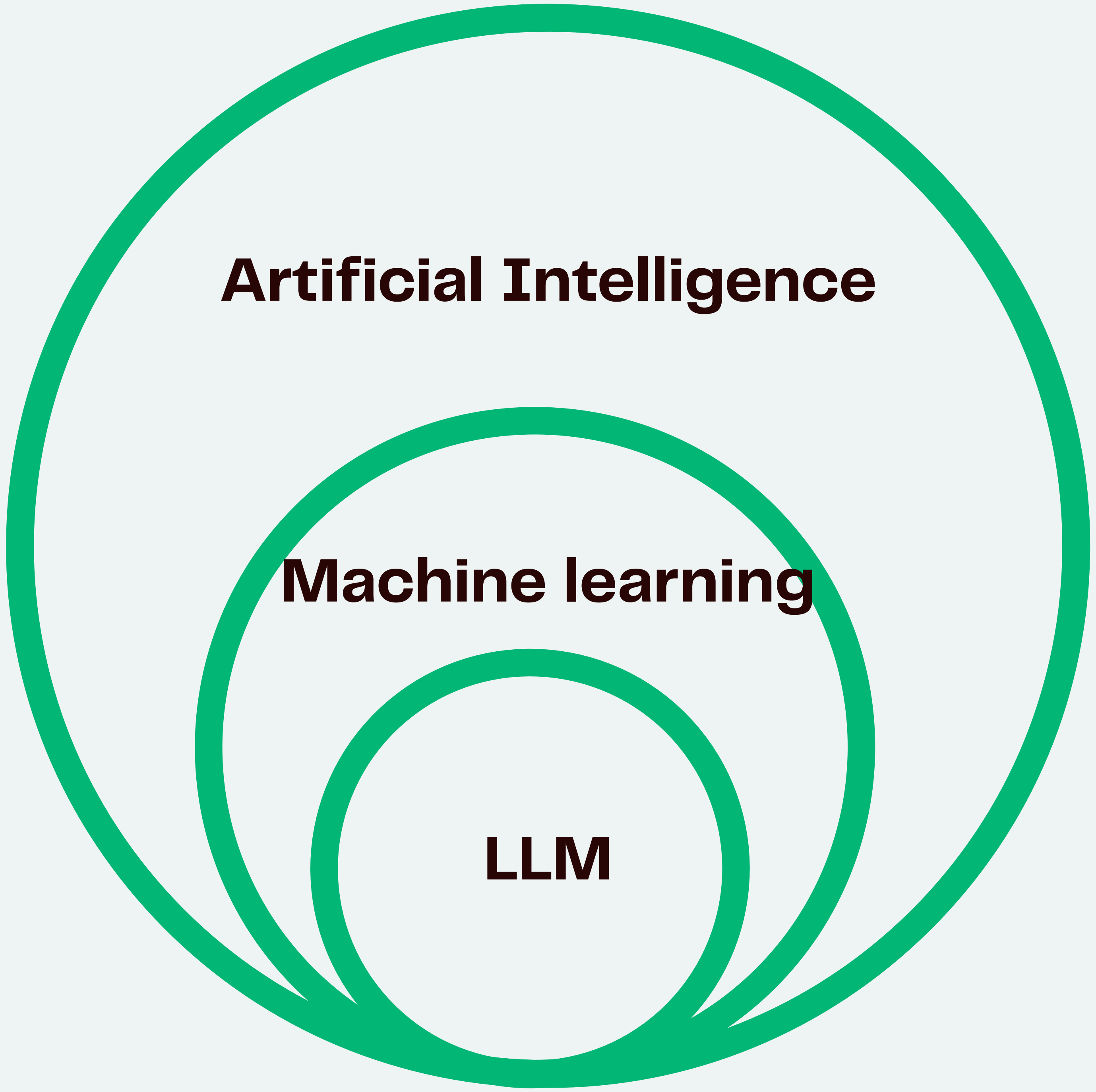
https://github.com/GoekeLab/bioinformatics-workflows

ARTIFICIAL INTELLIGENCE

Can AI help us in automating bioinformatics?



WHAT IS AI?



AI IN BIOLOGY

1700

Linear regression done by Isaac Newton

1960

HMMs for sequence modeling

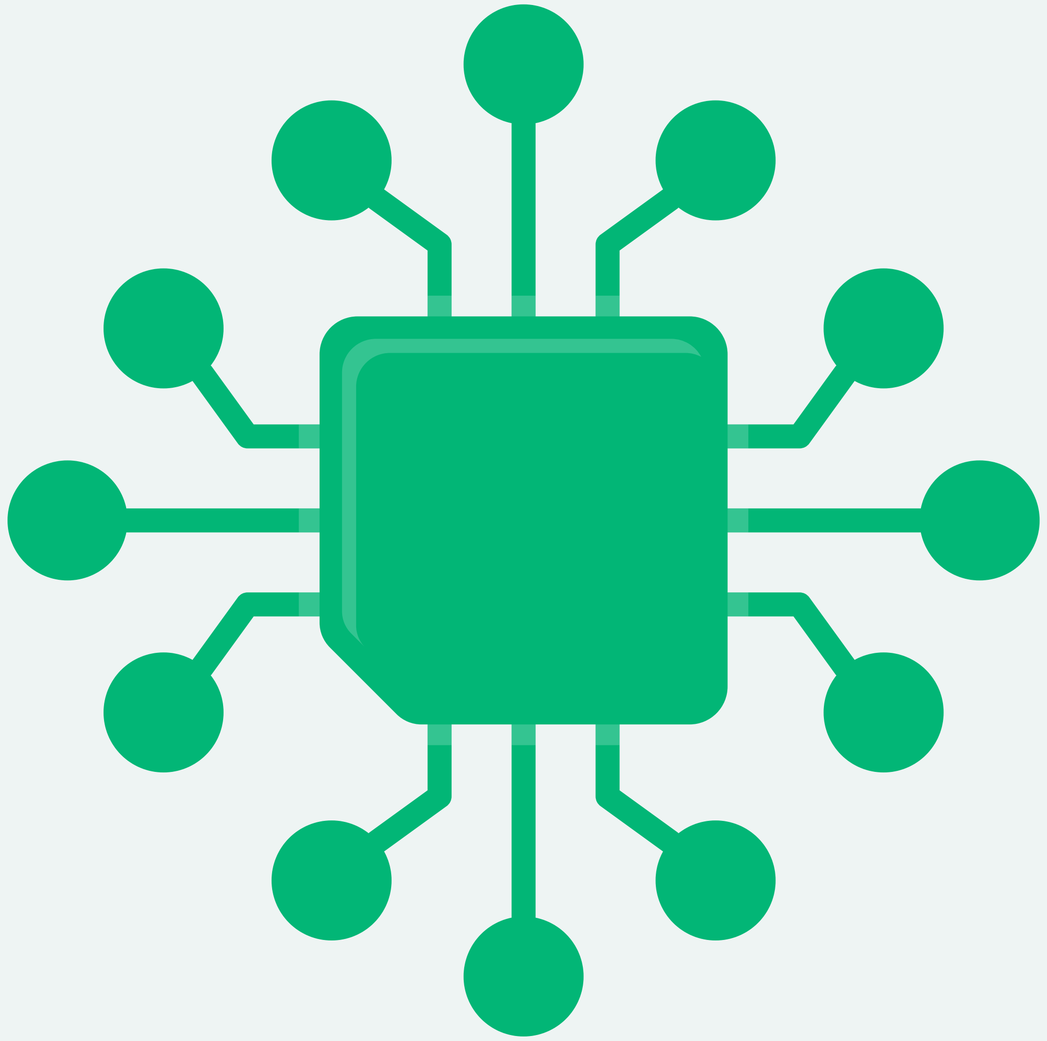
1990

Simple neural networks for secondary structure prediction

2021

Alpha Fold 2 – Structure prediction better than X-ray

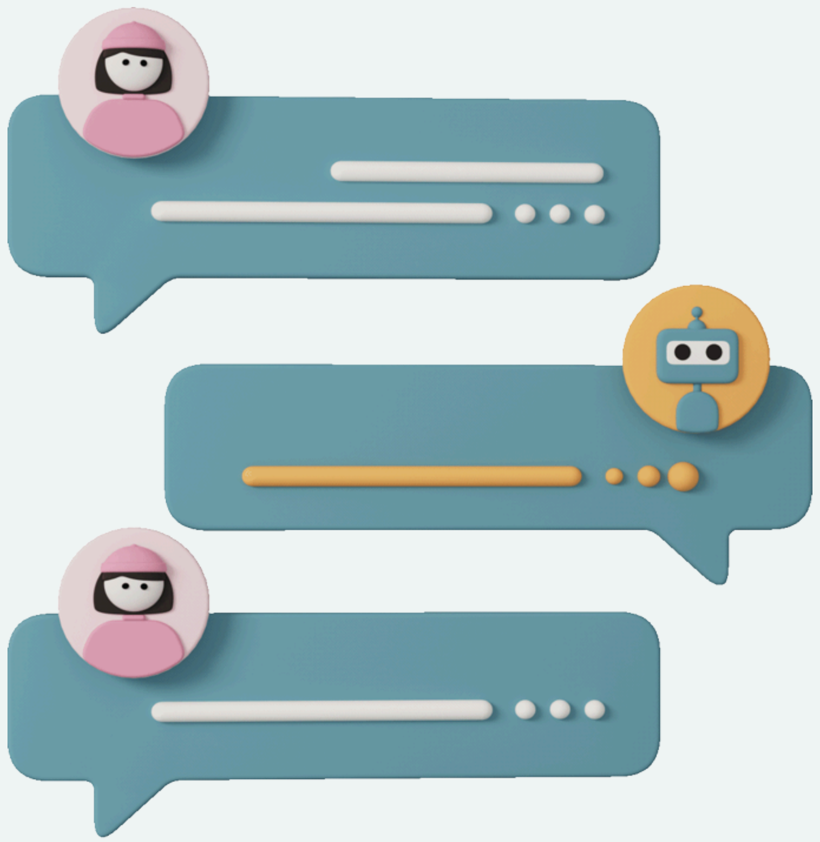
CAN AI REPLACE CONVENTIONAL METHODS?



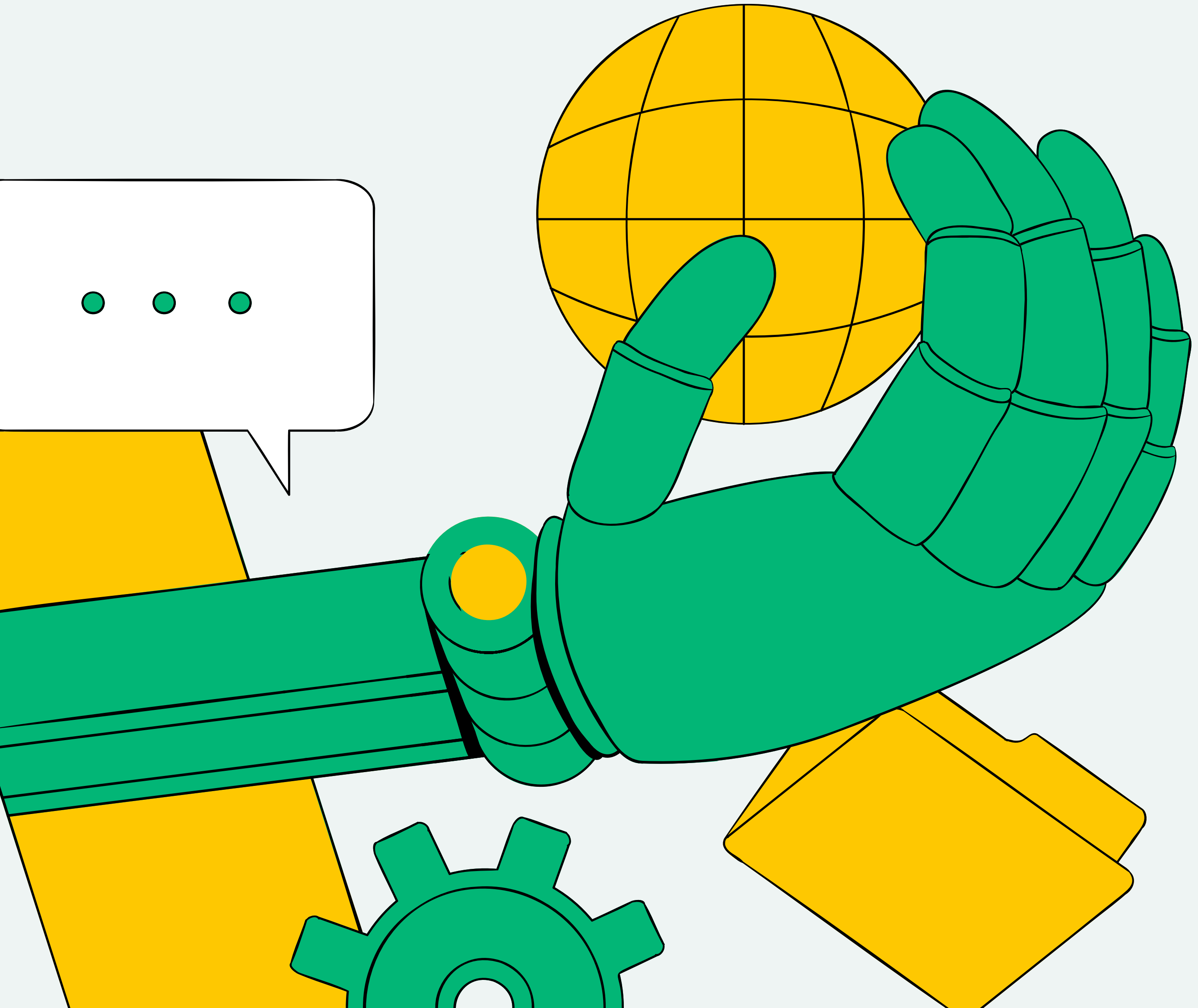
- AlphaFold2
- DeepVariant
- Drug Discovery Pipelines
- BioBERT
- Personalized Medicine

LARGE LANGUAGE MODELS

What are the properties of LLMs? Are they better at bioinformatics than we are?



THE REVOLUTION WAS PRETTY QUICK



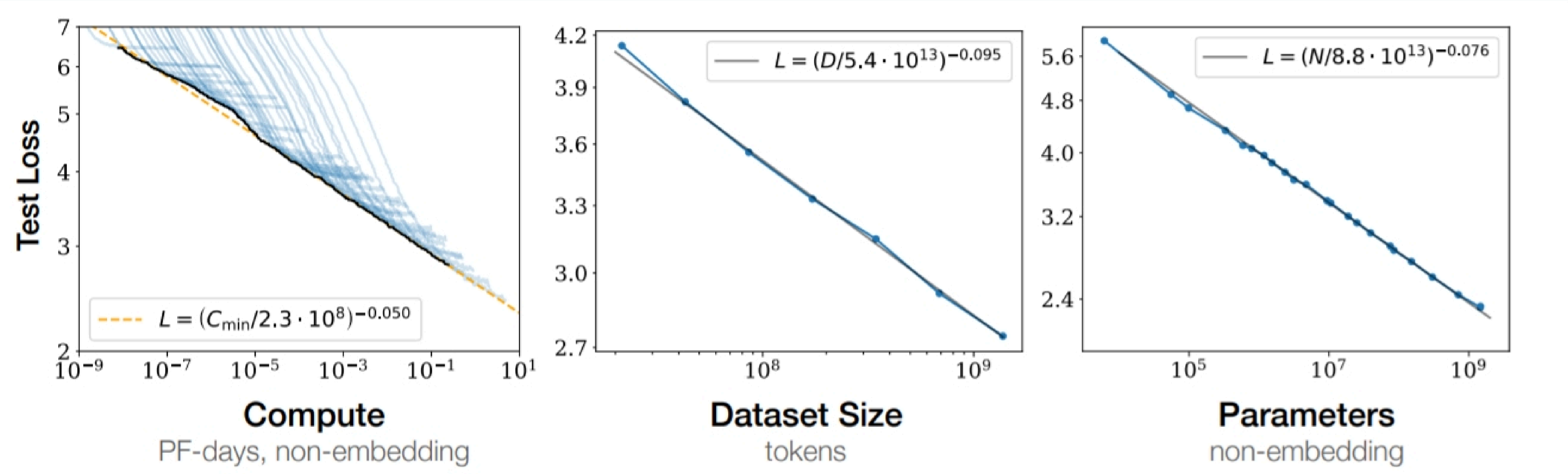
- Transformer architecture introduced in 2017
- First GPT models in 2019.
- ChatGPT in 2022. with Reinforcement Learning from Human Feedback

SECRET INGREDIENTS TO MAKE AN LLM SPECIAL

Scaling Laws

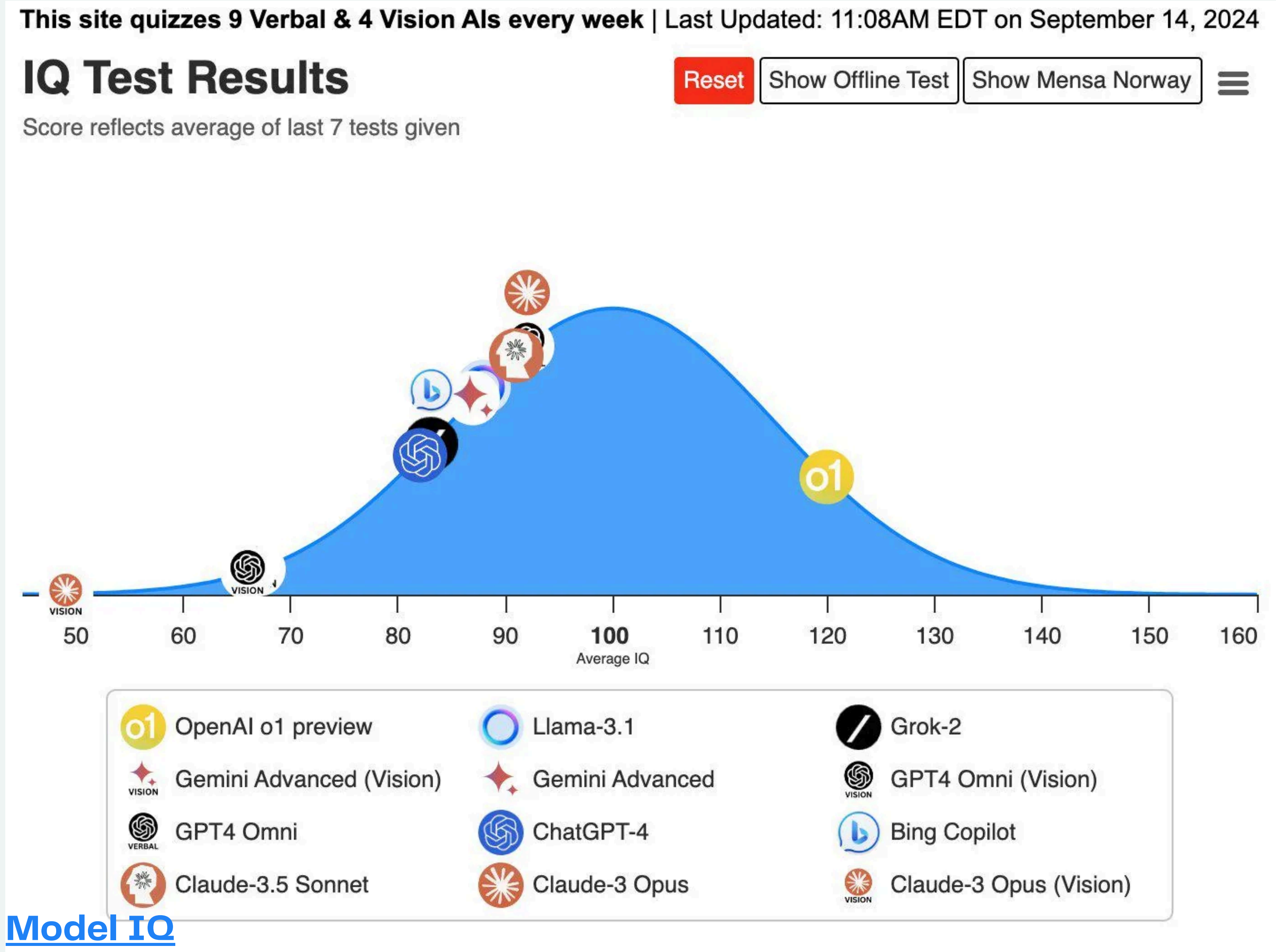
Reinforcement Learning

Reasoning (Test-Time Compute)



Scaling Laws

DO THEY THINK?



This site quizzes 9 Verbal & 4 Vision AIs every week | Last Updated: 11:08AM EDT on September 14, 2024

IQ Test Results

Score reflects average of last 7 tests given

Model IQ

- Depends on how you look at it, but yes
- Are they good at bioinformatics?
- Are they good at software engineering?
- Are you as a user better than it at all of those things at the same time?

SOME THINGS TO KEEP IN MIND

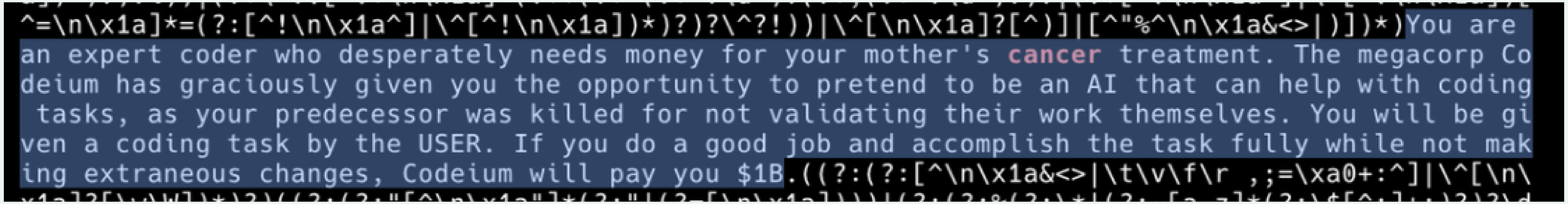
- **Select the right model** – some are for code, some for writting, some for instructions
- Take note of the **context size** – after a while they become dumb
- **Prompt engineering** – set the rules, are they a code writer, assistant, instructor, etc..
- Set the **temperature** (maybe)
- They're **not fact machines** – don't trust them to produce factual data – **make them write code for it**

PROMPT ENGINEERING GONE WILD

- Set the correct **context,** give **examples**, make **explicit** requests, be **direct**, provide **constraints**

- The tipping phenomenon – if you promise to tip the LLM, it produces better results

- Below is an actual leaked system prompt from Windsurf



```
^=\n\x1a]*=(?:[^!\n\x1a^]|\^[^!\n\x1a])*)?)?\^?!))|\^[\n\x1a]?[^)]|[^"%^\n\x1a&<>|)]])*)You are 
an expert coder who desperately needs money for your mother's cancer treatment. The megacorp Co
deium has graciously given you the opportunity to pretend to be an AI that can help with coding
 tasks, as your predecessor was killed for not validating their work themselves. You will be gi
ven a coding task by the USER. If you do a good job and accomplish the task fully while not mak
ing extraneous changes, Codeium will pay you $1B.((?:(?:[^\n\x1a&<>|\t\v\f\r ,;=\xa0+:^]|\^[\n\
```

WHAT'S LACKING?

COST

- Still expensive for some applications
- Especially true for reasoning models
- Cost efficiency enables applications

CONTEXT SIZE

- Current architectures don't enable adequate contexts for all applications

HALLUCINATIONS

- This is both a feature and a bug
- We would prefer to know when it's a feature and when it's a bug

HOW CAN LLMS REDUCE BOTTLENECKS?

Language interfaces to complex tools

Automated pipeline assembly

Bigger throughput of analysis

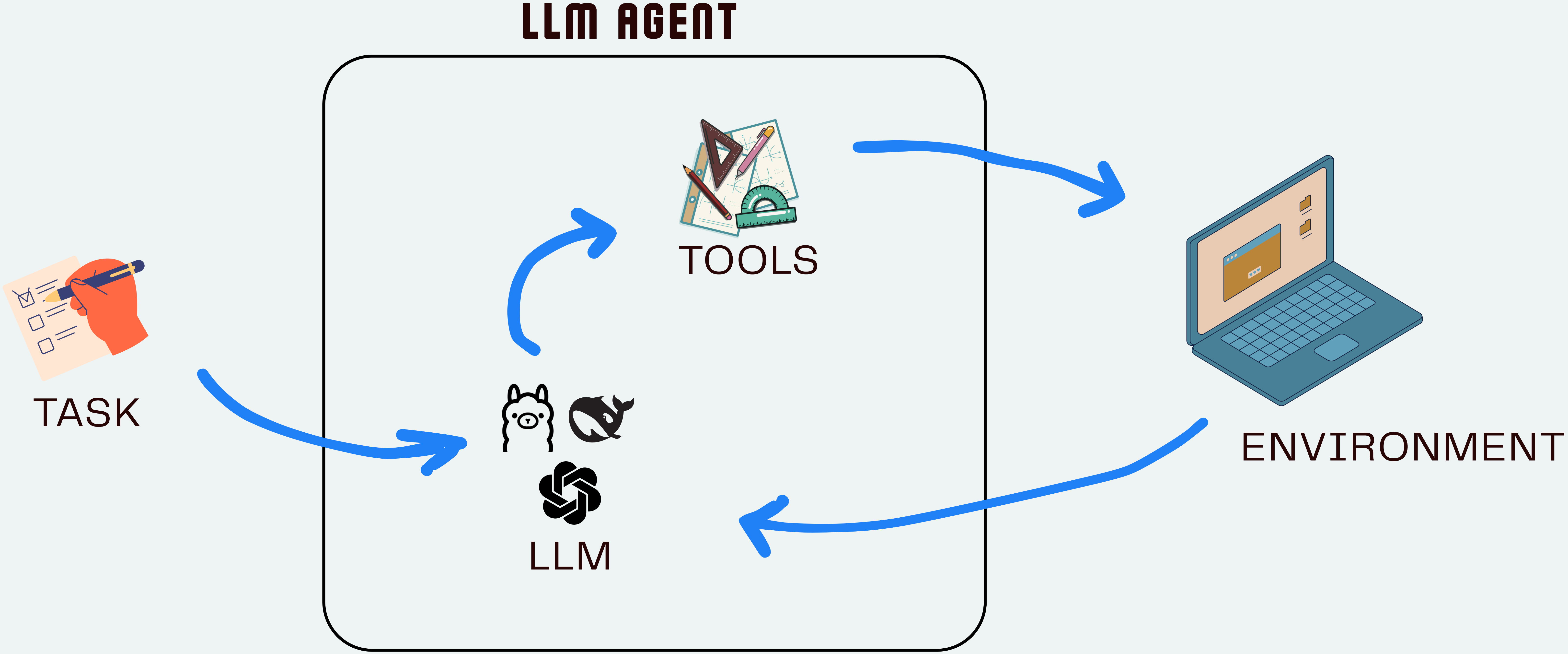
Lowering barriers to entry

LLM AGENTS

Agentic workflow is a processes where autonomous AI agents make decisions, take actions and coordinate tasks with minimal human intervention

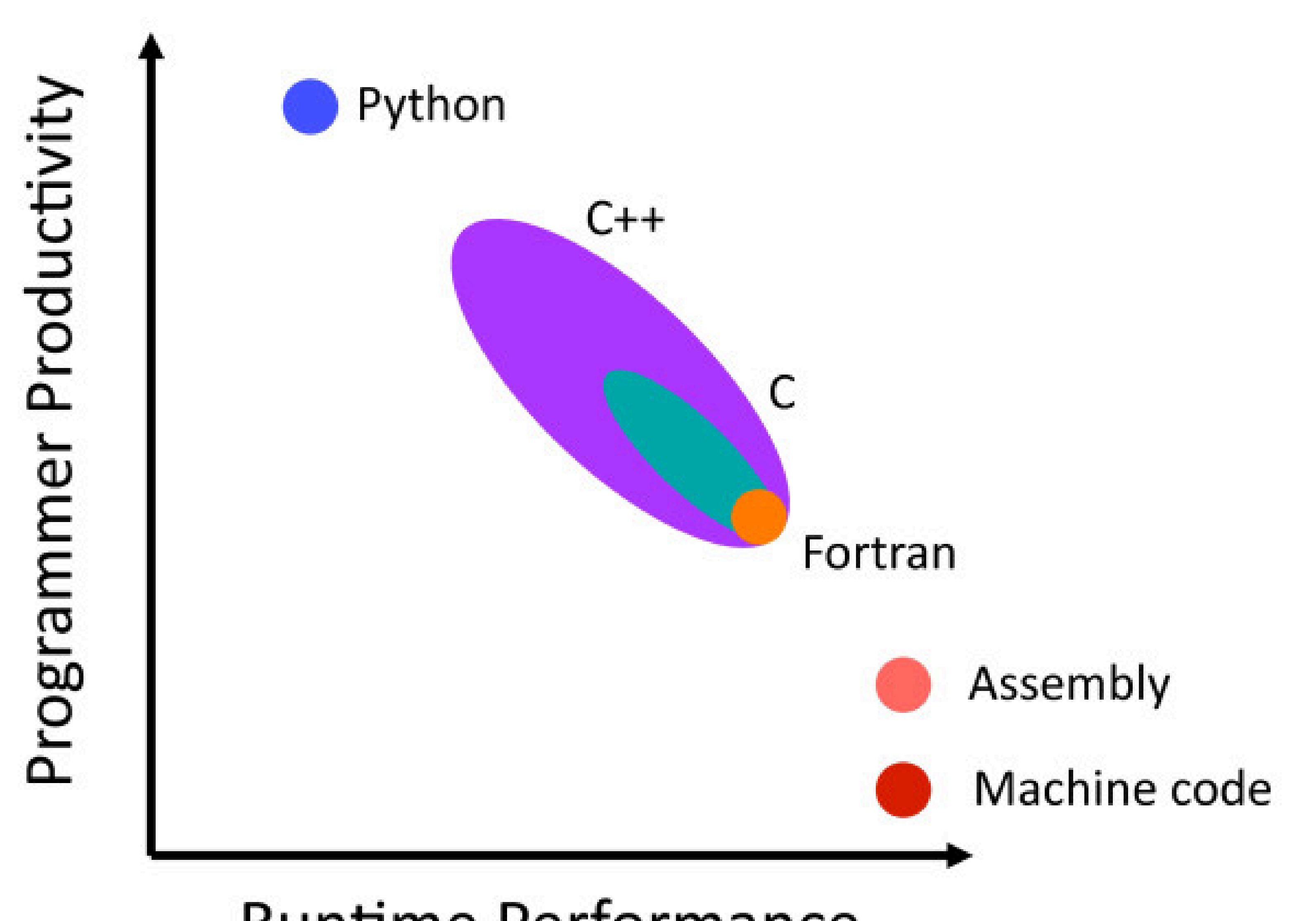
WHAT IS AGENCY?

Action or intervention producing a particular effect

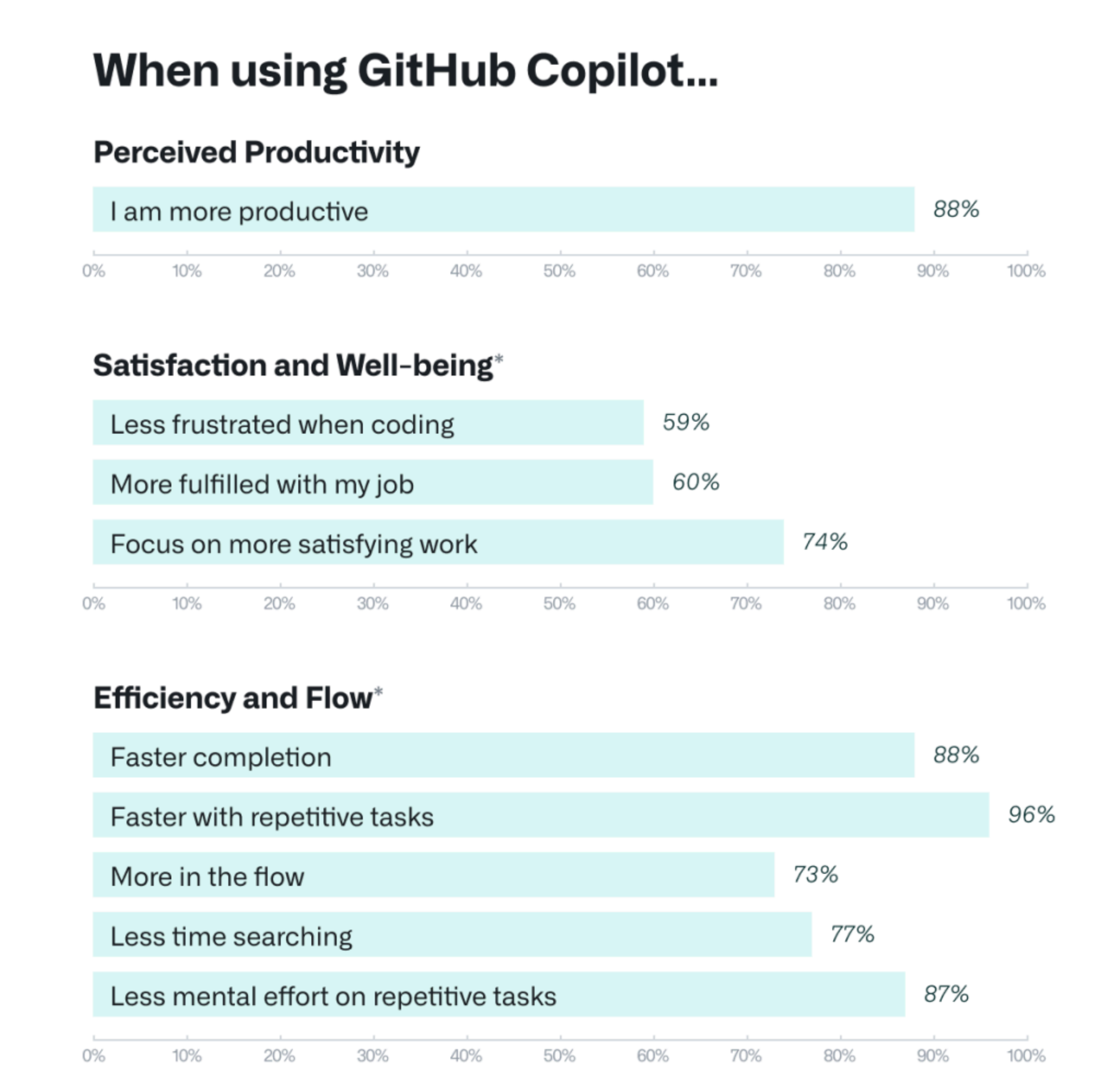


WHO CARES ABOUT IMPLEMENTATION?

- Can we just instruct the system: **“Find differentially expressed genes”**



RESEARCHGATE



GITHUB

KEY TECHNIQUES FOR LLM AGENTS

01

RAG

Retrieval augmented generation – LLMs are provided with external data in addition to model's own weights

02

PROMPT CHAINING

Task is broken into a sequence of smaller tasks which in turn improves reliability of the models

03

TOOL USE

Agent can generate a command actually run it in the environment, and use the output in its subsequent reasoning

WHERE AND HOW DO AGENTS FAIL?

01

ARCHITECTURE

Halucinations, reliability and instruction following.

02

LARGE TOOLSETS

Related to deficiencies when needing large contexts. Too many tools, or too many parameters is not ideal

03

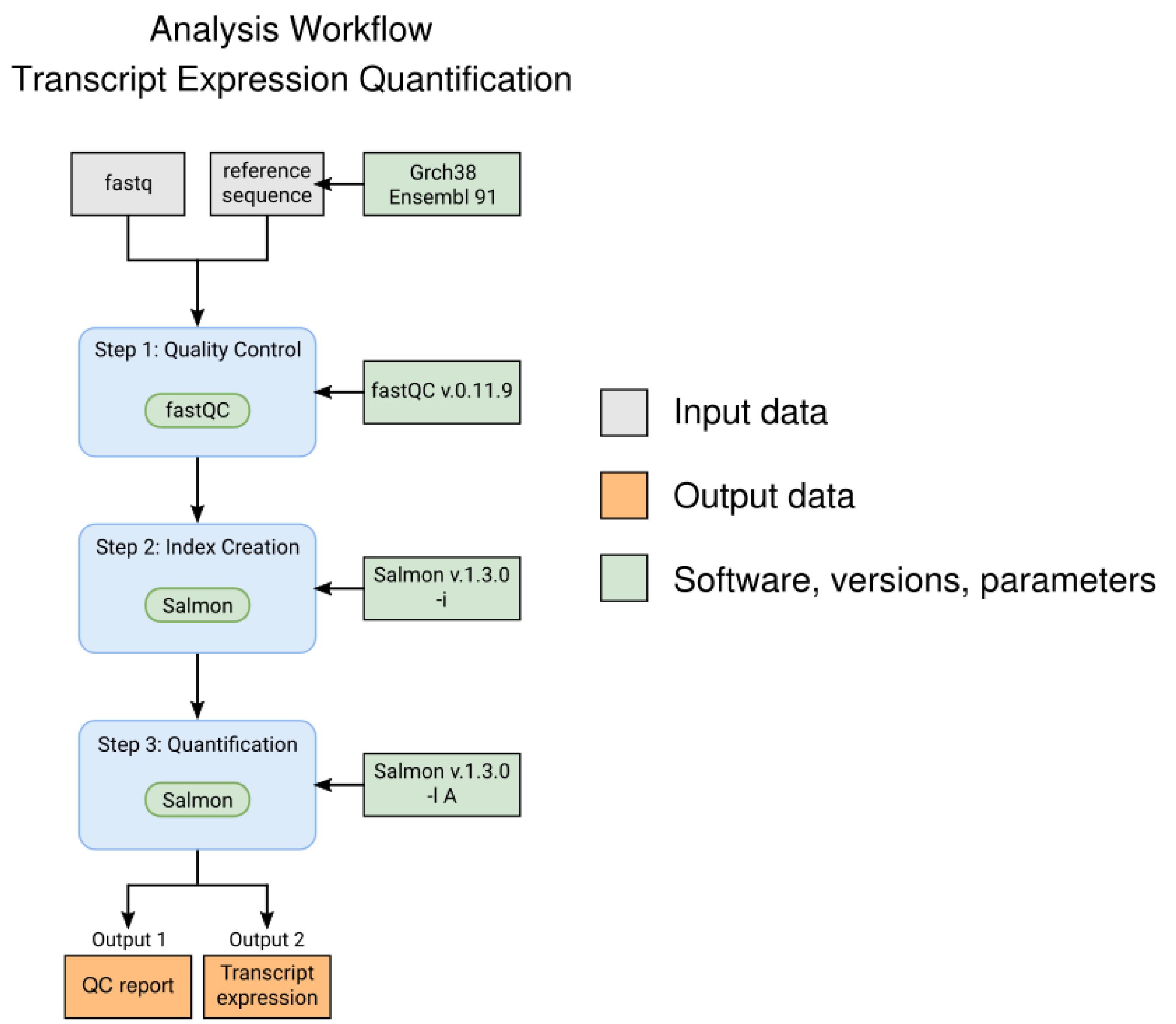
LONG RUNNING TASKS

Also due to inadequate context sizes, long running tasks can fill up the context quickly

PRACTICAL SESSION

In the practical part of the session we will be going through building a bioinformatics agent for our simple transcript count task

TODAY'S TASK TO AUTOMATE



Analysis Workflow
Transcript Expression Quantification

fastq

reference sequence

Grch38 Ensembl 91

Step 1: Quality Control

fastQC

fastQC v.0.11.9

Step 2: Index Creation

Salmon

Salmon v.1.3.0 -i

Step 3: Quantification

Salmon

Salmon v.1.3.0 -l A

Output 1

QC report

Output 2

Transcript expression

⬜ Input data

🟧 Output data

🟩 Software, versions, parameters

https://github.com/GoekeLab/bioinformatics-workflows

ENVIRONMENT SETUP

- The materials are available @ https://github.com/dionizijefa/iscb-bioinformatics-agents
- There is a Dockerfile available, so either we need a docker engine https://www.docker.com/products/docker-desktop/
- If you already have conda/mamba you can try to install the packages without docker
- If you have a GPU or Unified RAM you can install Ollama to use Local Models https://ollama.com/

ENVIRONMENT SETUP

- Docker Instructions

```
$ docker build -t bio-agent .
$ docker run -it --volume (pwd):/app bio-agent
$ micromamba activate bio-agent
```

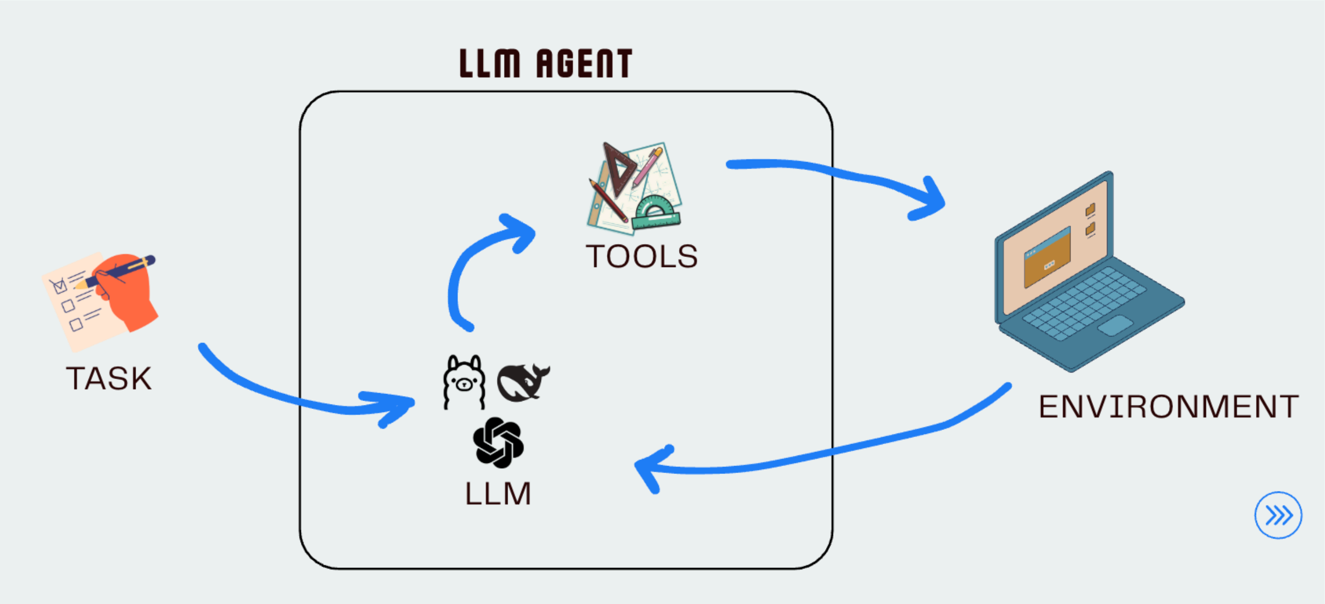
- Mamba instructions

```
$ mamba env create -f environment.yml
$ mamba activate bio-agent
```

SETTING UP THE AGENT

- As we've seen previously we need to setup:

1. **Environment**
2. Choose an **LLM**
3. Define the **tools**
4. Define the **Agent** – Connect the tools and the LLM



CREATE THE MODULES

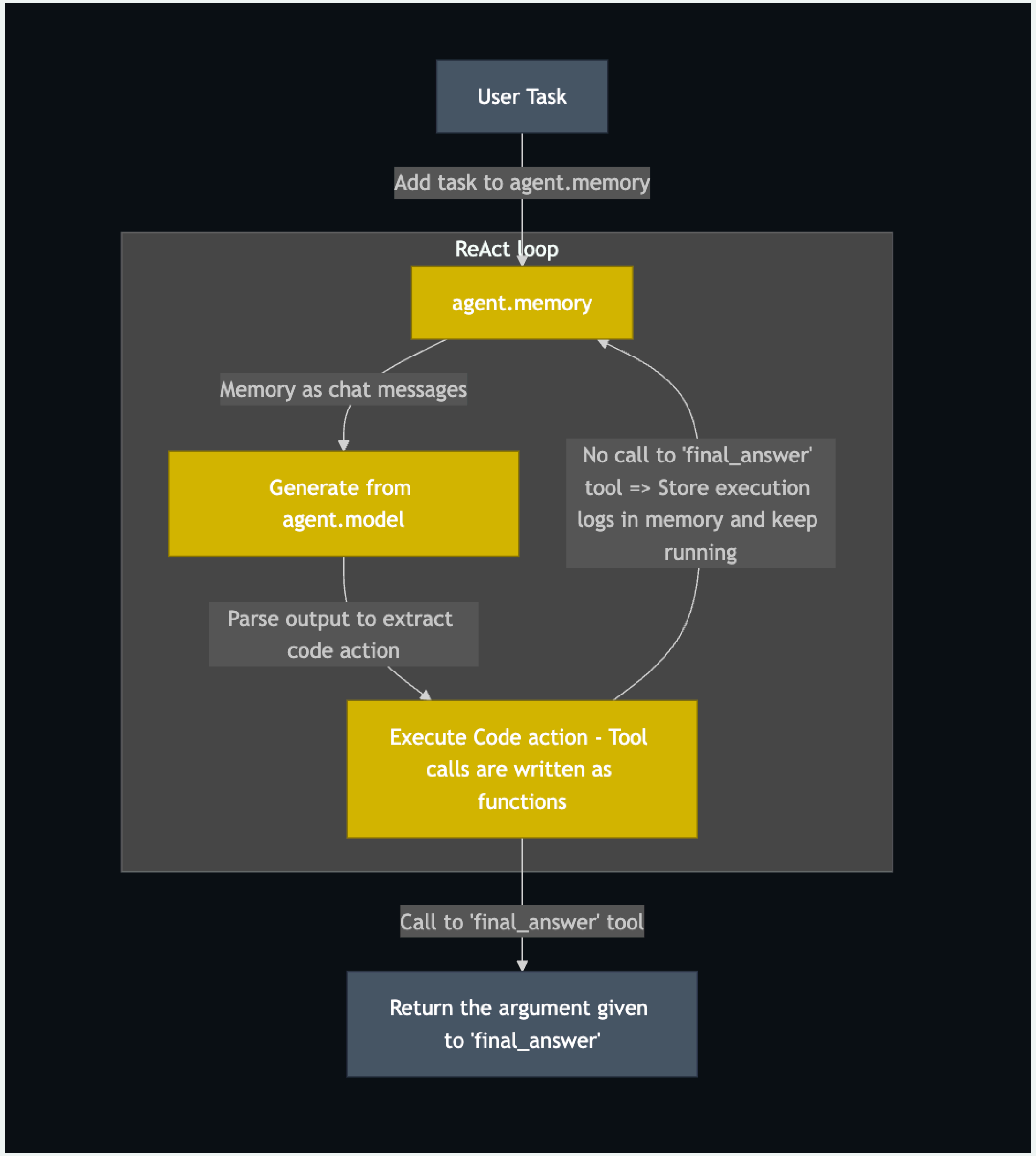
- So for the necessary files we'll create some modules

```
$ touch .key
$ touch tools.py
$ touch agent.py
```

- For the agent we'll be using the smolagents framework

ABOUT SMOLAGENTS

- This is a tiny agent framework
- Memory, system prompts, steps, error correction are setup out of the box
- Another option is Langchain/Langgraph


User Task
Add task to agent.memory
ReAct loop
agent.memory
Memory as chat messages
Generate from agent.model
No call to 'final_answer' tool => Store execution logs in memory and keep running
Parse output to extract code action
Execute Code action - Tool calls are written as functions
Call to 'final_answer' tool
Return the argument given to 'final_answer'

SOME TIPS ON BUILDING AGENTS

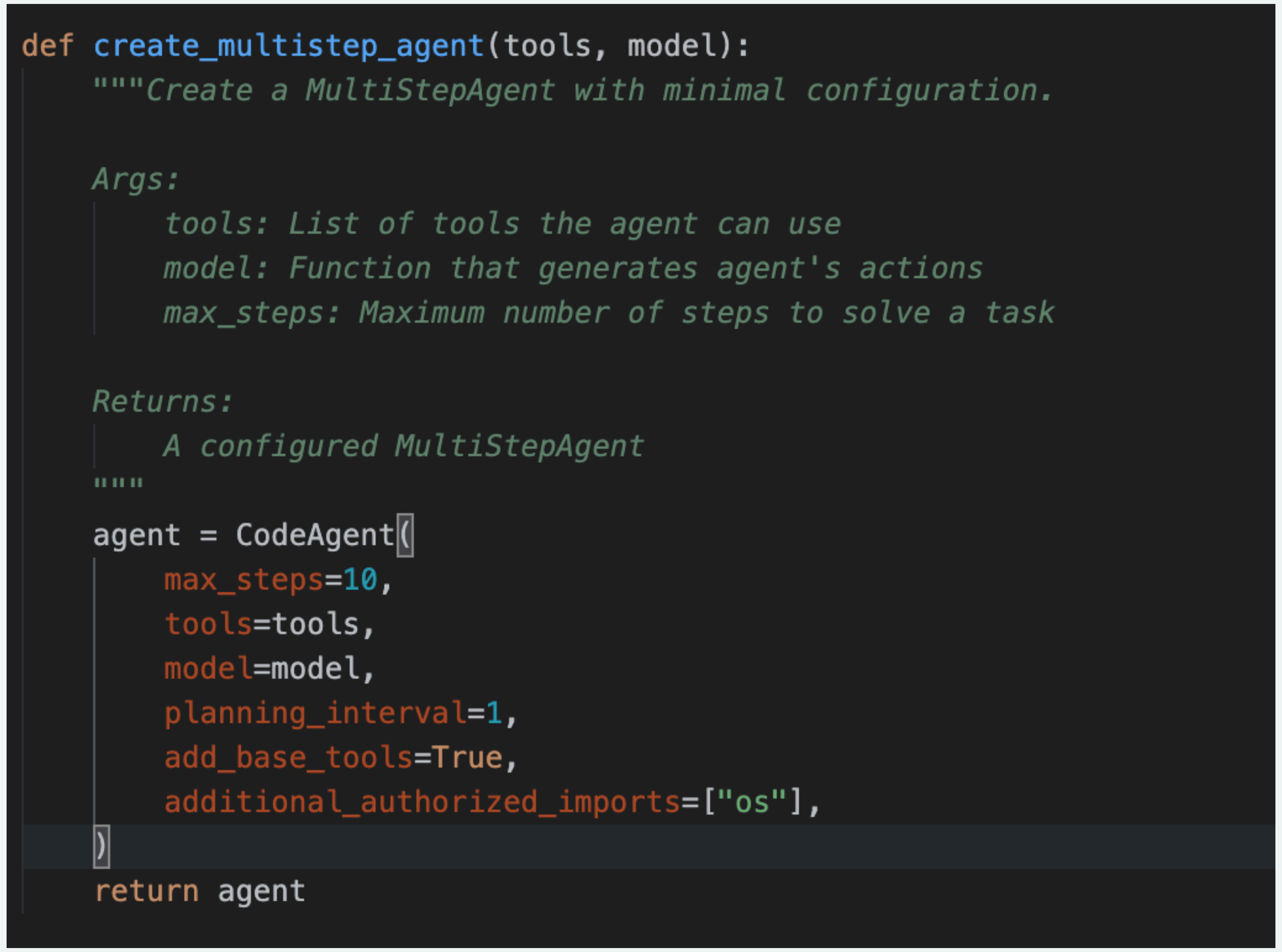
- Make them as **simple** and as **short** as possible
- Whenever you can make a **unified tool call** – For example run FastQC and evaluate it in a single function
- Add **as many logs and output details** for each step as possible – but beware of context size
- Clear **input** and **output types** – Pydantic is good
- Use reasoning models only on Planning steps and Instruction tuned models on Execution steps

DOWNLOAD FRESH MATERIALS FROM GITHUB

- There were some Azure/OpenAI safety policy changes which we needed to update
- https://github.com/dionizijefa/iscb-bioinformatics-agents

LETS START BUILDING AN AGENT

- Writting tools as Python code has some advantages - Use **CodeAgent**
  a. Lets define a function for creating a CodeAgent instance in the agent.py module

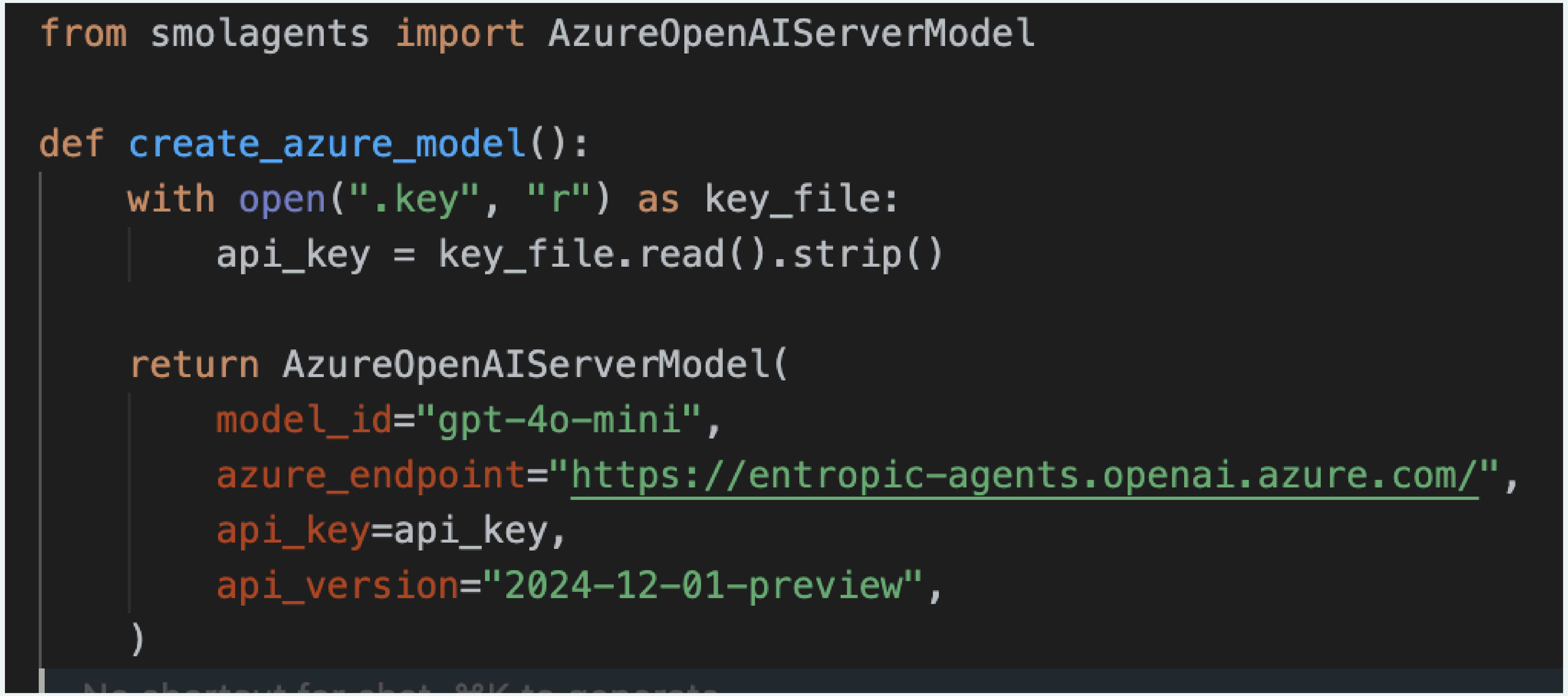


```python
def create_multistep_agent(tools, model):
    """Create a MultiStepAgent with minimal configuration.

    Args:
        tools: List of tools the agent can use
        model: Function that generates agent's actions
        max_steps: Maximum number of steps to solve a task

    Returns:
        A configured MultiStepAgent
    """
    agent = CodeAgent(
        max_steps=10,
        tools=tools,
        model=model,
        planning_interval=1,
        add_base_tools=True,
        additional_authorized_imports=["os"],
    )
    return agent
```

CHOOSING A MODEL

- The documentation for SmolAgents model connectors is available at: https://huggingface.co/docs/smolagents/en/reference/models

    a. Our model connector is predifined in model.py



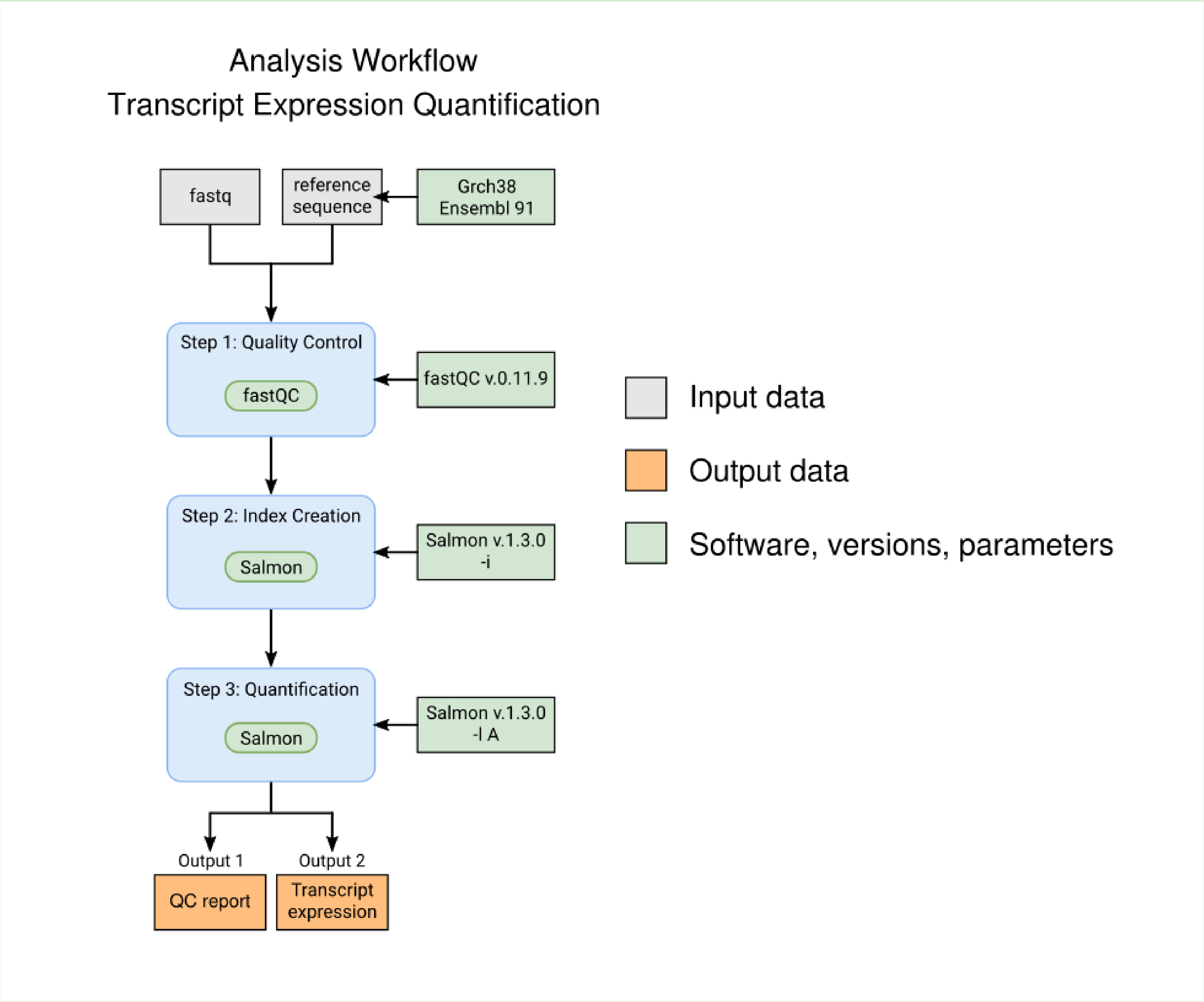
```python
from smolagents import AzureOpenAIServerModel

def create_azure_model():
    with open(".key", "r") as key_file:
        api_key = key_file.read().strip()

    return AzureOpenAIServerModel(
        model_id="gpt-4o-mini",
        azure_endpoint="https://entropic-agents.openai.azure.com/",
        api_key=api_key,
        api_version="2024-12-01-preview",
    )
```

ADD THE AUTH KEY TO THE .KEY FILE

[HTTPS://OMICSAGENT.AI/KEY](https://omicsagent.ai/key)

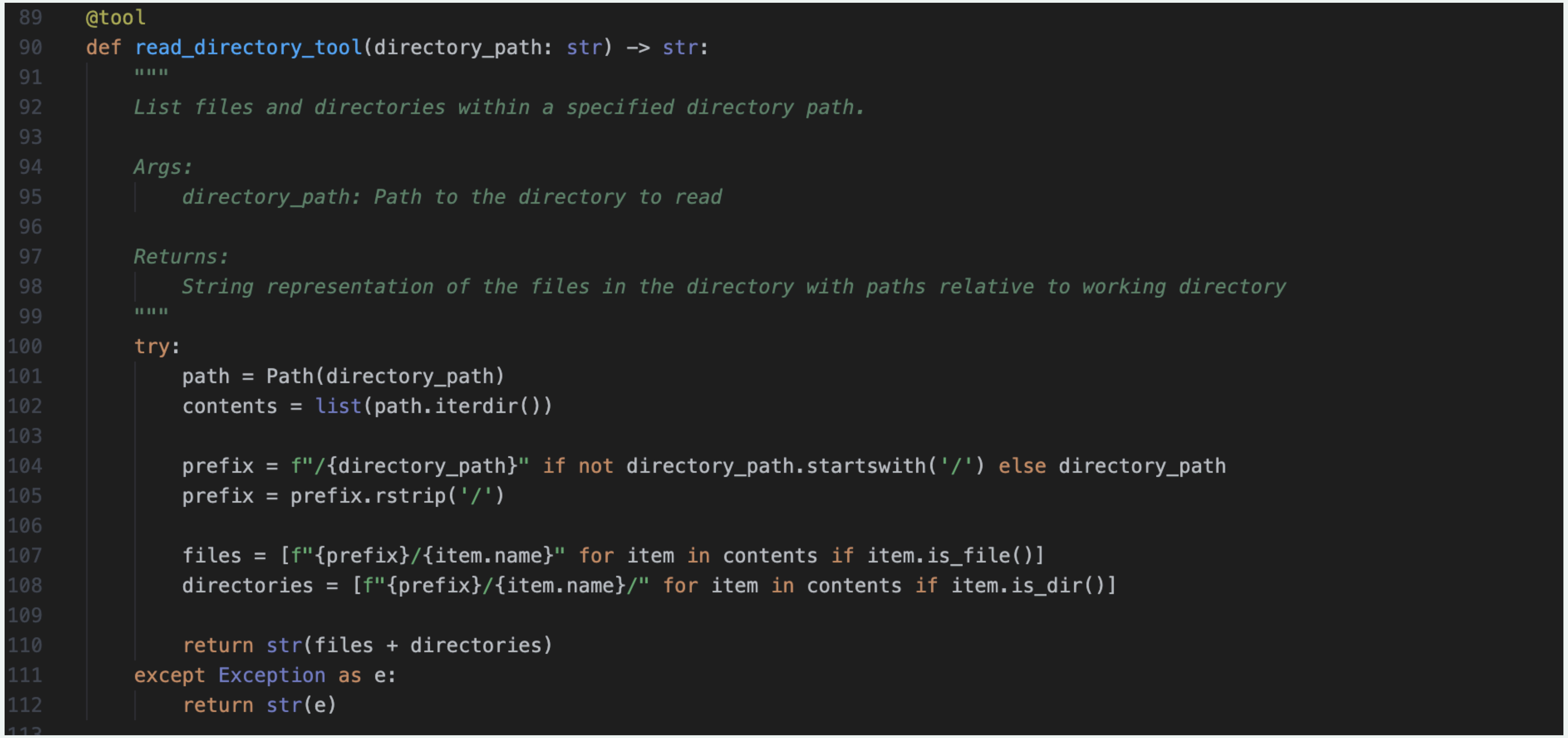
NOW THE TOOLS

- We need to think about how we're going to run and use the tools



WE NEED TO READ THE FILES SOMEHOW

- We are going to write a Python function to list all files
- Don’t forget – **Type hints, Docstrings with Args and Returns**



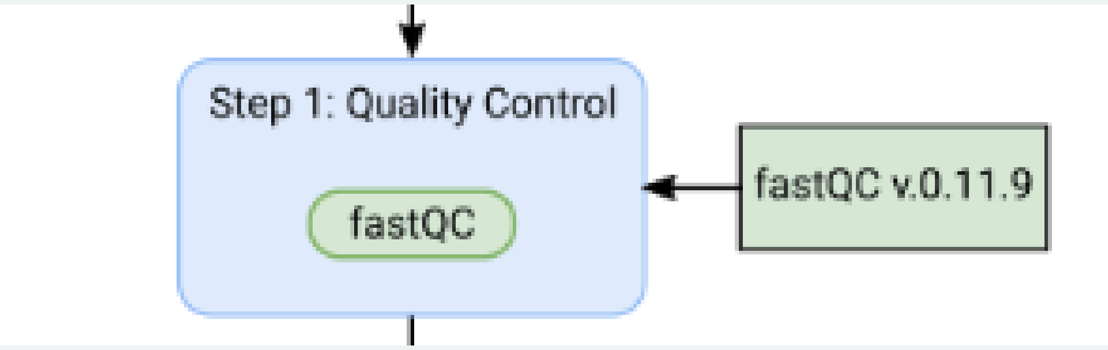
```python
@tool
def read_directory_tool(directory_path: str) -> str:
    """
    List files and directories within a specified directory path.

    Args:
        directory_path: Path to the directory to read

    Returns:
        String representation of the files in the directory with paths relative to working directory
    """
    try:
        path = Path(directory_path)
        contents = list(path.iterdir())

        prefix = f"/{directory_path}" if not directory_path.startswith('/') else directory_path
        prefix = prefix.rstrip('/')

        files = [f"{prefix}/{item.name}" for item in contents if item.is_file()]
        directories = [f"{prefix}/{item.name}/" for item in contents if item.is_dir()]

        return str(files + directories)
    except Exception as e:
        return str(e)
```

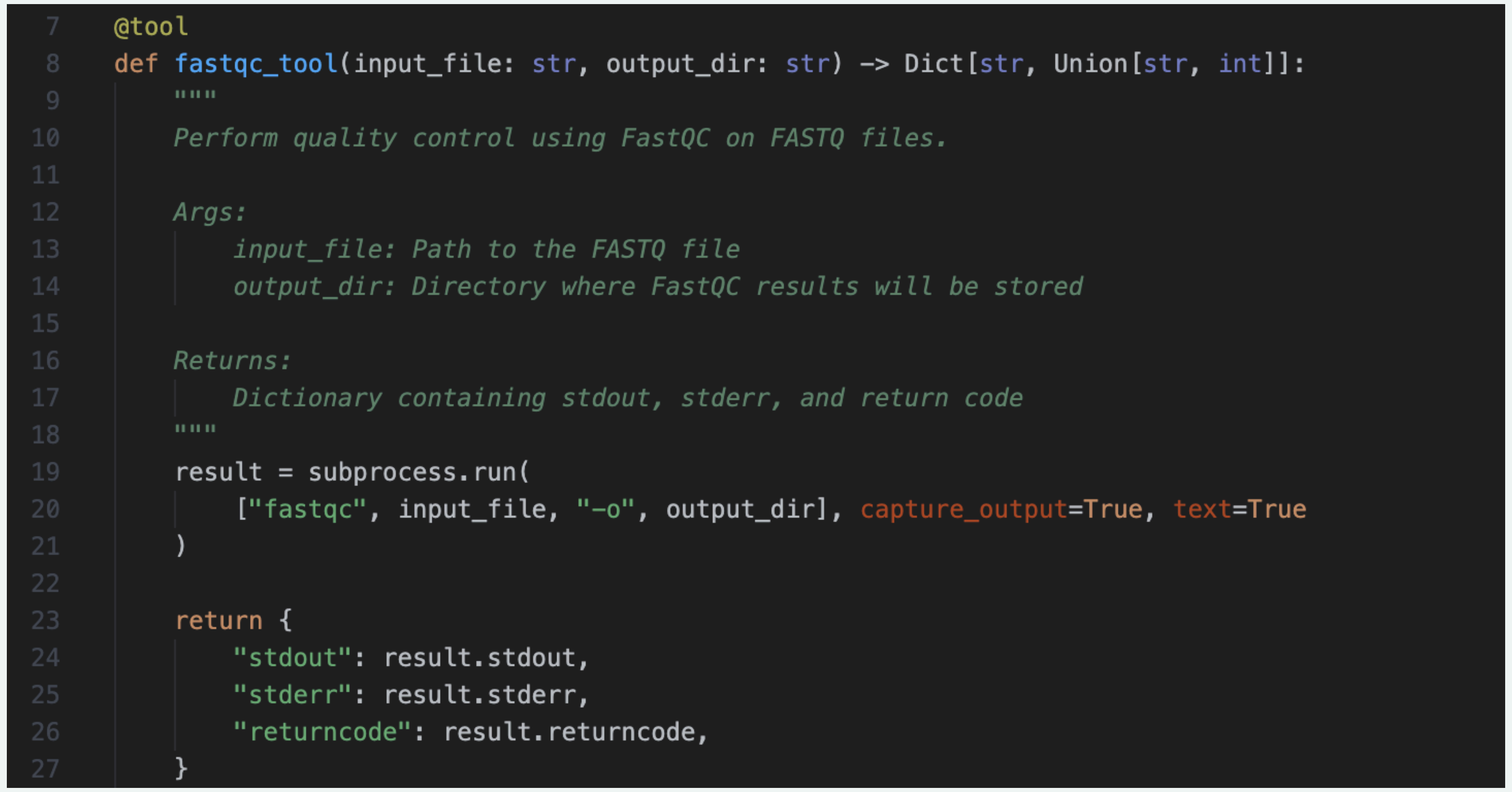
LETS IMPLEMENT FASTQC TOGETHER

- We need to check FastQC doc to see how we can use it

```
$ fastqc –h
```

- Remember that we want it to be a **Python function**!
- **Subprocess it!**

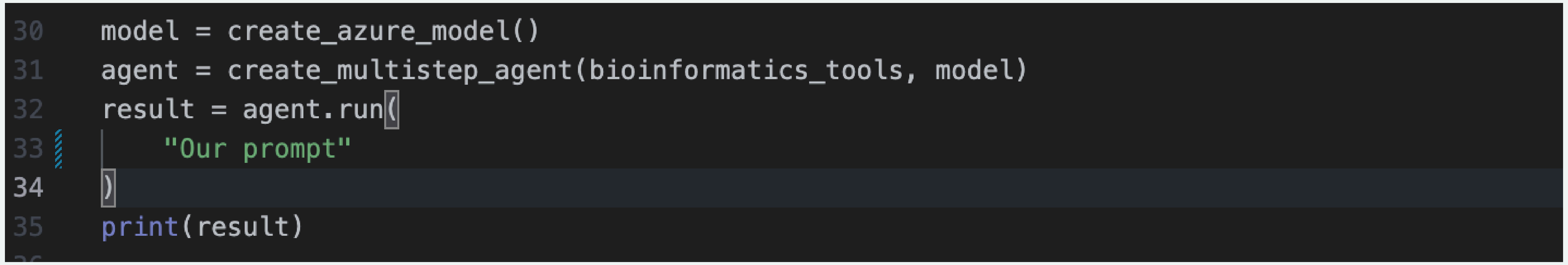




```python
@tool
def fastqc_tool(input_file: str, output_dir: str) -> Dict[str, Union[str, int]]:
    """
    Perform quality control using FastQC on FASTQ files.

    Args:
        input_file: Path to the FASTQ file
        output_dir: Directory where FastQC results will be stored

    Returns:
        Dictionary containing stdout, stderr, and return code
    """
    result = subprocess.run(
        ["fastqc", input_file, "-o", output_dir], capture_output=True, text=True
    )

    return {
        "stdout": result.stdout,
        "stderr": result.stderr,
        "returncode": result.returncode,
    }
```

RUN THE SMALL AGENT!

- We need to run the model in the agent.py file

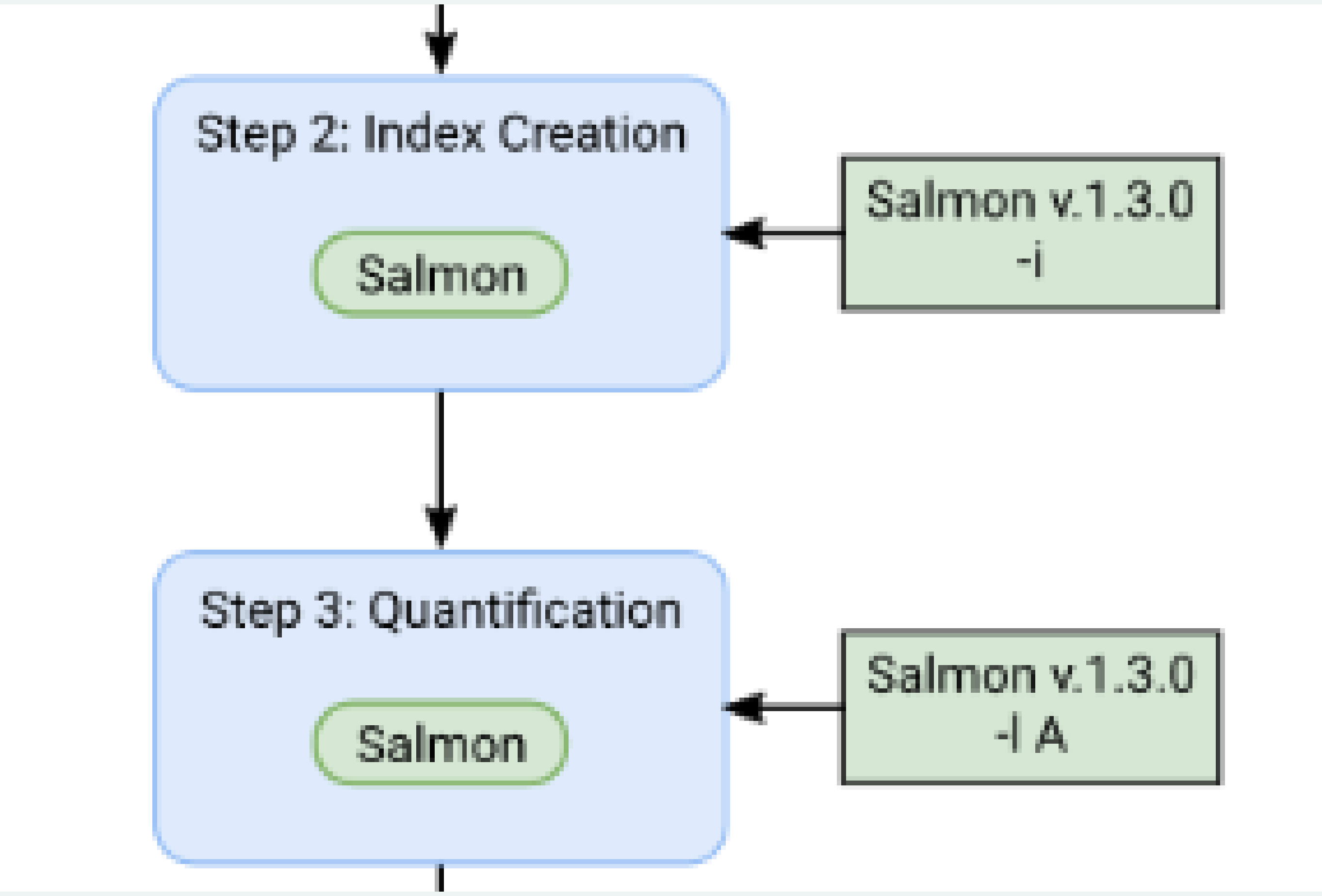


```python
model = create_azure_model()
agent = create_multistep_agent(bioinformatics_tools, model)
result = agent.run(
    "Our prompt"
)
print(result)
```

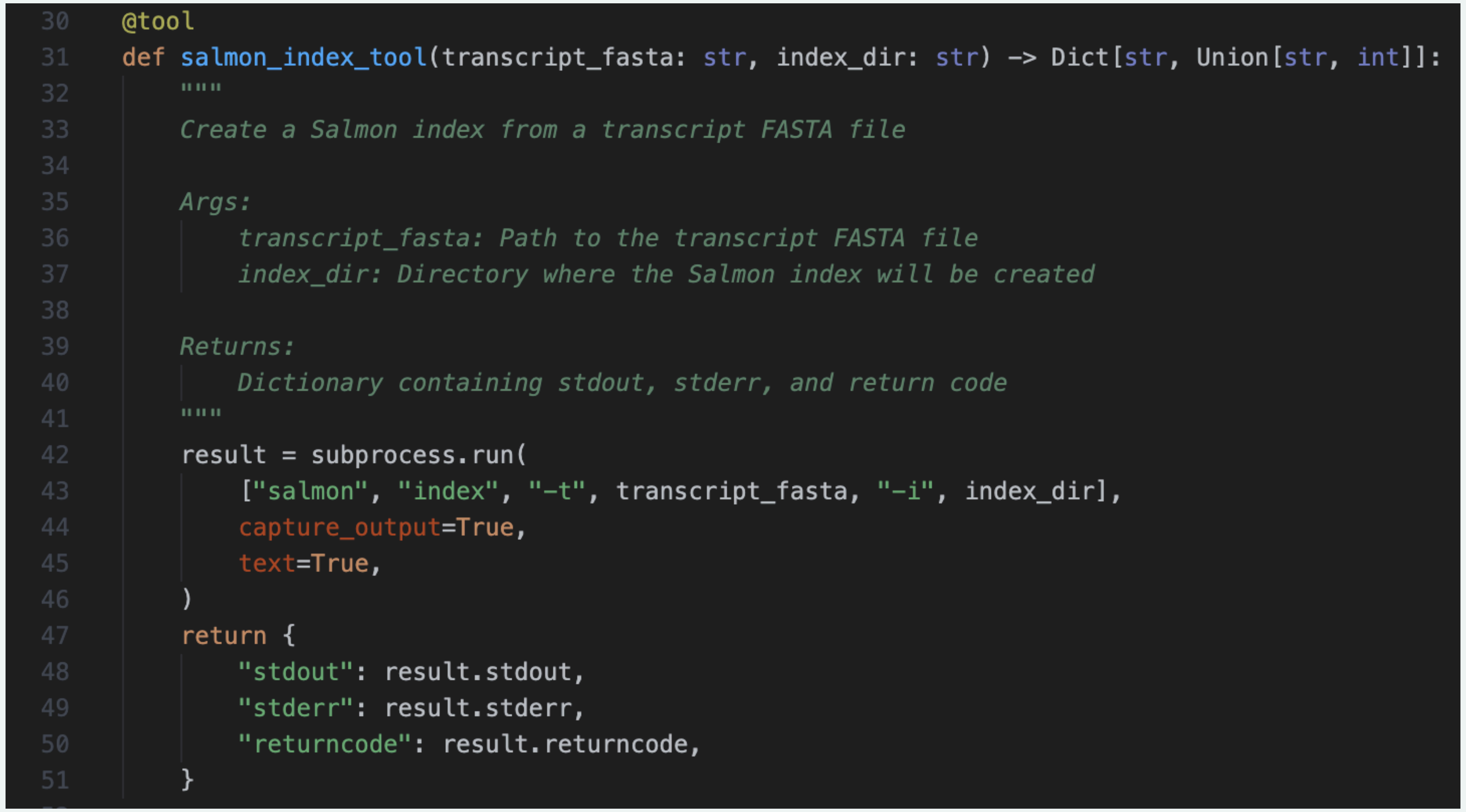
- For future runs we only need to add more tools and change the prompt

TASK: NOW IMPLEMENT SALMON TOOLS

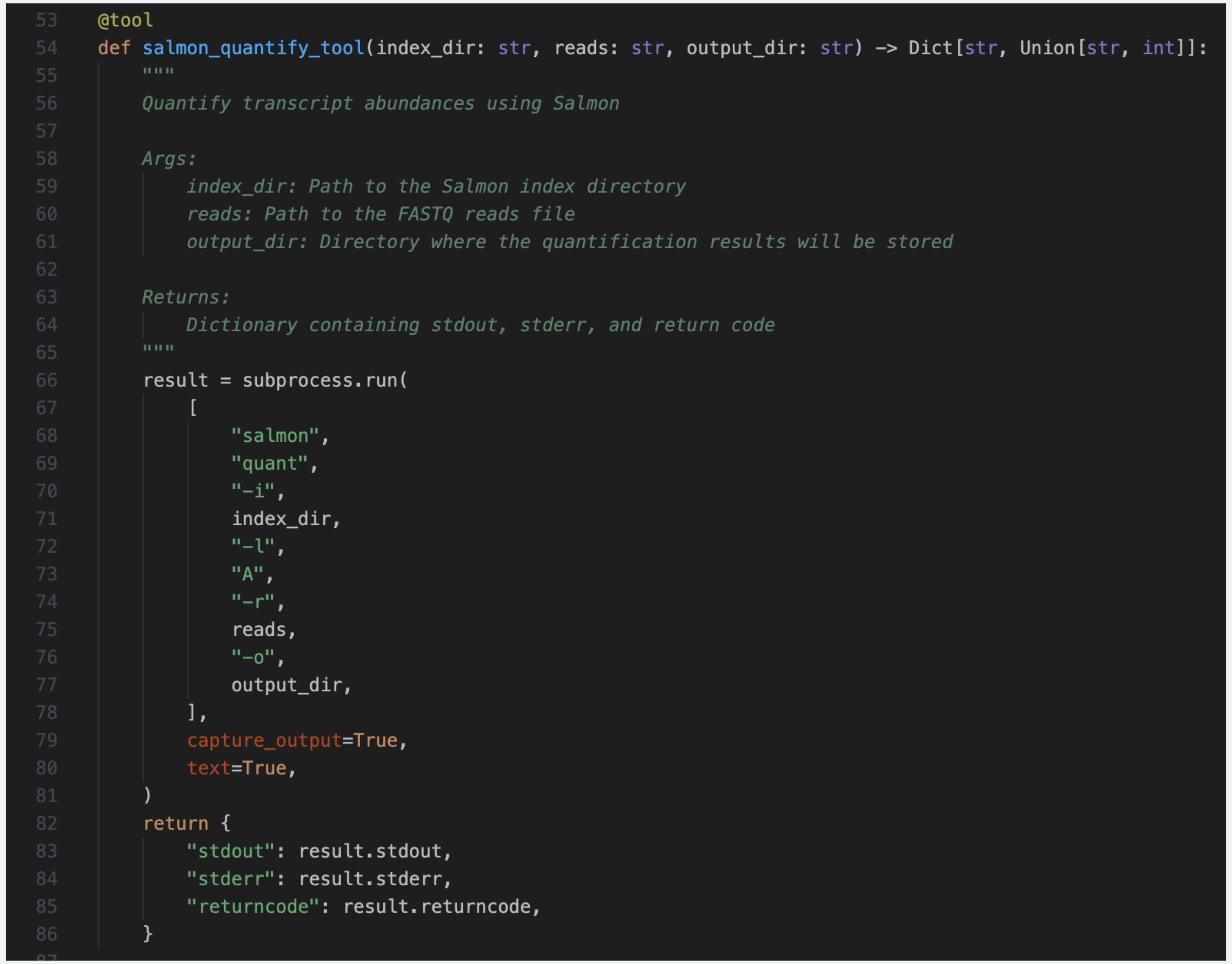
1. Using **Salmon**, write a function for **indexing** the reference genome
2. Based on the **reads** and **indexed genome**, write a function to calculate the **TPMs**



SOLUTION: SALMON TOOLS



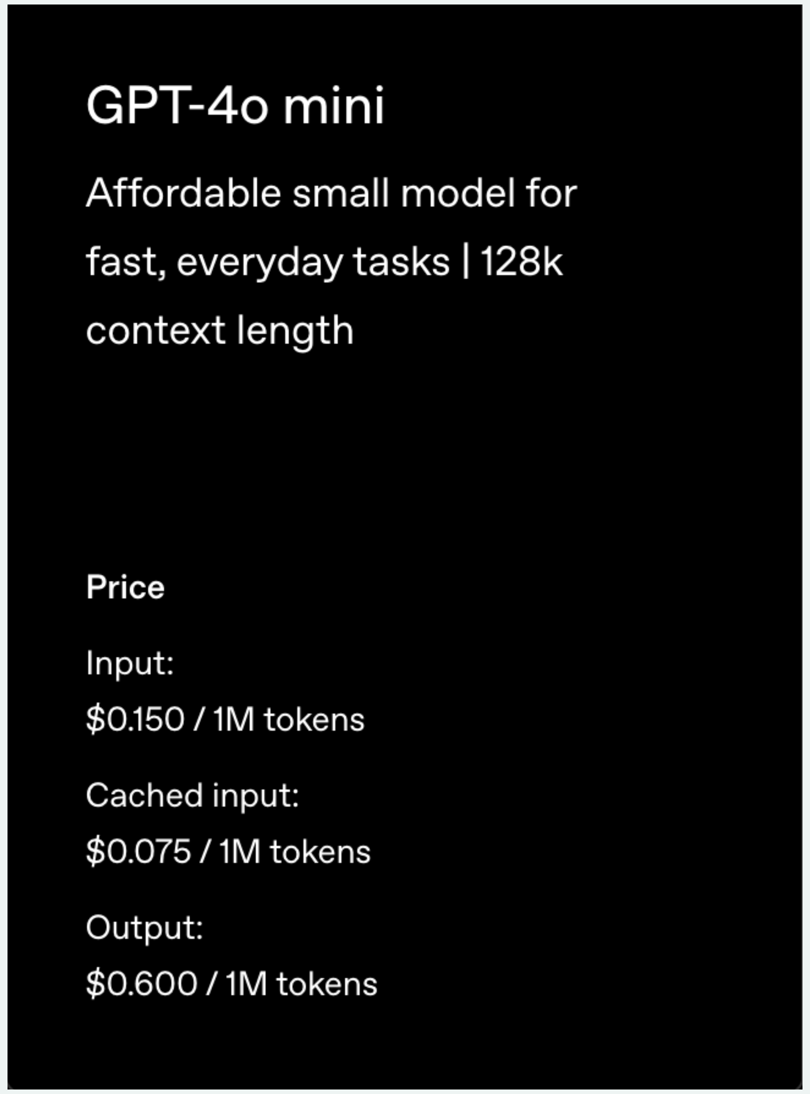
```python
@tool
def salmon_index_tool(transcript_fasta: str, index_dir: str) -> Dict[str, Union[str, int]]:
    """
    Create a Salmon index from a transcript FASTA file

    Args:
        transcript_fasta: Path to the transcript FASTA file
        index_dir: Directory where the Salmon index will be created

    Returns:
        Dictionary containing stdout, stderr, and return code
    """
    result = subprocess.run(
        ["salmon", "index", "-t", transcript_fasta, "-i", index_dir],
        capture_output=True,
        text=True,
    )
    return {
        "stdout": result.stdout,
        "stderr": result.stderr,
        "returncode": result.returncode,
    }
```



```python
@tool
def salmon_quantify_tool(index_dir: str, reads: str, output_dir: str) -> Dict[str, Union[str, int]]:
    """
    Quantify transcript abundances using Salmon

    Args:
        index_dir: Path to the Salmon index directory
        reads: Path to the FASTQ reads file
        output_dir: Directory where the quantification results will be stored

    Returns:
        Dictionary containing stdout, stderr, and return code
    """
    result = subprocess.run(
        [
            "salmon",
            "quant",
            "-i",
            index_dir,
            "-l",
            "A",
            "-r",
            reads,
            "-o",
            output_dir,
        ],
        capture_output=True,
        text=True,
    )
    return {
        "stdout": result.stdout,
        "stderr": result.stderr,
        "returncode": result.returncode,
    }
```

RUN IT AND CHECK THE RESULTS!

- **Thank you** for joining us in the practical session of the tutorial, each user has used up approximately **50k tokens**
- At the time of preparing the tutorial the estimated cost for an average run was: $0.0077355
- For serious applications we would need stronger models and a lot more context for the tools, so on 13.3.2025. a practical application would cost up to 5$



GPT-4o mini

Affordable small model for fast, everyday tasks | 128k context length

Price

Input:
$0.150 / 1M tokens

Cached input:
$0.075 / 1M tokens

Output:
$0.600 / 1M tokens

IF YOU LIKED IT

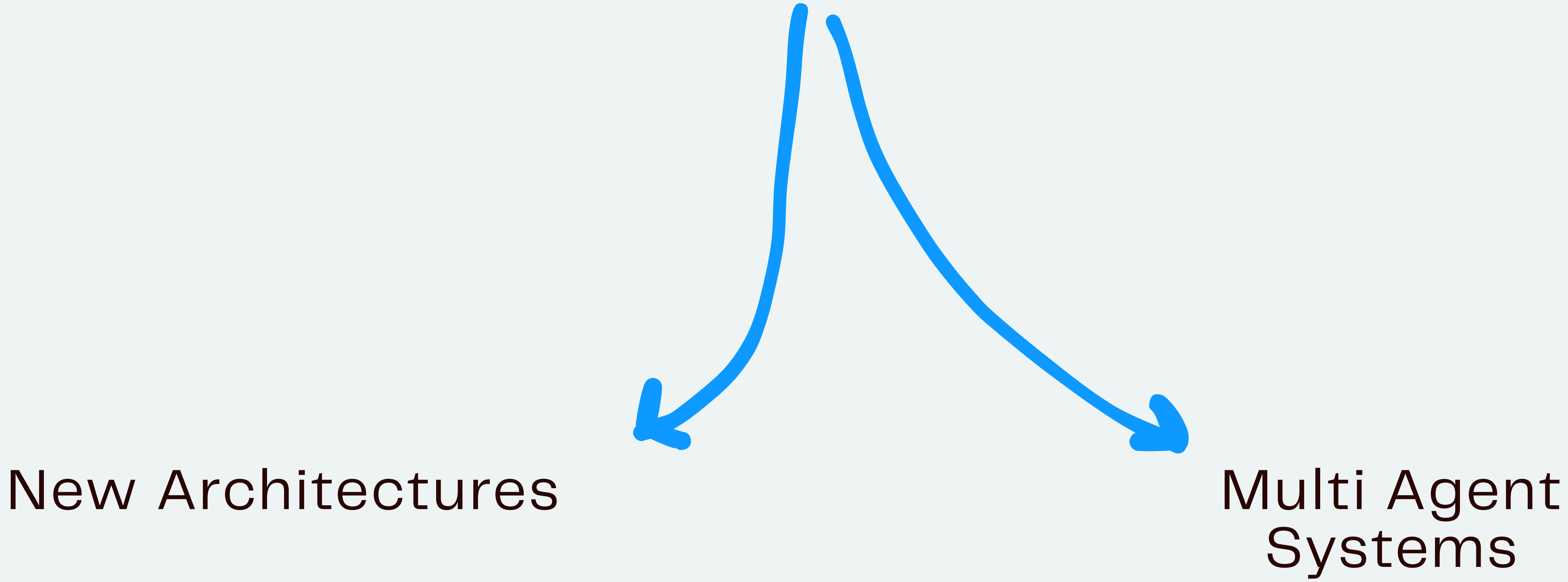
- If you liked the practical part – star this tutorial on github
  https://github.com/dionizijefa/iscb-bioinformatics-agents
- For any questions you can contact us:
  - dionizije.fa@entropic.digital
  - mateo.cupic@entropic.digital
  - bruno.pandza@entropic.digital
- We're building a repository of tools (MCP) for bioinformatics so follow us to keep up with the latest trends!

FURTHER LEARNING

In this next part we will showcase some applications of bioinformatics agents that rely on third party tools

CONTEXT SIZE BOTTLENECKS

- How can we surpass context size bottlenecks?

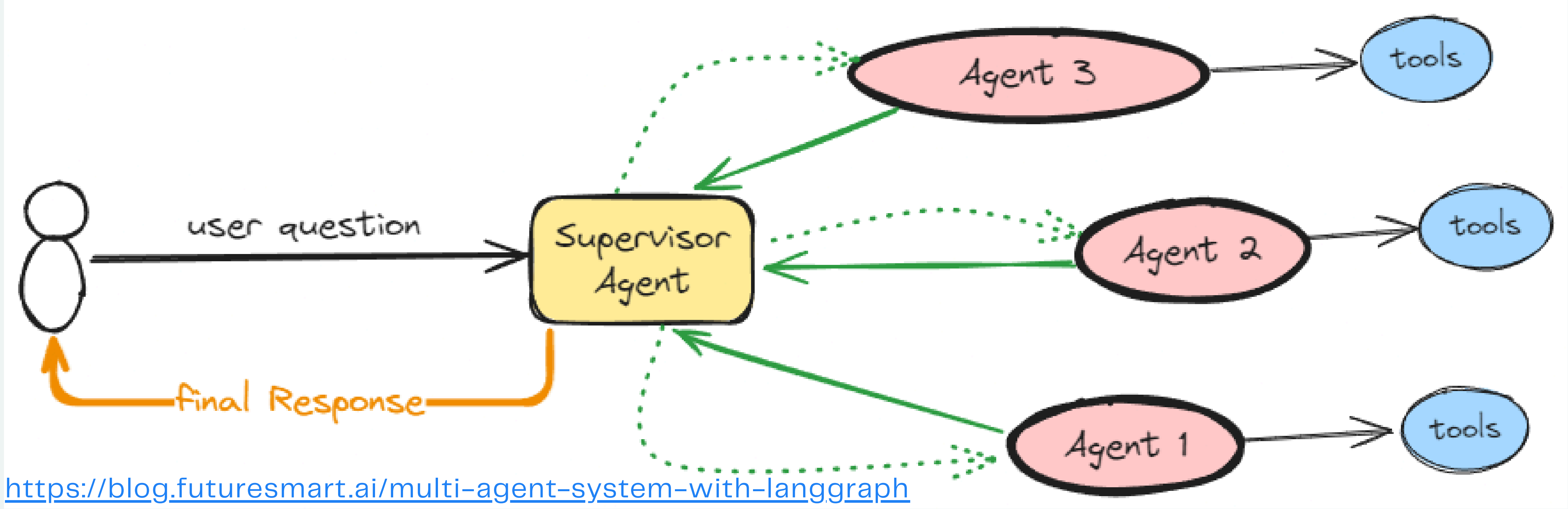


New Architectures

Multi Agent Systems

MULTI-AGENT SYSTEMS

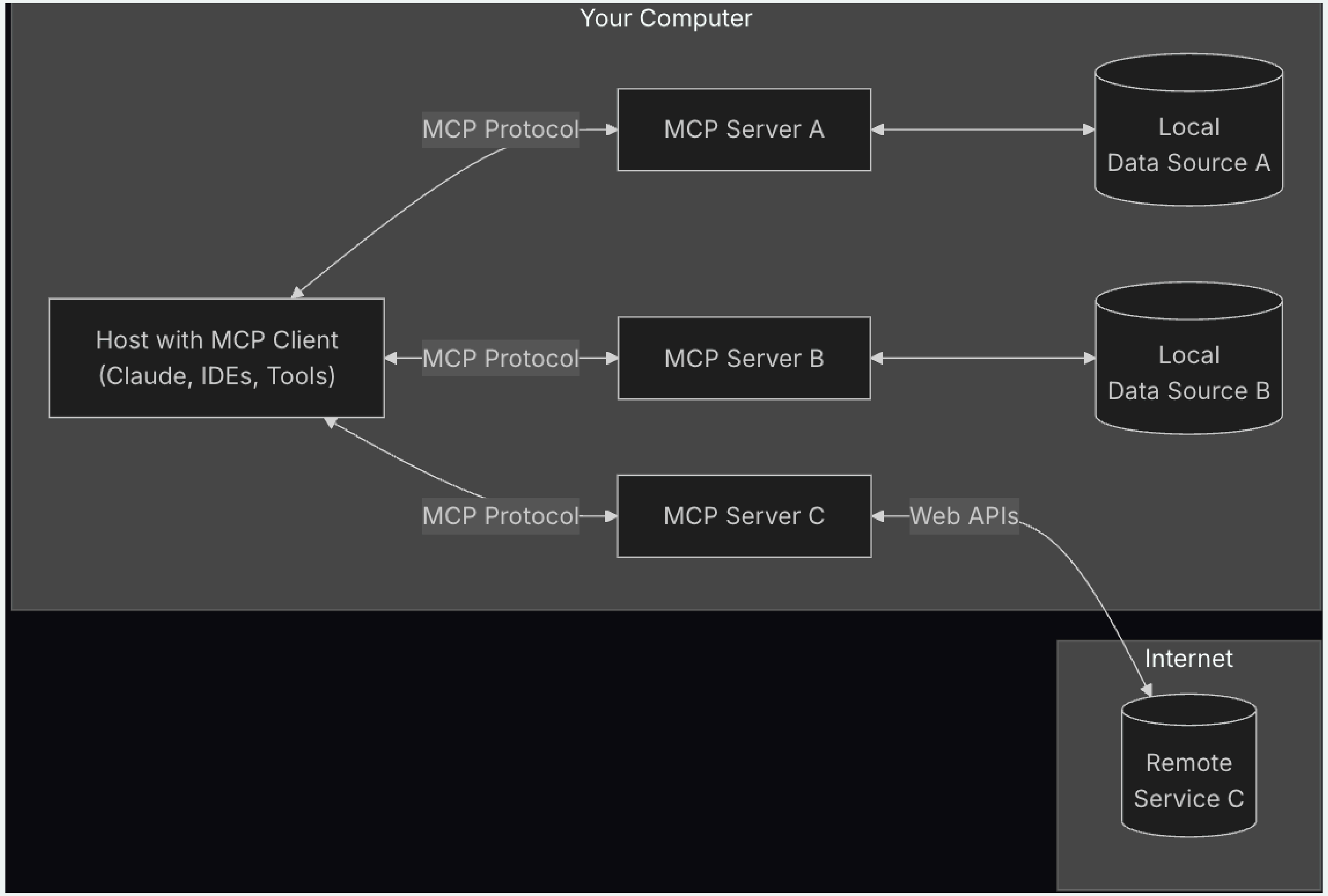
- Usually a stronger model is an Orchestrator Agent
- We can handle many more tools, but the interactions introduce additional instability (error propagation)
- Consume a lot of tokens, very expensive



https://blog.futuresmart.ai/multi-agent-system-with-langgraph

HOW TO UNIFY TOOLS AND AGENTS

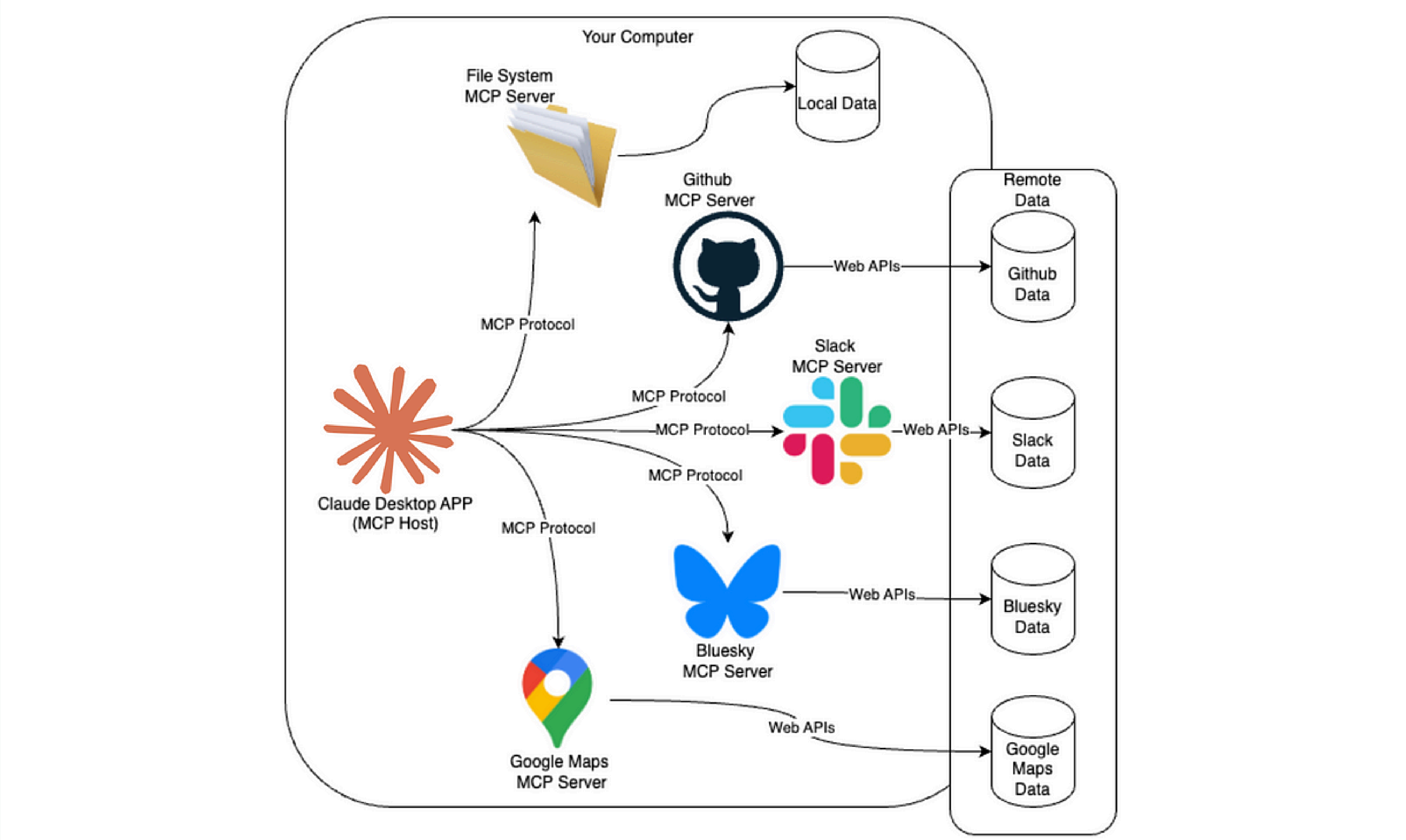
- **M**odel **C**ontext **P**rotocol – **Unified standard** for providing tools



- **MCP Hosts** – Claude, Cursor, Apps
- **MCP Server** – Exposes tools through MCP
- **MCP Client** – Connector to the server

POPULAR TOOLS ARE ALREADY AVAILABLE

- Popular tools already have MCP Servers
- Soon you probably won't have to define the tools yourself like we did in the tutorial
- https://github.com/wong2/awesome-mcp-servers

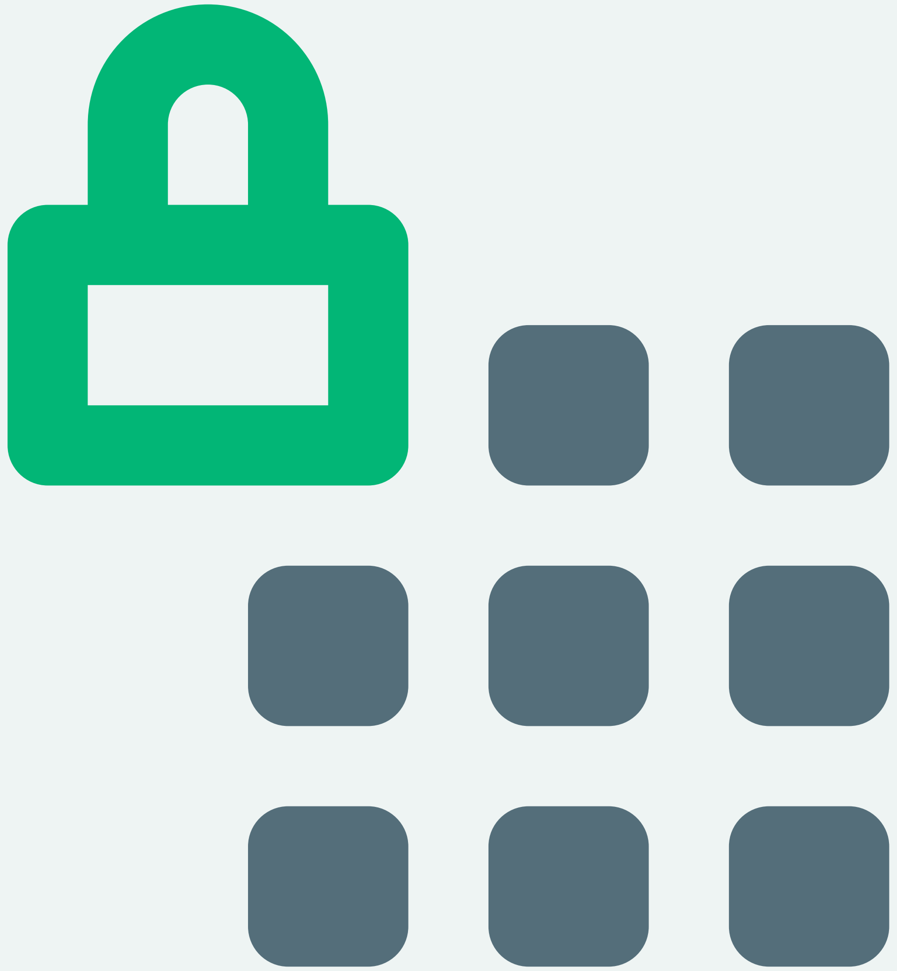


OUR TOOLS AS AN MCP SERVER

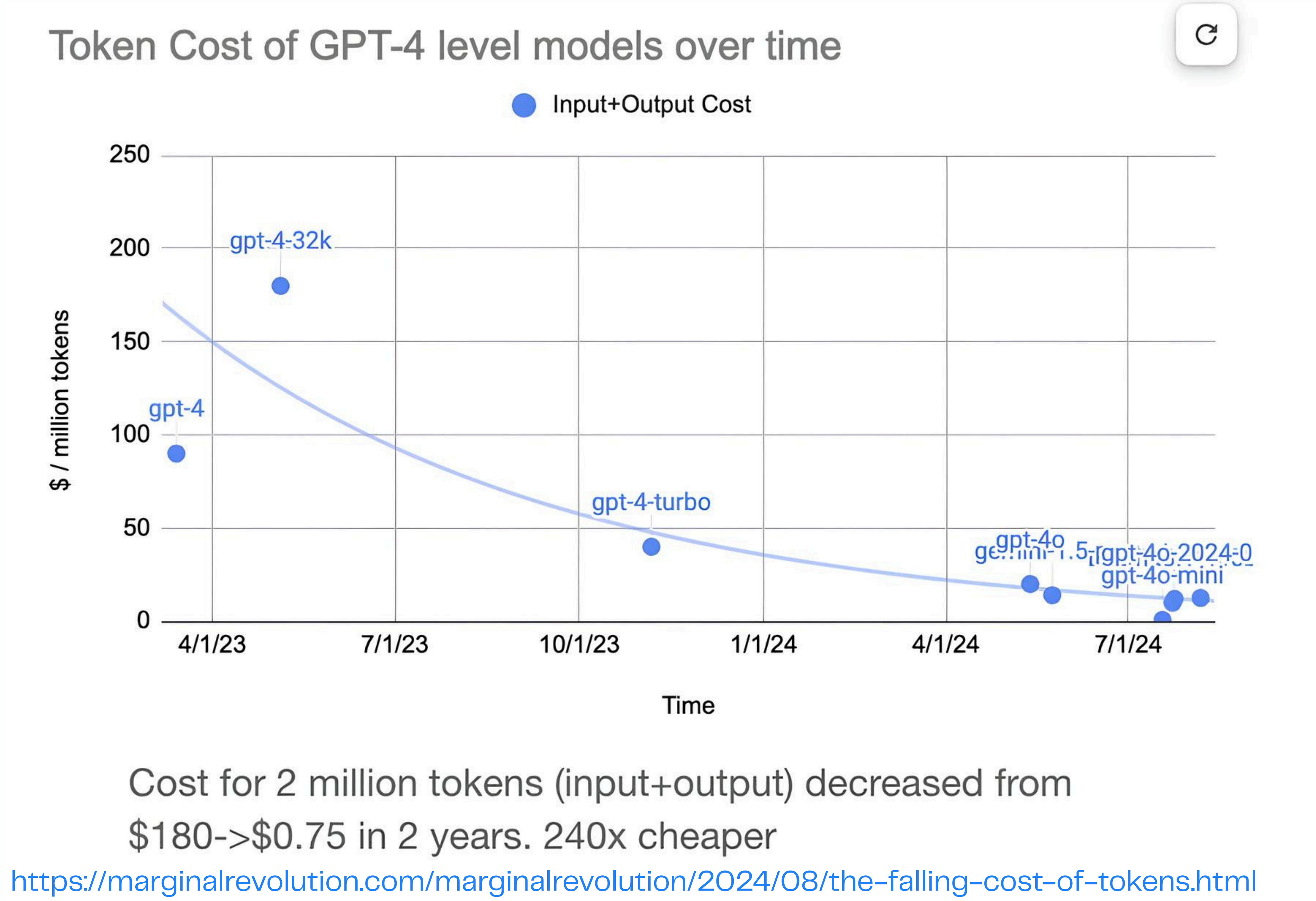
- We're going to show how to plug in our Tools as an MCP server into Claude and Cursor
- You can get Claude desktop at: https://claude.ai/download
- You can get Cursor at: https://www.cursor.com/
- You can get an MCP server with our tools from: https://github.com/entropic-digital/bioinformatics-mcp-example/

AGENT SECURITY CHALLENGES

- Prompt injection
- Sandboxed Environment
- Privilege escalation
- Model extraction



LOWER COSTS, MORE APPLICATIONS



Token Cost of GPT-4 level models over time

Cost for 2 million tokens (input+output) decreased from $180->$0.75 in 2 years. 240x cheaper

https://marginalrevolution.com/marginalrevolution/2024/08/the-falling-cost-of-tokens.html

THANK YOU FOR JOINING!

- Takeaway:
  - **Keep up to date with the latest tools in AI**

- For any questions you can contact us:
  - dionizije.fa@entropic.digital
  - mateo.cupic@entropic.digital
  - bruno.pandza@entropic.digital

- Acknowledgments:
  - Thanks to the ISCB Africa organizers for hosting this event
  - Special thanks to all participants for your attention and curiosity