How Parameters Are Handled in Functions Decorated with @function_tool?

✅ What is function_tool?
The @function_tool decorator from the OpenAI Agent SDK is used to turn a regular Python function into a usable tool for the AI agent. This lets the agent call the function automatically when it determines the function can help answer the user's request.

How Parameters Are Handled:
+++++++++++++++++++++++++

* 1-Type Hints Are Required
You must define the data type of each parameter (str, int, bool, etc.). This allows the Agent SDK to generate a JSON Schema from your function, which helps the LLM know what kind of input is expected.

* 2-LLM Automatically Fills Parameters

Based on the user’s message, the LLM tries to map the message to your function and auto-fill the parameters.

Example:

User says: “I want a red t-shirt under 1000 PKR”
→ The LLM calls: search_products(color="red", max_price=1000)

* 3-Parameter Validation:

The SDK automatically checks the inputs before calling your function. If the parameters are missing or incorrect, the function won't run, and the agent might ask the user for clarification.

How @function_tool Works Under the Hood
==

1. 🧠 Converts Your Function to JSON Schema
===
When you write:
****

python
++++++

```
@function_tool
def search_products(category: str, max_price: int):
    ...
```
The SDK turns it into:

json
++++
```
{
  "name": "search_products",
  "description": "Search for products based on category and/or price
limit.",
  "parameters": {
    "type": "object",
    "properties": {
      "category": { "type": "string" },
      "max_price": { "type": "integer" }
    },
    "required": ["category", "max_price"]
  }
}
```
This helps the LLM know:

What the function does

What parameters are required

What kind of input to expect

2. ✅ Auto Validation

Before calling the function, the Agent SDK checks:

Are all required inputs provided?

Are the values of correct types?

If anything’s missing or invalid, it can ask the user again.

3. 🔄 LLM → Tool Execution

The flow looks like this:

sql
+++
User Query → LLM interprets → Matches Function → Fills Parameters → Calls Tool → Returns Output
So, if the user says:

“Find me electronics under 3000”

The LLM figures out:

```
search_products(category="electronics", max_price=3000)
```