# Stream whales

Information about whales — an example of streamed structured response validation.

Demonstrates:

- streaming structured responses

This script streams structured responses from GPT-4 about whales, validates the data and displays it as a dynamic table using `rich` as the data is received.

## Running the Example

With dependencies installed and environment variables set, run:

**pip**

```
python -m pydantic_ai_examples.stream_whales
```

**uv**

```
uv run -m pydantic_ai_examples.stream_whales
```

Should give an output like this:

## Example Code

**stream_whales.py**

```python
from typing import Annotated, NotRequired, TypedDict

import devtools
import logfire
from pydantic import Field, ValidationError
from rich.console import Console
from rich.live import Live
from rich.table import Table

from pydantic_ai import Agent

# 'if-token-present' means nothing will be sent (and the example will work) if you don't have logfire configured
logfire.configure(send_to_logfire='if-token-present')


class Whale(TypedDict):
    name: str
    length: Annotated[
        float, Field(description='Average length of an adult whale in meters.')
    ]
    weight: NotRequired[
        Annotated[
            float,
            Field(description='Average weight of an adult whale in kilograms.', ge=50),
        ]
    ]
    ocean: NotRequired[str]
    description: NotRequired[Annotated[str, Field(description='Short Description')]]
```

```python
agent = Agent('openai:gpt-4', result_type=list[Whale])


def check_validation_error(e: ValidationError) -> bool:
    devtools.debug(e.errors())
    return False


async def main():
    console = Console()
    with Live('\n' * 36, console=console) as live:
        console.print('Requesting data...', style='cyan')
        async with agent.run_stream(
            'Generate me details of 5 species of Whale.'
        ) as result:
            console.print('Response:', style='green')

            async for message, last in result.stream_structured(debounce_by=0.01):
                try:
                    whales = await result.validate_structured_result(
                        message, allow_partial=not last
                    )
                except ValidationError as exc:
                    if all(
                        e['type'] == 'missing' and e['loc'] == ('response',)
                        for e in exc.errors()
                    ):
                        continue
                    else:
                        raise

                table = Table(
                    title='Species of Whale',
                    caption='Streaming Structured responses from GPT-4',
                    width=120,
                )
                table.add_column('ID', justify='right')
                table.add_column('Name')
                table.add_column('Avg. Length (m)', justify='right')
                table.add_column('Avg. Weight (kg)', justify='right')
                table.add_column('Ocean')
                table.add_column('Description', justify='right')

                for wid, whale in enumerate(whales, start=1):
                    table.add_row(
                        str(wid),
                        whale['name'],
                        f'{whale["length"]:0.0f}',
                        f'{w:0.0f}' if (w := whale.get('weight')) else '…',
                        whale.get('ocean') or '…',
                        whale.get('description') or '…',
                    )
                live.update(table)


if __name__ == '__main__':
    import asyncio

    asyncio.run(main())
```