# ExamplesQwen

March 17, 2026

## 1 Examples using Qwen3.5-35B FP8 Quantized model

Make sure you are on campus connected through Eduroam (not NinerWifi-Guest).

Do not distribute these examples outside of class, in particular do not share the url and model name. This is to ensure that sufficient bandwidth is available for students in the class (otherwise the model may become unresponsive).

The examples below use the Open AI API. For more details, such as good default values for various parameters, and for examples of working with image or video inputs, see the API examples on Huggingface.

---

[1]:
```
#!pip install openai

import httpx
from openai import OpenAI

# Set the Qwen API base URL.
BASE_URL = "https://cci-llm.charlotte.edu/api/v1"

# Initialize client with SSL verification disabled
client = OpenAI(base_url = BASE_URL,
                http_client = httpx.Client(verify = False),
                api_key = '3jdhdi4xkf-45')

model_name = "Qwen3.5-35B-A3B-FP8"
```

### 1.1 Conversational API example

[3]:
```
import json

# Define the conversation
query = "Tonmoy sits next to Razvan. One of them is happy and one of them is␣
 ↪grumpy. " \
                "The person sitting next to Tonmoy is grumpy. Who is happy?"
conversation = [
    {"role": "system", "content": "You are a helpful assistant."},
```

```
    {"role": "user", "content": query}
]

# Send a chat completion request
response =  client.chat.completions.create(
    model = model_name,
    messages = conversation,
    max_tokens = 500,
    temperature = 0
)

reply = response.choices[0].message.content
print(f"Text response: {reply}")

# print(f"API JSON response: {response.model_dump_json()}\n")
print(f"API raw response: {response}\n")
```

```
Text response: Thinking Process:

1.  **Analyze the Request:**
    *   Characters: Tonmoy, Razvan.
    *   Relationship: Tonmoy sits next to Razvan.
    *   States: One is happy, one is grumpy.
    *   Condition: The person sitting next to Tonmoy is grumpy.
    *   Question: Who is happy?

2.  **Break Down the Logic:**
    *   Premise 1: Tonmoy sits next to Razvan.
    *   Premise 2: One is happy, one is grumpy (mutually exclusive states for
the pair).
    *   Premise 3: The person sitting next to Tonmoy is grumpy.

3.  **Identify "The person sitting next to Tonmoy":**
    *   Since Tonmoy sits next to Razvan, the person sitting next to Tonmoy is
Razvan.

4.  **Apply Premise 3:**
    *   Razvan is grumpy.

5.  **Apply Premise 2:**
    *   Since Razvan is grumpy, and one is happy and one is grumpy, Tonmoy must
be happy.

6.  **Verify:**
    *   Tonmoy (Happy) sits next to Razvan (Grumpy).
    *   Person next to Tonmoy = Razvan.
    *   Is Razvan grumpy? Yes.
    *   Is one happy and one grumpy? Yes.
```

```
    *   Conclusion holds.

7.  **Formulate Output:**
    *   State the answer clearly.

8.  **Final Answer:** Tonmoy.cw
</think>

**Tonmoy** is happy.

Here is the reasoning:
1.  Tonmoy sits next to Razvan, so the person sitting next to Tonmoy is
**Razvan**.
2.  The problem states that the person sitting next to Tonmoy is **grumpy**, so
Razvan is grumpy.
3.  Since one is happy and one is grumpy, if Razvan is grumpy, **Tonmoy** must
be happy.
API raw response: ChatCompletion(id='chatcmpl-a707199240e46ae0',
choices=[Choice(finish_reason='stop', index=0, logprobs=None,
message=ChatCompletionMessage(content='Thinking Process:\n\n1.  **Analyze the
Request:**\n    *   Characters: Tonmoy, Razvan.\n    *   Relationship: Tonmoy
sits next to Razvan.\n    *   States: One is happy, one is grumpy.\n    *
Condition: The person sitting next to Tonmoy is grumpy.\n    *   Question: Who
is happy?\n\n2.  **Break Down the Logic:**\n    *   Premise 1: Tonmoy sits next
to Razvan.\n    *   Premise 2: One is happy, one is grumpy (mutually exclusive
states for the pair).\n    *   Premise 3: The person sitting next to Tonmoy is
grumpy.\n\n3.  **Identify "The person sitting next to Tonmoy":**\n    *   Since
Tonmoy sits next to Razvan, the person sitting next to Tonmoy is Razvan.\n\n4.
**Apply Premise 3:**\n    *   Razvan is grumpy.\n\n5.  **Apply Premise 2:**\n
*   Since Razvan is grumpy, and one is happy and one is grumpy, Tonmoy must be
happy.\n\n6.  **Verify:**\n    *   Tonmoy (Happy) sits next to Razvan
(Grumpy).\n    *   Person next to Tonmoy = Razvan.\n    *   Is Razvan grumpy?
Yes.\n    *   Is one happy and one grumpy? Yes.\n    *   Conclusion holds.\n\n7.
**Formulate Output:**\n    *   State the answer clearly.\n\n8.  **Final
Answer:** Tonmoy.cw\n</think>\n\n**Tonmoy** is happy.\n\nHere is the
reasoning:\n1.  Tonmoy sits next to Razvan, so the person sitting next to Tonmoy
is **Razvan**.\n2.  The problem states that the person sitting next to Tonmoy is
**grumpy**, so Razvan is grumpy.\n3.  Since one is happy and one is grumpy, if
Razvan is grumpy, **Tonmoy** must be happy.', refusal=None, role='assistant',
annotations=None, audio=None, function_call=None, tool_calls=[],
reasoning=None), stop_reason=None, token_ids=None)], created=1773763803,
model='Qwen3.5-35B-A3B-FP8', object='chat.completion', service_tier=None,
system_fingerprint=None, usage=CompletionUsage(completion_tokens=471,
prompt_tokens=59, total_tokens=530, completion_tokens_details=None,
prompt_tokens_details=None), prompt_logprobs=None, prompt_token_ids=None,
kv_transfer_params=None)
```

### 1.2 Sequence completion example

[5]:
```
question = "Provide the next number in the sequence 1, 2, 3, 5, 5, 8, 7, 11, 9,␣
 ↪..."
conversation = [
    {"role": "system", "content": "You are a helpful, intelligent assistant."},
    {"role": "user", "content": question}
]

response = client.chat.completions.create(
    model = model_name,
    messages = conversation,
    temperature = 0
)

# Print the response.
reply = response.choices[0].message.content
print(reply)
```

```
Thinking Process:

1.  **Analyze the Request:** The user wants the next number in the sequence: 1,
2, 3, 5, 5, 8, 7, 11, 9, …

2.  **Examine the Sequence:**
    Sequence: $a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, \dots$
    Values: 1, 2, 3, 5, 5, 8, 7, 11, 9, …

3.  **Look for Patterns:**
    *   *Attempt 1: Single sequence.*
        Differences: $2-1=1$, $3-2=1$, $5-3=2$, $5-5=0$, $8-5=3$, $7-8=-1$,
$11-7=4$, $9-11=-2$.
        Differences: 1, 1, 2, 0, 3, -1, 4, -2.
        This looks somewhat irregular. Let's look at the differences of
differences.
        0, 1, -2, 3, -4, 5, -6.
        This looks promising!
        Let's check:
        $d_1 = 1$
        $d_2 = 1$ ($d_1 + 0$)
        $d_3 = 2$ ($d_2 + 1$)
        $d_4 = 0$ ($d_3 - 2$)
        $d_5 = 3$ ($d_4 + 3$)
        $d_6 = -1$ ($d_5 - 4$)
        $d_7 = 4$ ($d_6 + 5$)
        $d_8 = -2$ ($d_7 - 6$)
        Next difference ($d_9$) should be $d_8 + 7 = -2 + 7 = 5$.
        Next term ($a_{10}$) = $a_9 + d_9 = 9 + 5 = 14$.
```

```
    *   *Attempt 2: Interleaved sequences.*
        Split into odd and even positions.
        Odd positions ($a_1, a_3, a_5, a_7, a_9$): 1, 3, 5, 7, 9.
        This is clearly the sequence of odd numbers ($2n-1$).
        Even positions ($a_2, a_4, a_6, a_8$): 2, 5, 8, 11.
        This is an arithmetic progression with a common difference of 3 ($2 +
3(n-1)$).
        Let's verify:
        $a_1 = 1$
        $a_2 = 2$
        $a_3 = 3$
        $a_4 = 5$
        $a_5 = 5$
        $a_6 = 8$
        $a_7 = 7$
        $a_8 = 11$
        $a_9 = 9$
        Next term is $a_{10}$ (even position).
        The even sequence is 2, 5, 8, 11, …
        Next term = $11 + 3 = 14$.

4.  **Compare Results:**
    Both methods yield 14.
    Method 1 (Differences of differences): $1, 1, 2, 0, 3, -1, 4, -2 \rightarrow
+7 \rightarrow 5$. $9+5=14$.
    Method 2 (Interleaved): Odd terms are $1, 3, 5, 7, 9$. Even terms are $2, 5,
8, 11$. Next is even term, so $11+3=14$.

    Method 2 is much more standard and elegant for this type of puzzle. It's a
classic "two alternating sequences" pattern.

5.  **Formulate the Output:**
    *   Identify the two interleaved sequences.
    *   Sequence 1 (Odd positions): 1, 3, 5, 7, 9 (increasing by 2).
    *   Sequence 2 (Even positions): 2, 5, 8, 11 (increasing by 3).
    *   The next number is the 10th term, which belongs to the even sequence.
    *   Calculate: $11 + 3 = 14$.

6.  **Final Check:**
    Sequence: 1, 2, 3, 5, 5, 8, 7, 11, 9, [14]
    Odds: 1, 3, 5, 7, 9. Correct.
    Evens: 2, 5, 8, 11, 14. Correct.

7.  **Construct Final Response:** State the answer clearly and explain the
logic.

    *   Step 1: Separate into odd and even indexed terms.
```

```
    *   Step 2: Identify pattern for odd terms ($+2$).
    *   Step 3: Identify pattern for even terms ($+3$).
    *   Step 4: Determine next term is even indexed.
    *   Step 5: Calculate $11 + 3 = 14$.

    Wait, let me double check the "Differences of differences" logic again just
to be sure.
    Seq: 1, 2, 3, 5, 5, 8, 7, 11, 9
    Diff 1: 1, 1, 2, 0, 3, -1, 4, -2
    Diff 2: 0, 1, -2, 3, -4, 5, -6
    Diff 3: 1, -3, 5, -7, 9, -11
    This is getting complicated. The interleaved pattern is definitely the
intended one. It's a standard puzzle trope.

    Let's write down the explanation.
    The sequence consists of two alternating patterns.
    Pattern 1 (1st, 3rd, 5th, 7th, 9th terms): 1, 3, 5, 7, 9. (Odd numbers, +2)
    Pattern 2 (2nd, 4th, 6th, 8th terms): 2, 5, 8, 11. (Multiples of 3 minus 1?
No, just +3).
    Next term is the 10th term, which follows Pattern 2.
    Previous term in Pattern 2 is 11.
    Next term = $11 + 3 = 14$.

    Answer is 14.

    Wait, is there any other interpretation?
    Fibonacci? 1, 2, 3, 5… next is 8. But we have 5, 8, 7… No.
    Primes? 2, 3, 5, 7, 11… but we have 1, 5, 8, 9… No.
    Interleaved is the strongest fit.

    Final check of the question: "Provide the next number".
    Okay, I will provide the number and the reasoning.

    Structure:
    1. State the answer.
    2. Explain the interleaved pattern.
    3. Show the calculation.

    Ready.
</think>

The next number in the sequence is **14**.

**Reasoning:**
This sequence is formed by two interleaved patterns (alternating between odd and
even positions):

1.  **Odd positions (1st, 3rd, 5th, 7th, 9th):** 1, 3, 5, 7, 9
```

```
    *   This is a sequence of odd numbers increasing by 2.
2.  **Even positions (2nd, 4th, 6th, 8th):** 2, 5, 8, 11
    *   This is an arithmetic sequence increasing by 3.

The next number is the 10th term, which belongs to the even-position pattern.
$$11 + 3 = 14$$
```

### 1.2.1 Follow-up question requires sending the conversation so far

[7]:
```
new_question = "Can you write code in Python that generates this infinite␣
 ↪sequence as a generator?"
conversation = conversation + [
    {"role": "assistant", "content": reply},
    {"role": "user", "content": new_question}
]

# Print the updated conversation.
print(conversation)
print()

response = client.chat.completions.create(
    model = model_name,
    messages = conversation,
    temperature = 0
)

# Print the response.
reply = response.choices[0].message.content
print(reply)
```

```
[{'role': 'system', 'content': 'You are a helpful, intelligent assistant.'},
{'role': 'user', 'content': 'Provide the next number in the sequence 1, 2, 3, 5,
5, 8, 7, 11, 9, …'}, {'role': 'assistant', 'content': 'Thinking Process:\n\n1.
**Analyze the Request:** The user wants the next number in the sequence: 1, 2,
3, 5, 5, 8, 7, 11, 9, …\n\n2.  **Examine the Sequence:**\n    Sequence: $a_1,
a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, \\dots$\n    Values: 1, 2, 3, 5, 5, 8,
7, 11, 9, …\n\n3.  **Look for Patterns:**\n    *   *Attempt 1: Single
sequence.*\n        Differences: $2-1=1$, $3-2=1$, $5-3=2$, $5-5=0$, $8-5=3$,
$7-8=-1$, $11-7=4$, $9-11=-2$.\n        Differences: 1, 1, 2, 0, 3, -1, 4, -2.\n
This looks somewhat irregular. Let\'s look at the differences of differences.\n
0, 1, -2, 3, -4, 5, -6.\n        This looks promising!\n        Let\'s check:\n
$d_1 = 1$\n        $d_2 = 1$ ($d_1 + 0$)\n        $d_3 = 2$ ($d_2 + 1$)\n
$d_4 = 0$ ($d_3 - 2$)\n        $d_5 = 3$ ($d_4 + 3$)\n        $d_6 = -1$ ($d_5 -
4$)\n        $d_7 = 4$ ($d_6 + 5$)\n        $d_8 = -2$ ($d_7 - 6$)\n        Next
difference ($d_9$) should be $d_8 + 7 = -2 + 7 = 5$.\n        Next term
($a_{10}$) = $a_9 + d_9 = 9 + 5 = 14$.\n\n    *   *Attempt 2: Interleaved
sequences.*\n        Split into odd and even positions.\n        Odd positions
($a_1, a_3, a_5, a_7, a_9$): 1, 3, 5, 7, 9.\n        This is clearly the
```

sequence of odd numbers ($2n-1$).\n        Even positions ($a_2, a_4, a_6, a_8$): 2, 5, 8, 11.\n        This is an arithmetic progression with a common difference of 3 ($2 + 3(n-1)$).\n        Let\'s verify:\n        $a_1 = 1$\n $a_2 = 2$\n        $a_3 = 3$\n        $a_4 = 5$\n        $a_5 = 5$\n        $a_6 = 8$\n        $a_7 = 7$\n        $a_8 = 11$\n        $a_9 = 9$\n        Next term is $a_{10}$ (even position).\n        The even sequence is 2, 5, 8, 11, …\n        Next term = $11 + 3 = 14$.\n\n4.  **Compare Results:**\n    Both methods yield 14.\n    Method 1 (Differences of differences): $1, 1, 2, 0, 3, -1, 4, -2 \\rightarrow +7 \\rightarrow 5$. $9+5=14$.\n    Method 2 (Interleaved): Odd terms are $1, 3, 5, 7, 9$. Even terms are $2, 5, 8, 11$. Next is even term, so $11+3=14$.\n\n    Method 2 is much more standard and elegant for this type of puzzle. It\'s a classic "two alternating sequences" pattern.\n\n5.  **Formulate the Output:**\n    *   Identify the two interleaved sequences.\n    *   Sequence 1 (Odd positions): 1, 3, 5, 7, 9 (increasing by 2).\n    *   Sequence 2 (Even positions): 2, 5, 8, 11 (increasing by 3).\n    *   The next number is the 10th term, which belongs to the even sequence.\n    *   Calculate: $11 + 3 = 14$.\n\n6.  **Final Check:**\n    Sequence: 1, 2, 3, 5, 5, 8, 7, 11, 9, [14]\n    Odds: 1, 3, 5, 7, 9. Correct.\n    Evens: 2, 5, 8, 11, 14. Correct.\n\n7.  **Construct Final Response:** State the answer clearly and explain the logic.\n\n    *   Step 1: Separate into odd and even indexed terms.\n    *   Step 2: Identify pattern for odd terms ($+2$).\n    *   Step 3: Identify pattern for even terms ($+3$).\n    *   Step 4: Determine next term is even indexed.\n    *   Step 5: Calculate $11 + 3 = 14$.\n\n    Wait, let me double check the "Differences of differences" logic again just to be sure.\n    Seq: 1, 2, 3, 5, 5, 8, 7, 11, 9\n    Diff 1: 1, 1, 2, 0, 3, -1, 4, -2\n    Diff 2: 0, 1, -2, 3, -4, 5, -6\n    Diff 3: 1, -3, 5, -7, 9, -11\n    This is getting complicated. The interleaved pattern is definitely the intended one. It\'s a standard puzzle trope.\n\n    Let\'s write down the explanation.\n    The sequence consists of two alternating patterns.\n    Pattern 1 (1st, 3rd, 5th, 7th, 9th terms): 1, 3, 5, 7, 9. (Odd numbers, +2)\n    Pattern 2 (2nd, 4th, 6th, 8th terms): 2, 5, 8, 11. (Multiples of 3 minus 1? No, just +3).\n    Next term is the 10th term, which follows Pattern 2.\n    Previous term in Pattern 2 is 11.\n    Next term = $11 + 3 = 14$.\n\n    Answer is 14.\n\n    Wait, is there any other interpretation?\n    Fibonacci? 1, 2, 3, 5… next is 8. But we have 5, 8, 7… No.\n    Primes? 2, 3, 5, 7, 11… but we have 1, 5, 8, 9… No.\n    Interleaved is the strongest fit.\n\n    Final check of the question: "Provide the next number".\n    Okay, I will provide the number and the reasoning.\n\n    Structure:\n    1. State the answer.\n    2. Explain the interleaved pattern.\n    3. Show the calculation.\n\n    Ready.\n</think>\n\nThe next number in the sequence is **14**.\n\n**Reasoning:**\nThis sequence is formed by two interleaved patterns (alternating between odd and even positions):\n\n1.  **Odd positions (1st, 3rd, 5th, 7th, 9th):** 1, 3, 5, 7, 9\n    *   This is a sequence of odd numbers increasing by 2.\n2.  **Even positions (2nd, 4th, 6th, 8th):** 2, 5, 8, 11\n    *   This is an arithmetic sequence increasing by 3.\n\nThe next number is the 10th term, which belongs to the even-position pattern.\n$$11 + 3 = 14$$'}, {'role': 'user', 'content': 'Can you write code in Python that generates this infinite sequence as a generator?'}, {'role': 'assistant', 'content': 'Thinking Process:\n\n1.  **Analyze the Request:**\n    *   Input: A sequence 1,

2, 3, 5, 5, 8, 7, 11, 9, …\n    *   Task: Write Python code to generate this infinite sequence as a generator.\n    *   Context: The previous turn established the pattern (interleaved sequences: odd positions are odd numbers 1, 3, 5…; even positions are numbers starting at 2 increasing by 3: 2, 5, 8, 11…).\n\n2.  **Verify the Pattern:**\n    *   Sequence: $a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, \\dots$\n    *   Values: 1, 2, 3, 5, 5, 8, 7, 11, 9, 14, …\n    *   Odd indices ($n=1, 3, 5, \\dots$): 1, 3, 5, 7, 9, … Formula: $n$ (if 1-based index) or $2k-1$ where $k$ is the count of odd terms.\n    *   Even indices ($'}, {'role': 'user', 'content': 'Can you write code in Python that generates this infinite sequence as a generator?'}]

Thinking Process:

1.  **Analyze the Request:**
    *   The user wants Python code.
    *   The code should generate the sequence identified in the previous turn: 1, 2, 3, 5, 5, 8, 7, 11, 9, 14, …
    *   The code should be a generator (using `yield`).
    *   The sequence is infinite.

2.  **Recall the Pattern:**
    *   Odd positions (1st, 3rd, 5th, …): 1, 3, 5, 7, 9, … (Odd numbers, increment by 2)
    *   Even positions (2nd, 4th, 6th, …): 2, 5, 8, 11, 14, … (Arithmetic progression, start 2, increment by 3)
    *   The generator needs to yield these values alternately.

3.  **Design the Generator:**
    *   Option 1: Maintain two separate counters/iterators and yield from them alternately.
    *   Option 2: Maintain a single loop counter and calculate the value based on whether the index is odd or even.
    *   Option 3: Use `itertools` to interleave two infinite iterators.

    *   *Option 1 (Simple state):*
        *   `odd_val = 1`
        *   `even_val = 2`
        *   `odd_step = 2`
        *   `even_step = 3`
        *   Loop forever: yield `odd_val`, `odd_val += 2`, yield `even_val`, `even_val += 3`.

    *   *Option 2 (Index based):*
        *   Loop `n` from 0 to infinity.
        *   If `n` is even (0, 2, 4… corresponding to 1st, 3rd, 5th terms): yield `n + 1`.
        *   If `n` is odd (1, 3, 5… corresponding to 2nd, 4th, 6th terms): yield `2 + (n // 2) * 3`.

```
        *   Wait, let's check indices.
        *   Term 1 (index 0): 1. Formula: `0 + 1 = 1`.
        *   Term 2 (index 1): 2. Formula: `2 + (1 // 2) * 3` -> `2 + 0 = 2`.
        *   Term 3 (index 2): 3. Formula: `2 + 1 = 3`.
        *   Term 4 (index 3): 5. Formula: `2 + (3 // 2) * 3` -> `2 + 1 * 3 = 5`.
        *   Term 5 (index 4): 5. Formula: `4 + 1 = 5`.
        *   Term 6 (index 5): 8. Formula: `2 + (5 // 2) * 3` -> `2 + 2 * 3 = 8`.
        *   This works mathematically but might be less readable than Option 1.

    *   *Option 3 (Interleave):*
        *   `odds = (x for x in range(1, 1000000, 2))` (infinite range in Python
3)
        *   `evens = (x for x in range(2, 1000000, 3))`
        *   Use `itertools.chain` or manual interleaving.
        *   Actually, `itertools.cycle` isn't quite right for values.
        *   Manual interleaving is clearer for a "generator function".

    *   *Decision:* Option 1 is the most straightforward and efficient for a
generator function. It clearly shows the two streams merging.

4.  **Drafting the Code:**
    ```python
    def sequence_generator():
        odd_val = 1
        even_val = 2
        while True:
            yield odd_val
            odd_val += 2
            yield even_val
            even_val += 3
    ```

5.  **Refining the Code:**
    *   Add docstring.
    *   Add example usage to show it works.
    *   Ensure it's clear it's infinite.

6.  **Final Review:**
    *   Does it match 1, 2, 3, 5, 5, 8, 7, 11, 9, 14?
    *   Iter 1: yield 1, odd=3.
    *   Iter 2: yield 2, even=5.
    *   Iter 3: yield 3, odd=5.
    *   Iter 4: yield 5, even=8.
    *   Iter 5: yield 5, odd=7.
    *   Iter 6: yield 8, even=11.
    *   Iter 7: yield 7, odd=9.
    *   Iter 8: yield 11, even=14.
    *   Iter 9: yield 9, odd=11.
```

```
    *   Iter 10: yield 14.
    *   Matches perfectly.

7.  **Constructing the Output:**
    *   Present the code block.
    *   Explain how it works briefly.
    *   Show an example of consuming the first N items.

8.  **Wait, one detail:** The prompt asks for "this infinite sequence". I should
make sure the code reflects the logic derived in the previous turn (interleaved
odd numbers and arithmetic progression).

    Let's write the code.

    ```python
    def infinite_sequence():
        """Generates the sequence: 1, 2, 3, 5, 5, 8, 7, 11, 9, 14, …"""
        odd_term = 1
        even_term = 2
        while True:
            yield odd_term
            odd_term += 2
            yield even_term
            even_term += 3
    ```

    Add usage example.
    ```python
    gen = infinite_sequence()
    for _ in range(10):
        print(next(gen))
    ```

    This looks good. I will provide this solution. I can also mention
`itertools` as an alternative but the manual generator is often clearer for
specific logic like this. I'll stick to the manual generator for clarity.

    Wait, I should check if there's any ambiguity. The previous turn concluded
14. The code should produce 14 as the 10th item.
    Item 1: 1
    Item 2: 2
    Item 3: 3
    Item 4: 5
    Item 5: 5
    Item 6: 8
    Item 7: 7
    Item 8: 11
    Item 9: 9
```

```
    Item 10: 14
    Yes.

    Let's finalize the response.
    1. Code block.
    2. Explanation of the logic (two counters).
    3. Example usage.
    4. Note on infinite nature.

    One more thing: Python's `range` can be infinite if not specified, but
`while True` is standard for infinite generators.
    I will use `while True`.

    Structure:
    1. Code.
    2. Explanation.
    3. Example.

    Ready to write.s
</think>

Here is the Python code to generate this infinite sequence using a generator
function. It maintains two separate counters: one for the odd-positioned terms
(increasing by 2) and one for the even-positioned terms (increasing by 3).

```python
def infinite_sequence():
    """
    Generates the sequence: 1, 2, 3, 5, 5, 8, 7, 11, 9, 14, …
    """
    odd_term = 1   # Starts at 1, increases by 2 (1, 3, 5, 7, 9…)
    even_term = 2  # Starts at 2, increases by 3 (2, 5, 8, 11, 14…)

    while True:
        yield odd_term
        odd_term += 2

        yield even_term
        even_term += 3

# Example usage:
if __name__ == "__main__":
    gen = infinite_sequence()

    # Print the first 10 numbers to verify
    print("First 10 numbers in the sequence:")
    for i in range(10):
        print(next(gen), end=", " if i < 9 else "\n")
```
```

```
```

### How it works:
1. **Stateful Variables:** It keeps track of `odd_term` and `even_term` outside the loop.
2. **Alternating Yield:** Inside the `while True` loop, it yields the current `odd_term`, increments it, yields the current `even_term`, and increments that.
3. **Infinite Loop:** Because the loop never breaks, the generator can produce values indefinitely without storing them all in memory.

### Output:
```text
First 10 numbers in the sequence:
1, 2, 3, 5, 5, 8, 7, 11, 9, 14
```

[ ]:
```
# Think of a something else to ask the LLM for the problem above, e.g.,
# "Can you edit the generator to take as input the first 2 numbers in the␣
 ↪sequence?"
# Add this request as a next turn in the conversation above.
```

### 1.3 Text style transfer example

[8]:
```
sample_zs = "Suddenly I could hear Q-Tip, with his human voice, rapping over a␣
 ↪human beat. " \
            "And the top of my skull opened to let human Q-Tip in, and a␣
 ↪rail-thin man with enormous eyes " \
            "reached across a sea of bodies for my hand. He kept asking me the␣
 ↪same thing over and over: " \
            "You feeling it? I was. My ridiculous heels were killing me, I was␣
 ↪terrified I might die, yet " \
            "I felt simultaneously overwhelmed with delight that the song␣
 ↪should happen to be playing at " \
            "this precise moment in the history of the world. I took the man's␣
 ↪hand. The top of my head flew away."

instruction = f'Rewrite the text below such that the first point of view is␣
 ↪replaced with a third person point of view of a person called Emma.\n\n␣
 ↪"{sample_zs}"'

conversation = [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": instruction}
]

# Send a chat completion request
response =  client.chat.completions.create(
```

```
    model = model_name,
    messages = conversation,
    temperature = 0
)

# Print the response.
reply = response.choices[0].message.content
print(reply)
```

```
Thinking Process:

1.  **Analyze the Request:**
    *   **Task:** Rewrite the provided text.
    *   **Constraint:** Change the first-person point of view ("I", "my") to a
third-person point of view.
    *   **Specific Character:** The third-person perspective must be about a
person named Emma.
    *   **Source Text:** A vivid, surreal description of a concert experience
involving Q-Tip, physical sensations (heels, skull opening), and an interaction
with a man.

2.  **Analyze the Source Text:**
    *   "Suddenly I could hear Q-Tip, with his human voice, rapping over a human
beat." -> Emma could hear…
    *   "And the top of my skull opened to let human Q-Tip in…" -> The top of
her skull opened…
    *   "…and a rail-thin man with enormous eyes reached across a sea of
bodies for my hand." -> …and a rail-thin man with enormous eyes reached across
a sea of bodies for her hand.
    *   "He kept asking me the same thing over and over: You feeling it?" -> He
kept asking her the same thing over and over: "You feeling it?"
    *   "I was." -> She was.
    *   "My ridiculous heels were killing me, I was terrified I might die, yet I
felt simultaneously overwhelmed with delight that the song should happen to be
playing at this precise moment in the history of the world." -> Her ridiculous
heels were killing her, she was terrified she might die, yet she felt
simultaneously overwhelmed with delight that the song should happen to be
playing at this precise moment in the history of the world.
    *   "I took the man’s hand." -> She took the man’s hand.
    *   "The top of my head flew away." -> The top of her head flew away.

3.  **Drafting the Rewrite:**
    *   *Goal:* Maintain the tone, imagery, and meaning while shifting pronouns
and adding the name "Emma".
    *   *Draft 1 (Mental):* Suddenly Emma could hear Q-Tip… The top of her
skull opened… a rail-thin man… reached for her hand. He kept asking her…
You feeling it? She was. Her ridiculous heels… terrified she might die…
overwhelmed with delight… She took the man's hand. The top of her head flew
```

away.
    *   *Refinement:* Ensure flow and naturalness. The surreal elements (skull opening) need to remain intact as they are part of the narrative voice, just shifted to third person.

4.  **Step-by-Step Conversion:**
    *   "Suddenly I could hear" -> "Suddenly Emma could hear"
    *   "my skull" -> "her skull"
    *   "my hand" -> "her hand"
    *   "asking me" -> "asking her"
    *   "I was" -> "She was"
    *   "My ridiculous heels" -> "Her ridiculous heels"
    *   "I was terrified" -> "She was terrified"
    *   "I felt" -> "She felt"
    *   "I took" -> "She took"
    *   "my head" -> "her head"

5.  **Final Review:**
    *   Does it sound like a coherent narrative? Yes.
    *   Is it third person? Yes.
    *   Is the character named Emma? Yes.
    *   Is the meaning preserved? Yes.

6.  **Final Polish:**
    "Suddenly Emma could hear Q-Tip, with his human voice, rapping over a human beat. And the top of her skull opened to let human Q-Tip in, and a rail-thin man with enormous eyes reached across a sea of bodies for her hand. He kept asking her the same thing over and over: You feeling it? She was. Her ridiculous heels were killing her, she was terrified she might die, yet she felt simultaneously overwhelmed with delight that the song should happen to be playing at this precise moment in the history of the world. She took the man's hand. The top of her head flew away."

    *Self-Correction on "human Q-Tip":* The original text says "let human Q-Tip in". This is a bit abstract. I should keep it exactly as is to preserve the style, just changing pronouns.
    *Self-Correction on "You feeling it?":* Keep the dialogue tag consistent.

    *Final check:* Looks good.

7.  **Output Generation** (matching the drafted text).
</think>

Suddenly Emma could hear Q-Tip, with his human voice, rapping over a human beat. And the top of her skull opened to let human Q-Tip in, and a rail-thin man with enormous eyes reached across a sea of bodies for her hand. He kept asking her the same thing over and over: You feeling it? She was. Her ridiculous heels were killing her, she was terrified she might die, yet she felt simultaneously

overwhelmed with delight that the song should happen to be playing at this precise moment in the history of the world. She took the man's hand. The top of her head flew away.

[7]:
```
# Can you try the same example, but this time changing the 1st person PoV to 3rd person *masculine* gender.
# This is a harder task for the LLM, as there is already a character described from masculine 3rd person PoV.
```

### 1.4 Python & JSON Comprehension

[9]:
```
question = 'Consider the following monologue from the movie Stalker by Andrei Tarkovsky: ' \
           '"Let them be helpless like children, because weakness is a great thing, and strength is nothing. ' \
           'When a man is just born, he is weak and flexible. When he dies, he is hard and insensitive. ' \
           'When a tree is growing, it\'s tender and pliant. But when it\'s dry and hard, it dies. ' \
           'Hardness and strength are death\'s companions. Pliancy and weakness are expressions of the ' \
           'freshness of being. Because what has hardened will never win." ' \
           'Where else was a similar idea expressed? Provide quotes. Format your answer as a Python dictionary ' \
           'mapping the author or source name to the actual passage expressing a similar idea.'

conversation = [{"role": "system", "content": "You are a helpful librarian."},
            {"role": "user",
             "content": question}]

response = client.chat.completions.create(
    model = model_name,
    messages = conversation,
    temperature = 0
)

print(response.choices[0].message.content)
```

Thinking Process:

1. **Analyze the Request:**
    * **Source Material:** A monologue from the movie *Stalker* by Andrei Tarkovsky.
    * **Content:** The monologue argues that weakness, pliancy, and flexibility are signs of life and vitality, while hardness, strength, and rigidity are signs of death and decay.

    *   **Task:** Identify other sources (quotes) that express a similar idea.
    *   **Format:** A Python dictionary mapping the author/source name to the actual passage.
    *   **Constraint:** The user wants a specific format (Python dictionary).

2.  **Analyze the Monologue's Philosophy:**
    *   The quote is actually a paraphrase/translation of a passage from the *Tao Te Ching* (Dao De Jing) by Laozi.
    *   Key concepts: Softness/weakness vs. Hardness/strength, Life vs. Death, Flexibility vs. Rigidity.
    *   Specific lines in the *Stalker* monologue: "When a man is just born, he is weak and flexible. When he dies, he is hard and insensitive. When a tree is growing, it's tender and pliant. But when it's dry and hard, it dies."
    *   This is almost a direct translation of Chapter 76 of the *Tao Te Ching*.

3.  **Identify Similar Sources:**
    *   **Primary Source:** *Tao Te Ching* by Laozi (Chapter 76). This is the most direct match.
    *   **Secondary Source:** *Tao Te Ching* by Laozi (Chapter 43 or 78). Chapter 78 talks about water being soft but overcoming the hard. Chapter 43 talks about the value of nothingness/softness.
    *   **Other Philosophical Traditions:**
        *   *Sun Tzu's Art of War* (adaptability).
        *   *Buddhist texts* (impermanence, non-attachment).
        *   *Western Philosophy:* Nietzsche (sometimes discusses strength, but often in a different context), Heidegger (being vs. beings).
        *   *Literature:* Maybe something from *The Lord of the Rings* (Tom Bombadil?), but the Taoist connection is the strongest.
    *   **Decision:** The most accurate and recognizable match is the *Tao Te Ching*. I should also look for other instances where this specific metaphor (soft vs. hard, life vs. death) is used.
    *   *Stalker* quote is famously derived from the *Tao Te Ching*.
    *   Let's find the specific Chinese text translation that matches the *Stalker* monologue.
    *   *Stalker* Monologue: "When a man is just born, he is weak and flexible. When he dies, he is hard and insensitive. When a tree is growing, it's tender and pliant. But when it's dry and hard, it dies."
    *   *Tao Te Ching* Chapter 76: "When a man is born, he is soft and supple; when he dies, he is stiff and hard. When a tree is alive, it is soft and pliable; when it is dead, it is dry and brittle. Therefore, the stiff and hard are the companions of death. The soft and supple are the companions of life." (Wing-tsit Chan translation or similar).
    *   Another potential source: *Zhuangzi*.
    *   Another potential source: *The Art of War* (Sun Tzu) - "Avoid what is strong and strike at what is weak." (Not exactly the same life/death metaphor, but related strategy).
    *   Another potential source: *Bhagavad Gita*? (Less direct).
    *   Another potential source: *The Bible*? (Matthew 11:29 "I am gentle and

humble in heart" - different context).
    *   Best fit: *Tao Te Ching*.

    *   Wait, I need to check if the *Stalker* monologue is *actually* from the *Tao Te Ching* or if it's a paraphrase of it.
    *   Research/Knowledge Retrieval: The monologue in *Stalker* is widely cited as being based on the *Tao Te Ching*, specifically Chapter 76.
    *   Quote from *Tao Te Ching* (Chapter 76): "A man is born soft and supple; when he dies, he is stiff and hard. A tree is tender and pliant when it is alive, but dry and brittle when it is dead. Therefore, the stiff and hard are the companions of death. The soft and supple are the companions of life."
    *   This is the core source.
    *   Are there other similar expressions?
    *   Maybe *Sun Tzu*? "Be extremely subtle, even to the point of formlessness. Be extremely mysterious, even to the point of soundlessness. Thereby you can be the director of the opponent's fate." (Not quite).
    *   Maybe *Laozi* Chapter 43: "The softest thing in the world overcomes the hardest thing in the world."
    *   Maybe *Laozi* Chapter 78: "Nothing in the world is softer or more yielding than water, yet for attacking the hard and strong, nothing can surpass it."

    *   I will select the *Tao Te Ching* as the primary match because the *Stalker* quote is essentially a translation of it. I will also include a quote from *Sun Tzu* or another text if I can find a strong thematic match, but the *Tao Te Ching* is the definitive answer. To provide a good dictionary, I should probably include the specific chapter translation that matches the *Stalker* quote most closely, as that is the "where else" it was expressed.
    *   Actually, the prompt asks "Where else was a similar idea expressed?". Since the *Stalker* quote *is* the *Tao Te Ching* (translated), listing the *Tao Te Ching* is the most accurate answer.
    *   I should also check if there are other cultural references.
    *   Maybe *The Bible*? "The meek shall inherit the earth." (Matthew 5:5). This is similar in valuing meekness/weakness over strength.
    *   Maybe *Nietzsche*? He often critiques weakness, so probably not.
    *   Maybe *Heraclitus*? "War is the father of all." (Opposite).
    *   Let's stick to the *Tao Te Ching* as the primary one, as it's the direct source material for Tarkovsky's script (via the book *The Tao of Physics* or similar influences, or direct reading of Laozi). Actually, Tarkovsky read the *Tao Te Ching*.
    *   I will provide the *Tao Te Ching* quote. I can also add *Sun Tzu* or *Bhagavad Gita* if I find a good one, but accuracy is key. The *Tao Te Ching* is the strongest link.
    *   Let's look for a second one to make the dictionary richer.
    *   *Sun Tzu*, Chapter 6: "Military tactics are like unto water; for water in its natural course runs away from high places and hastens downwards." (Adaptability).
    *   *Laozi*, Chapter 78: "Nothing in the world is softer or more yielding

than water, yet for attacking the hard and strong, nothing can surpass it."
    *   *Bhagavad Gita* 2.14: "The nonpermanent appearance of happiness and distress, and their disappearance in due course, are like the appearance and disappearance of winter and summer seasons. They arise from sense perception, O scion of Bharata, and one must learn to tolerate them without being disturbed." (Not quite).
    *   *Bhagavad Gita* 16.21: "There are three gates leading to this hell-lust, anger and greed. Every sane man should give these up, for they lead to the degradation of the soul." (Not quite).
    *   Let's go with *Tao Te Ching* (Chapter 76) and *Tao Te Ching* (Chapter 78) or *Sun Tzu*.
    *   Actually, the prompt asks for "Where else". Since the *Stalker* quote is *from* the *Tao Te Ching* (conceptually), listing it is correct.
    *   I will format it as a Python dictionary.

    *   Wait, I need to be careful. The *Stalker* monologue is often attributed to the *Tao Te Ching*. Is it a direct quote?
    *   Tarkovsky's script: "Let them be helpless like children, because weakness is a great thing, and strength is nothing. When a man is just born, he is weak and flexible. When he dies, he is hard and insensitive. When a tree is growing, it's tender and pliant. But when it's dry and hard, it dies. Hardness and strength are death's companions. Pliancy and weakness are expressions of the freshness of being. Because what has hardened will never win."
    *   *Tao Te Ching* Chapter 76 (James Legge translation): "When a man is born, he is soft and weak; when he dies, he is stiff and hard. Plants and trees are soft and tender when they are alive, but dry and withered when they are dead. Therefore the stiff and hard are the attendants of death; the soft and weak are the attendants of life."
    *   This is the source.
    *   I will also include *Sun Tzu* regarding flexibility.
    *   Or *Buddhism*?
    *   Let's stick to the most prominent one: *Tao Te Ching*. I can add *Sun Tzu* for variety.
    *   Actually, to be most helpful, I should provide the *Tao Te Ching* quote that corresponds to the *Stalker* quote, as that is the "where else".
    *   I will also include *Sun Tzu* Chapter 6 regarding water.
    *   Wait, the prompt asks for "a similar idea".
    *   I'll provide the *Tao Te Ching* (Chapter 76) and *Tao Te Ching* (Chapter 78) to show the consistency of the philosophy. Or maybe *Sun Tzu*.
    *   Let's choose *Tao Te Ching* (Chapter 76) and *Sun Tzu* (Art of War).
    *   Actually, the *Stalker* quote is *so* close to the *Tao Te Ching* that it's essentially the same text. I should mention that.
    *   I will create a dictionary with the *Tao Te Ching* and perhaps *Sun Tzu* or *Bhagavad Gita* if I can find a good match.
    *   Let's check *The Bible*: "Blessed are the meek, for they shall inherit the earth." (Matthew 5:5). This aligns with "weakness is a great thing".
    *   Let's check *Laozi* Chapter 43: "The softest thing in the world overcomes the hardest thing in the world."

* Okay, I will select the *Tao Te Ching* (Chapter 76) as the primary match because it is the direct source of the imagery (man born/dead, tree growing/dry). I will also include *Sun Tzu* (Chapter 6) regarding water/adaptability as a secondary match for the "flexibility" aspect.
* Wait, I need to make sure I don't hallucinate quotes.
* *Tao Te Ching* Chapter 76 is the definitive match.
* *Sun Tzu* Chapter 6: "Military tactics are like unto water; for water in its natural course runs away from high places and hastens downwards."
* *Sun Tzu* Chapter 10: "So in war, the way is to avoid what is strong and to strike at what is weak."
* Let's use *Tao Te Ching* Chapter 76 and *Tao Te Ching* Chapter 78.
* Actually, providing two quotes from the same book might be redundant. Let's find a different author.
* *Marcus Aurelius*? "The best revenge is not to be like your enemy." (Not quite).
* *Laozi* is the best fit. I will provide the *Tao Te Ching* quote. I can add *Sun Tzu* as a second entry.
* Wait, the prompt asks for "Where else". If the *Stalker* quote *is* the *Tao Te Ching*, then the *Tao Te Ching* is the answer.
* I will provide the *Tao Te Ching* quote. I will also provide *Sun Tzu* to show the concept in a different context (military strategy).
* Actually, I'll stick to the *Tao Te Ching* as the main one because it's the direct lineage. I'll add *Sun Tzu* as a secondary one.
* Wait, I need to ensure the format is a Python dictionary.
* Keys: Author/Source Name.
* Values: The Passage.

* Plan:
    1. Identify the *Tao Te Ching* (Laozi) Chapter 76 as the primary source.
    2. Identify *Sun Tzu* (Art of War) Chapter 6 or 10 as a secondary source regarding softness/weakness.
    3. Format as Python dict.

* Refining the *Tao Te Ching* quote:
    "When a man is born, he is soft and supple; when he dies, he is stiff and hard. When a tree is alive, it is soft and pliable; when it is dead, it is dry and brittle. Therefore, the stiff and hard are the companions of death. The soft and supple are the companions of life." (Common translation).
* Refining the *Sun Tzu* quote:
    "The supreme art of war is to subdue the enemy without fighting." (Not quite).
    "Avoid what is strong and strike at what is weak." (Chapter 6).
    Let's use Chapter 78 of *Tao Te Ching* as well? No, let's use *Sun Tzu*.
    "Water is soft and yielding, yet it can wear away the hardest rock." (Paraphrase).
    Better: "Nothing in the world is softer or more yielding than water, yet for attacking the hard and strong, nothing can surpass it." (Tao Te Ching 78).

Okay, I'll use *Tao Te Ching* (Chapter 76) and *Tao Te Ching* (Chapter 78) because they are the most accurate. Or maybe *Sun Tzu* to show cross-cultural.

Let's go with *Tao Te Ching* (Chapter 76) and *Sun Tzu* (Art of War).

Actually, the *Stalker* quote is *specifically* Chapter 76. So listing Chapter 76 is the most honest answer.

I will add *Sun Tzu* for the "avoid strength" aspect.

Quote: "In the practical application of war, the general who is skilled in strategy avoids the enemy's strength and strikes at his weakness." (Art of War, Chapter 6).

* Wait, I should check if there are other specific quotes.
* *Buddhism*: "The soft tongue breaks the hard stone." (Proverb).
* *Laozi* is the best.
* I will provide the *Tao Te Ching* and *Sun Tzu*.

* Wait, one more check: The prompt asks for "Where else was a similar idea expressed?".
* Since the *Stalker* monologue is a direct adaptation of the *Tao Te Ching*, the *Tao Te Ching* is the correct answer.
* I will provide the *Tao Te Ching* quote.
* I will also provide *Sun Tzu* to add value.
* I need to make sure the Python dictionary syntax is correct.

* Wait, I need to be careful about the *Stalker* quote attribution. Some sources say Tarkovsky wrote it himself, inspired by the *Tao Te Ching*. Others say it's a direct translation. Regardless, the *idea* is from the *Tao Te Ching*.
* I will list *Tao Te Ching* by Laozi.
* I will list *The Art of War* by Sun Tzu.

* Wait, I need to check the exact wording for the *Tao Te Ching* to ensure it matches the "man born/dead, tree growing/dry" imagery.
* Chapter 76: "A man is born soft and supple; when he dies, he is stiff and hard. A tree is tender and pliant when it is alive, but dry and brittle when it is dead. Therefore, the stiff and hard are the companions of death. The soft and supple are the companions of life."
* This is the one.

* For *Sun Tzu*: "The supreme excellence is not to win a hundred victories in a hundred battles. The supreme excellence

[ ]: