```
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch
```

```
model_id = "key-life/codegen-alpaca-1b"
```

```
# Load model & tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_id)
model = AutoModelForCausalLM.from_pretrained(model_id, device_map="auto")
```

```
tokenizer_config.json:  4.14k/?  [00:00<00:00,  97.2kB/s]
vocab.json:  777k/?  [00:00<00:00,  3.13MB/s]
merges.txt:  442k/?  [00:00<00:00,  6.88MB/s]
tokenizer.json:  3.48M/?  [00:00<00:00,  13.2MB/s]
special_tokens_map.json:  100%  906/906  [00:00<00:00,  92.2kB/s]
adapter_config.json:  100%  706/706  [00:00<00:00,  68.6kB/s]
config.json:  100%  1.05k/1.05k  [00:00<00:00,  129kB/s]
model.safetensors:  100%  4.55G/4.55G  [00:28<00:00,  223MB/s]
generation_config.json:  100%  111/111  [00:00<00:00,  10.7kB/s]
adapter_model.safetensors:  100%  40.0/40.0  [00:01<00:00,  29.0B/s]
Loading adapter weights from key-life/codegen-alpaca-1b led to missing keys in the model: transformer.h.0.attn.c_attn.lora_A.de
```

```
# Example prompt
prompt = "### Instruction:\nWrite a C++ function to check if a number is even.\n\n### Response:\n"
inputs = tokenizer(prompt, return_tensors="pt").to(model.device)
```

```
# Generate code
outputs = model.generate(**inputs, max_new_tokens=128)
print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

```
Setting `pad_token_id` to `eos_token_id`:0 for open-end generation.
### Instruction:
Write a C++ function to check if a number is even.

### Response:
```
#include <iostream>
using namespace std;

bool isEven(int num)
{
    if(num % 2 == 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}

int main()
{
    int num;
    cout << "Enter a number: ";
    cin >> num;
    cout << "The number is " << isEven(num) << endl;
    return 0;
}
```
```

```
decoded = tokenizer.decode(outputs[0], skip_special_tokens=True)

# Try to extract code inside triple backticks
import re
code_block = re.findall(r"```(?:cpp|c\+\+|C\+\+)?(.*?)```", decoded, re.DOTALL)

if code_block:
    # Clean & strip extra spaces
    response = code_block[0].strip()
else:
```

```
    # Fallback: split at Response marker
    response = decoded.split("### Response:")[-1].strip()

print(response)
```

```
#include <iostream>
using namespace std;

bool isEven(int num)
{
    if(num % 2 == 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}

int main()
{
    int num;
    cout << "Enter a number: ";
    cin >> num;
    cout << "The number is " << isEven(num) << endl;
    return 0;
}
```