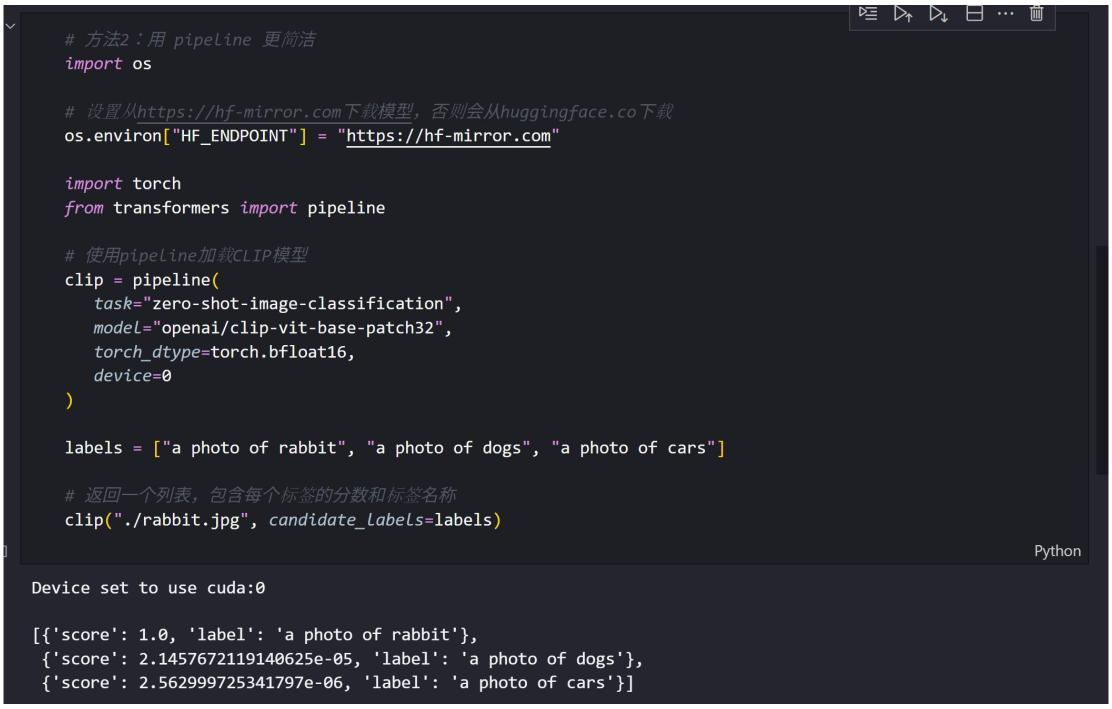
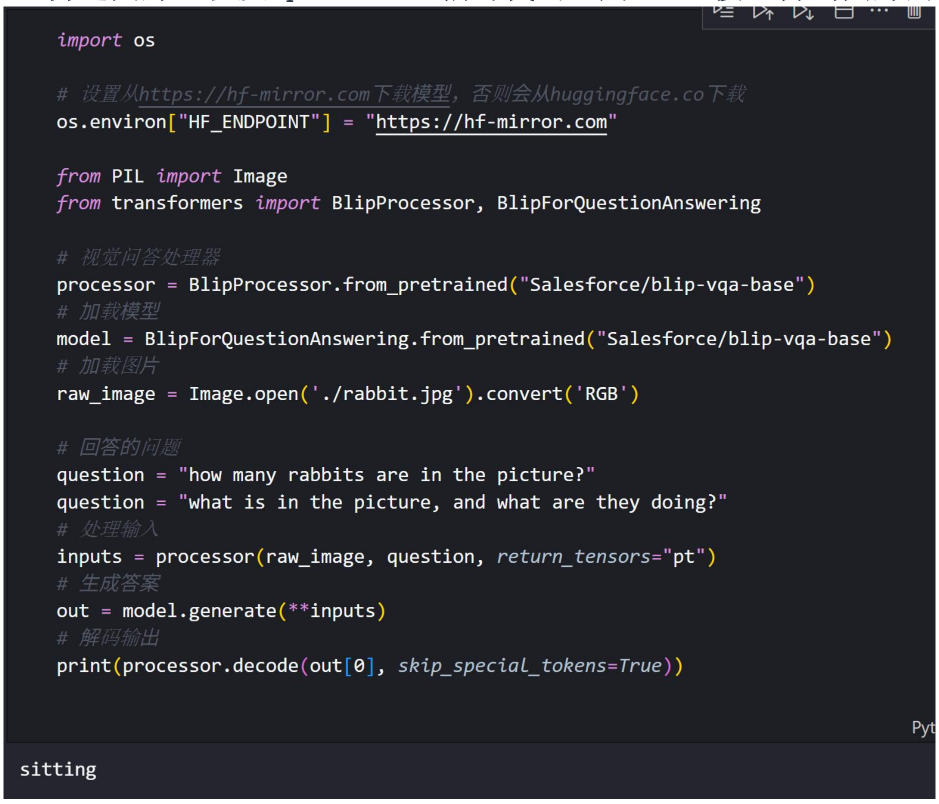
1. 自选图片，以及提示 labels，编写代码，用 CLIP 模型得到图片的分类



```python
# 方法2：用 pipeline 更简洁
import os

# 设置从https://hf-mirror.com下载模型，否则会从huggingface.co下载
os.environ["HF_ENDPOINT"] = "https://hf-mirror.com"

import torch
from transformers import pipeline

# 使用pipeline加载CLIP模型
clip = pipeline(
    task="zero-shot-image-classification",
    model="openai/clip-vit-base-patch32",
    torch_dtype=torch.bfloat16,
    device=0
)

labels = ["a photo of rabbit", "a photo of dogs", "a photo of cars"]

# 返回一个列表，包含每个标签的分数和标签名称
clip("./rabbit.jpg", candidate_labels=labels)
```

```
Device set to use cuda:0

[{'score': 1.0, 'label': 'a photo of rabbit'},
 {'score': 2.1457672119140625e-05, 'label': 'a photo of dogs'},
 {'score': 2.562999725341797e-06, 'label': 'a photo of cars'}]
```

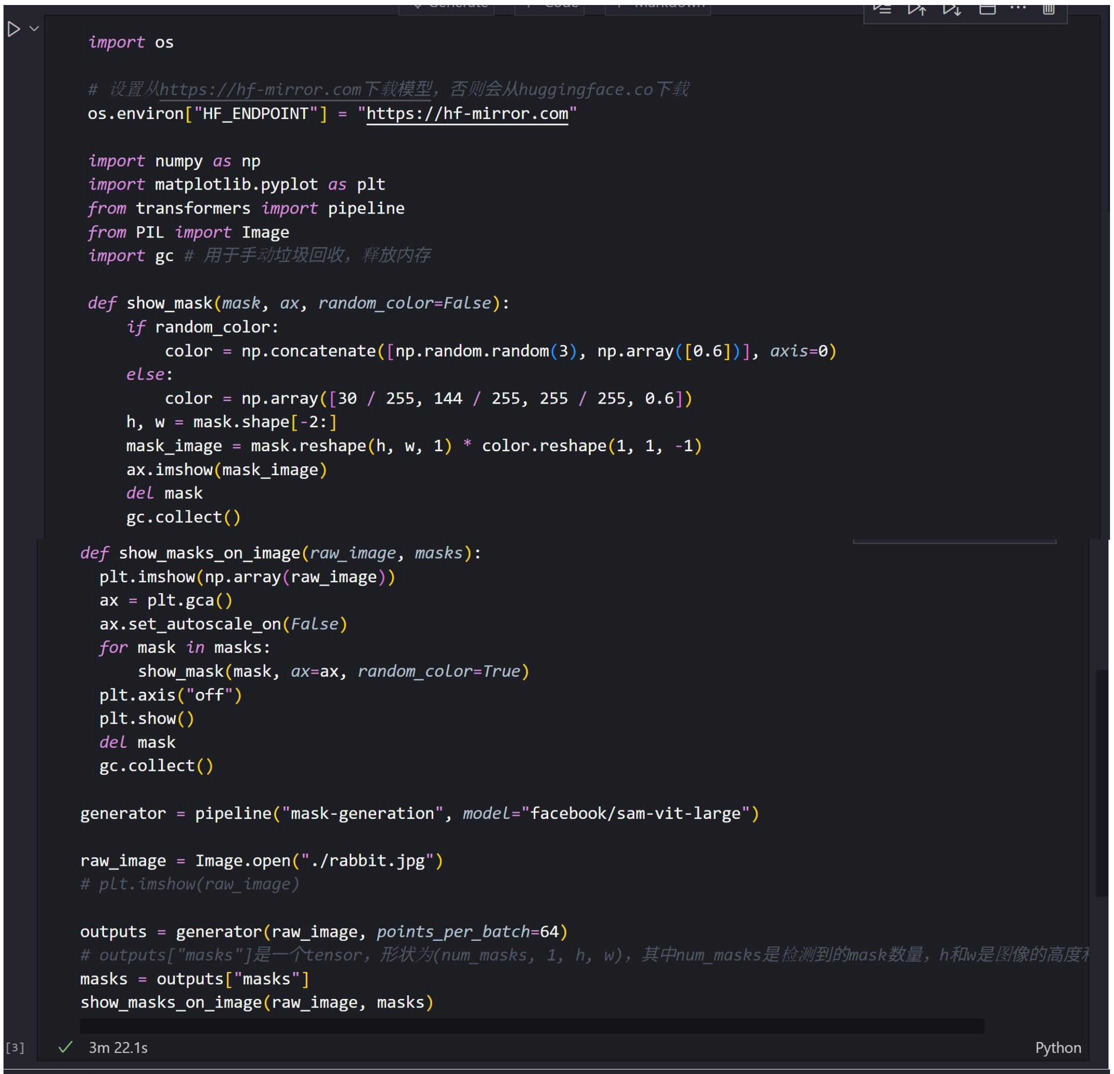
2. 自选图片，以及 question，编写代码，用 BLIP 模型得到图片的描述。



```python
import os

# 设置从https://hf-mirror.com下载模型，否则会从huggingface.co下载
os.environ["HF_ENDPOINT"] = "https://hf-mirror.com"

from PIL import Image
from transformers import BlipProcessor, BlipForQuestionAnswering

# 视觉问答处理器
processor = BlipProcessor.from_pretrained("Salesforce/blip-vqa-base")
# 加载模型
model = BlipForQuestionAnswering.from_pretrained("Salesforce/blip-vqa-base")
# 加载图片
raw_image = Image.open('./rabbit.jpg').convert('RGB')

# 回答的问题
question = "how many rabbits are in the picture?"
question = "what is in the picture, and what are they doing?"
# 处理输入
inputs = processor(raw_image, question, return_tensors="pt")
# 生成答案
out = model.generate(**inputs)
# 解码输出
print(processor.decode(out[0], skip_special_tokens=True))
```

```
sitting
```

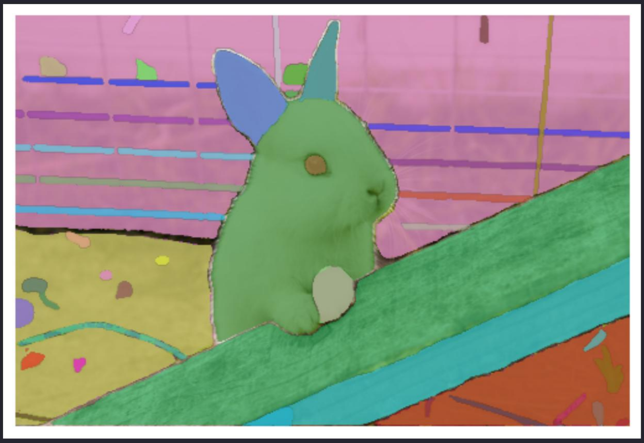
3. 自选图片，编写代码，用 SAM 模型对图片进行分割。



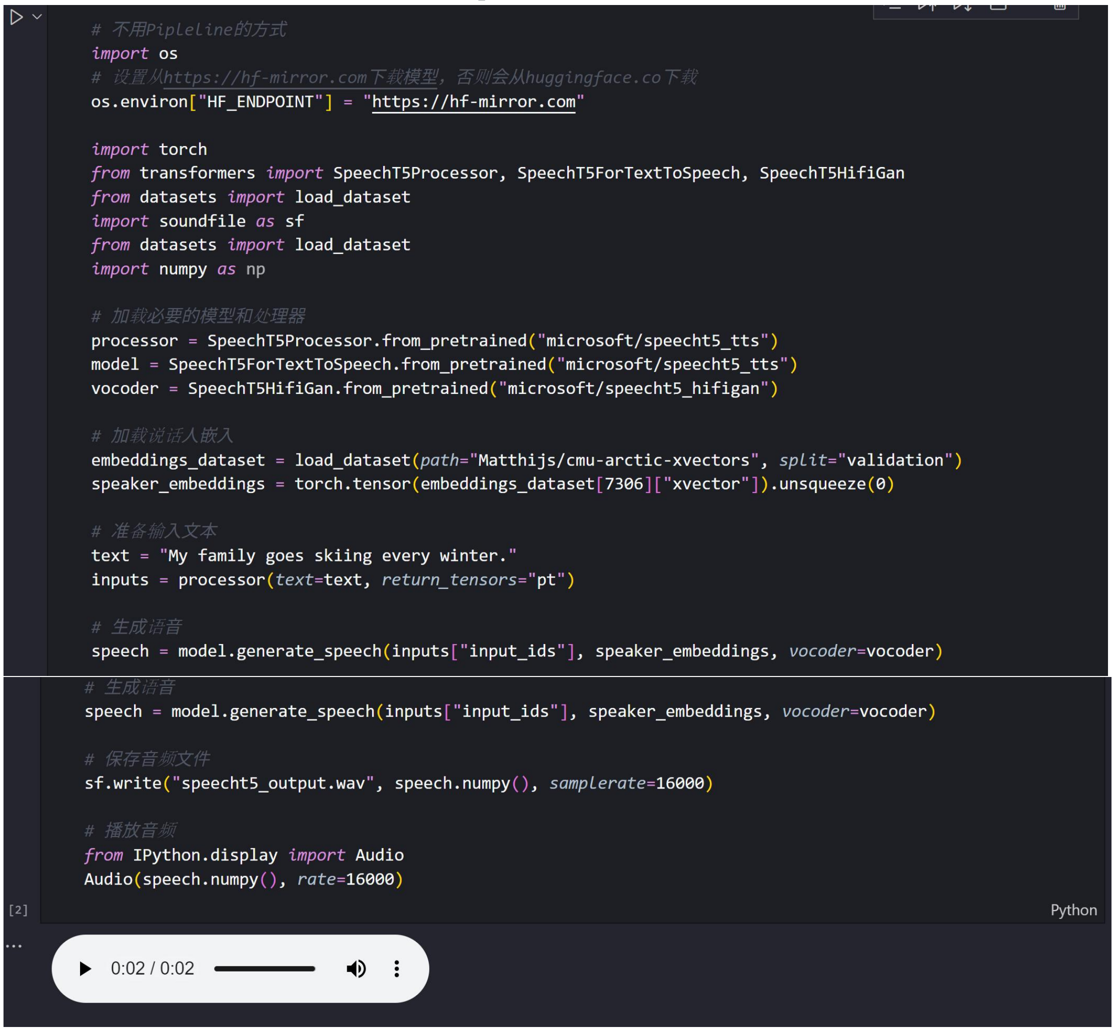
```python
import os

# 设置从https://hf-mirror.com下载模型，否则会从huggingface.co下载
os.environ["HF_ENDPOINT"] = "https://hf-mirror.com"

import numpy as np
import matplotlib.pyplot as plt
from transformers import pipeline
from PIL import Image
import gc # 用于手动垃圾回收，释放内存

def show_mask(mask, ax, random_color=False):
    if random_color:
        color = np.concatenate([np.random.random(3), np.array([0.6])], axis=0)
    else:
        color = np.array([30 / 255, 144 / 255, 255 / 255, 0.6])
    h, w = mask.shape[-2:]
    mask_image = mask.reshape(h, w, 1) * color.reshape(1, 1, -1)
    ax.imshow(mask_image)
    del mask
    gc.collect()
def show_masks_on_image(raw_image, masks):
  plt.imshow(np.array(raw_image))
  ax = plt.gca()
  ax.set_autoscale_on(False)
  for mask in masks:
      show_mask(mask, ax=ax, random_color=True)
  plt.axis("off")
  plt.show()
  del mask
  gc.collect()

generator = pipeline("mask-generation", model="facebook/sam-vit-large")

raw_image = Image.open("./rabbit.jpg")
# plt.imshow(raw_image)

outputs = generator(raw_image, points_per_batch=64)
# outputs["masks"]是一个tensor，形状为(num_masks, 1, h, w)，其中num_masks是检测到的mask数量，h和w是图像的高度
masks = outputs["masks"]
show_masks_on_image(raw_image, masks)
```

[3] ✓ 3m 22.1s Python

```
Device set to use cpu
```



4. 自选一段文字，编写代码，用 SpeechT5 模型将文字转成声音。

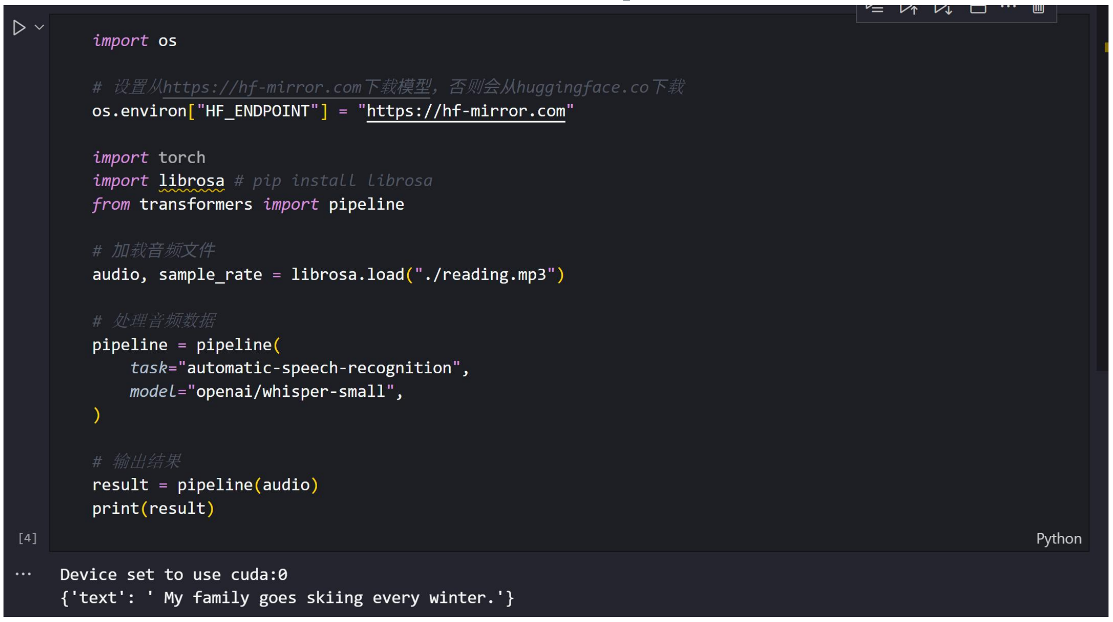


```python
# 不用Pipleline的方式
import os
# 设置从https://hf-mirror.com下载模型，否则会从huggingface.co下载
os.environ["HF_ENDPOINT"] = "https://hf-mirror.com"

import torch
from transformers import SpeechT5Processor, SpeechT5ForTextToSpeech, SpeechT5HifiGan
from datasets import load_dataset
import soundfile as sf
from datasets import load_dataset
import numpy as np

# 加载必要的模型和处理器
processor = SpeechT5Processor.from_pretrained("microsoft/speecht5_tts")
model = SpeechT5ForTextToSpeech.from_pretrained("microsoft/speecht5_tts")
vocoder = SpeechT5HifiGan.from_pretrained("microsoft/speecht5_hifigan")

# 加载说话人嵌入
embeddings_dataset = load_dataset(path="Matthijs/cmu-arctic-xvectors", split="validation")
speaker_embeddings = torch.tensor(embeddings_dataset[7306]["xvector"]).unsqueeze(0)

# 准备输入文本
text = "My family goes skiing every winter."
inputs = processor(text=text, return_tensors="pt")

# 生成语音
speech = model.generate_speech(inputs["input_ids"], speaker_embeddings, vocoder=vocoder)
```

```python
# 生成语音
speech = model.generate_speech(inputs["input_ids"], speaker_embeddings, vocoder=vocoder)
```

```python
# 保存音频文件
sf.write("speecht5_output.wav", speech.numpy(), samplerate=16000)

# 播放音频
from IPython.display import Audio
Audio(speech.numpy(), rate=16000)
```

5. 将上面题目得到声音，编写代码，用 Whisper 模型将声音再转成文字。



```python
import os

# 设置从https://hf-mirror.com下载模型，否则会从huggingface.co下载
os.environ["HF_ENDPOINT"] = "https://hf-mirror.com"

import torch
import librosa # pip install librosa
from transformers import pipeline

# 加载音频文件
audio, sample_rate = librosa.load("./reading.mp3")

# 处理音频数据
pipeline = pipeline(
    task="automatic-speech-recognition",
    model="openai/whisper-small",
)

# 输出结果
result = pipeline(audio)
print(result)
```

```
Device set to use cuda:0
{'text': ' My family goes skiing every winter.'}
```