Install required libraries

In [1]:

```
!pip install -q -U accelerate bitsandbytes git+https://github.com/huggingface/transforme
rs.git
```

```
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 315.1/315.1 kB 10.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 137.5/137.5 MB 6.5 MB/s eta 0:00:00
  Building wheel for transformers (pyproject.toml) ... done
```

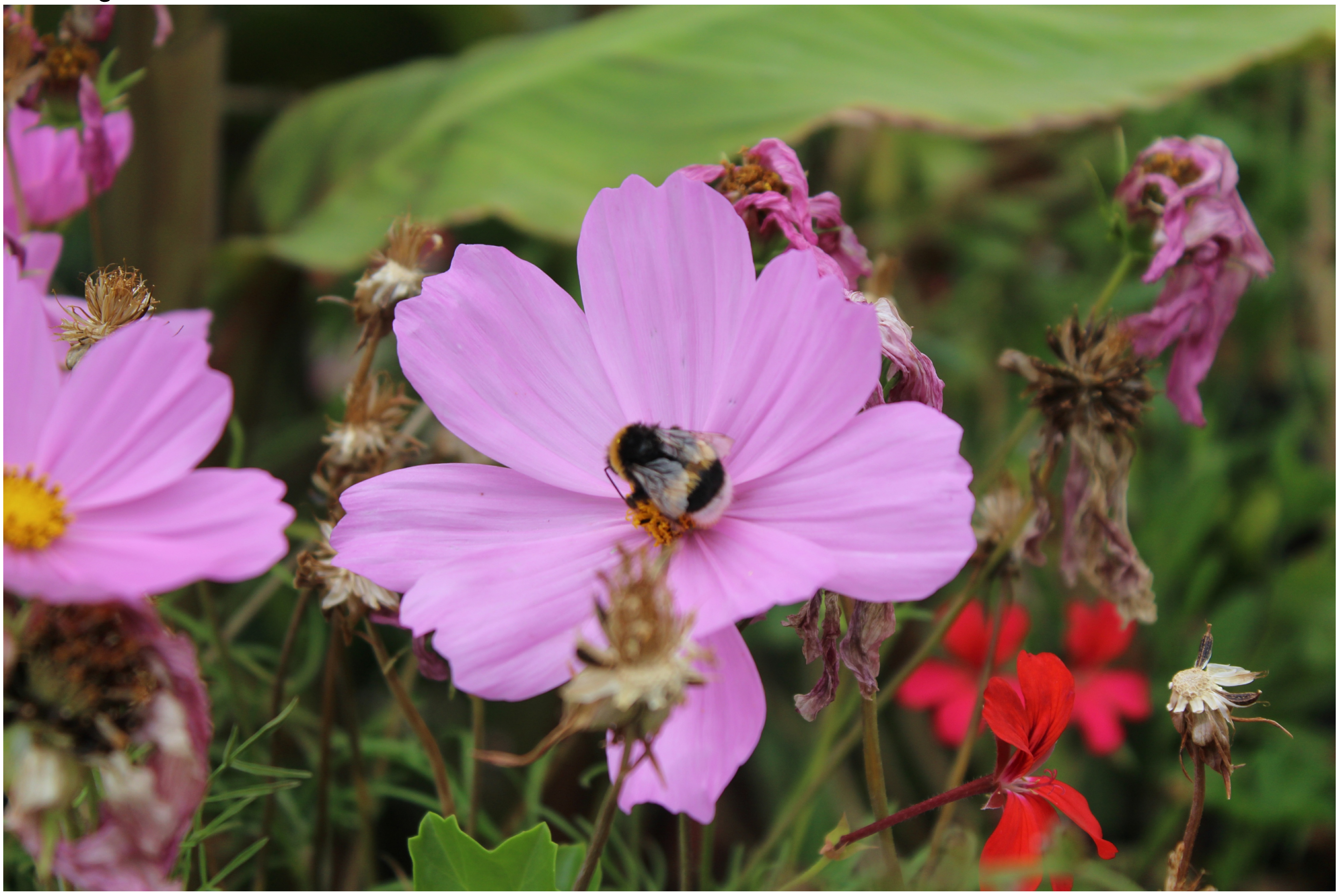
In [2]:

```
from huggingface_hub import notebook_login

notebook_login()
```

In [3]:

```
import torch
import numpy as np
from PIL import Image
import requests

input_text = "What color is the flower that bee is standing on?"
img_url = "https://huggingface.co/datasets/huggingface/documentation-images/resolve/main/
transformers/tasks/bee.JPG?download=true"
input_image = Image.open(requests.get(img_url, stream=True).raw)
```

The image looks like below.



Load PaliGemma Model

In [4]:

```
from transformers import AutoTokenizer, PaliGemmaForConditionalGeneration, PaliGemmaProce
ssor
import torch

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model_id = "google/paligemma-3b-mix-224"
model = PaliGemmaForConditionalGeneration.from_pretrained(model_id, torch_dtype=torch.bfl
oat16)
processor = PaliGemmaProcessor.from_pretrained(model_id)
```

```
`config.hidden_act` is ignored, you should use `config.hidden_activation` instead.
Gemma's activation function will be set to `gelu_pytorch_tanh`. Please, use
`config.hidden_activation` if you want to override this behaviour.
See https://github.com/huggingface/transformers/pull/29402 for more details.
```

In [5]:

```
inputs = processor(text=input_text, images=input_image,
                  padding="longest", do_convert_rgb=True, return_tensors="pt").to("cuda"
)
model.to(device)
inputs = inputs.to(dtype=model.dtype)
```

We can pass in our preprocessed inputs.

In [6]:

```
with torch.no_grad():
  output = model.generate(**inputs, max_length=496)

print(processor.decode(output[0], skip_special_tokens=True))
```

```
What color is the flower that bee is standing on?
pink
```

Load model in 4-bit

You can also load model in 4-bit and 8-bit, which offers memory gains during inference. First, initialize the `BitsAndBytesConfig`**.**

In [7]:

```
from transformers import BitsAndBytesConfig
import torch
nf4_config = BitsAndBytesConfig(
   load_in_4bit=True,
   bnb_4bit_quant_type="nf4",
   bnb_4bit_use_double_quant=True,
   bnb_4bit_compute_dtype=torch.bfloat16
)
```

We will now reload the model but pass in above object as `quantization_config`**.**

In [9]:

```
from transformers import AutoTokenizer, PaliGemmaForConditionalGeneration, PaliGemmaProce
ssor
import torch

device="cuda"
model_id = "google/paligemma-3b-mix-224"
model = PaliGemmaForConditionalGeneration.from_pretrained(model_id, torch_dtype=torch.bfl
oat16,
                                                          quantization_config=nf4_confi
g, device_map={"":0})
processor = PaliGemmaProcessor.from_pretrained(model_id)
```

In [10]:

```
with torch.no_grad():
  output = model.generate(**inputs, max_length=496)

print(processor.decode(output[0], skip_special_tokens=True))
```

```
What color is the flower that bee is standing on?
pink
```

In [ ]:

```

```