```
from transformers import pipeline
sentiment_analyzer = pipeline("sentiment-analysis")
text = "I am feeling happy"
sentiment_result = sentiment_analyzer(text)
print("Sentiment analysis:")
print(f"Sentiment: {sentiment_result[0]['label']} (score: {sentiment_result[0]['score']:.2f})\n")
```

```
No model was supplied, defaulted to distilbert/distilbert-base-uncased-finetuned-sst-2-english and revision 714eb0f (https://hug
Using a pipeline without specifying a model name and revision in production is not recommended.
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json: 100%  629/629 [00:00<00:00, 14.4kB/s]
model.safetensors: 100%  268M/268M [00:02<00:00, 151MB/s]
tokenizer_config.json: 100%  48.0/48.0 [00:00<00:00, 4.70kB/s]
vocab.txt:  232k/? [00:00<00:00, 1.57MB/s]
Device set to use cpu
Sentiment analysis:
Sentiment: POSITIVE (score: 1.00)
```

```
from transformers import pipeline
translator_en_to_fr = pipeline("translation_en_to_fr", model="Helsinki-NLP/opus-mt-en-fr")
text = "I am feeling happy"
translation_result = translator_en_to_fr(text)
print("Translation:")
print(f"Translated text: {translation_result[0]['translation_text']}\n")
```

```
config.json:  1.42k/? [00:00<00:00, 32.3kB/s]
pytorch_model.bin: 100%  301M/301M [00:02<00:00, 98.3MB/s]
model.safetensors: 100%  301M/301M [00:02<00:00, 180MB/s]
generation_config.json: 100%  293/293 [00:00<00:00, 9.66kB/s]
tokenizer_config.json: 100%  42.0/42.0 [00:00<00:00, 1.55kB/s]
source.spm: 100%  778k/778k [00:00<00:00, 9.20MB/s]
target.spm: 100%  802k/802k [00:00<00:00, 16.4MB/s]
vocab.json:  1.34M/? [00:00<00:00, 25.6MB/s]
/usr/local/lib/python3.11/dist-packages/transformers/models/marian/tokenization_marian.py:175: UserWarning: Recommended: pip ins
  warnings.warn("Recommended: pip install sacremoses.")
Device set to use cpu
Translation:
Translated text: Je me sens heureux.
```

```
!pip  install pytesseract
```

```
Collecting pytesseract
  Downloading pytesseract-0.3.13-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-packages (from pytesseract) (25.0)
Requirement already satisfied: Pillow>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from pytesseract) (11.3.0)
Downloading pytesseract-0.3.13-py3-none-any.whl (14 kB)
Installing collected packages: pytesseract
Successfully installed pytesseract-0.3.13
```

```
import pytesseract
from PIL import Image
from transformers import pipeline
```

```
import re
input_image = Image.open('/content/5.1.png')
text_output = pytesseract.image_to_string(input_image, lang='eng').strip()
clean_text = re.sub(r"\s+,", ",", text_output)
print(clean_text)
```

```
{import random

def (x)

return x02 4 40%

def hill_climbing(f, start_point, step_siz


1, max_dterations=100


current_point = start_poind]


current_value


#(current_point)

for _ in range(max_iterations):

neighbors


‘current_point - step_size, current_point + step_size]

neightor_values = [F(neighbor) for neighbor in neighbors]

best_value = max(neighbor_values)

bbest_neighbor = neighbors{neightor_values. index(best_value)]

Af best_value > current_value:

‘current_point = best_neighbor

current_value = best_value

else:

break

return current_point, current_value

start_point = candon.unsform(2,5)

optinal_x, optinal_velue = hill_clinbing(f, etart_point)

print(#"optinal x: (optinal_x)")

print(#"Maximum value of f(x): (optimal_value)")
```

```
image = Image.open('/content/5.1.png')
extracted_text = pytesseract.image_to_string(image).strip()
if not extracted_text:
  print("No text found in the image.")
else:
  print(f"Extracted text: {extracted_text}")
  sentiment_analyzer=pipeline("sentiment-analysis")
  sentiment_result=sentiment_analyzer(extracted_text)[0]
  #print("Sentiment analysis:")
  #print(f"Text: {extracted_text}")
  print(f"Sentiment: {sentiment_result['label']} (score: {sentiment_result['score']:.2f})")
```

```
No model was supplied, defaulted to distilbert/distilbert-base-uncased-finetuned-sst-2-english and revision 714eb0f (https://h
Using a pipeline without specifying a model name and revision in production is not recommended.
Extracted text: {import random

def (x)

return x02 4 40%

def hill_climbing(f, start_point, step_siz
```

```
1, max_dterations=100


current_point = start_poind]


current_value


#(current_point)
for _ in range(max_iterations):
neighbors


‘current_point - step_size, current_point + step_size]
neightor_values = [F(neighbor) for neighbor in neighbors]
best_value = max(neighbor_values)
bbest_neighbor = neighbors{neightor_values. index(best_value)]
Af best_value > current_value:
‘current_point = best_neighbor
current_value = best_value
else:
break
return current_point, current_value
start_point = candon.unsform(2,5)
optinal_x, optinal_velue = hill_clinbing(f, etart_point)
print(#"optinal x: (optinal_x)")
```

```
Start coding or generate with AI.
```