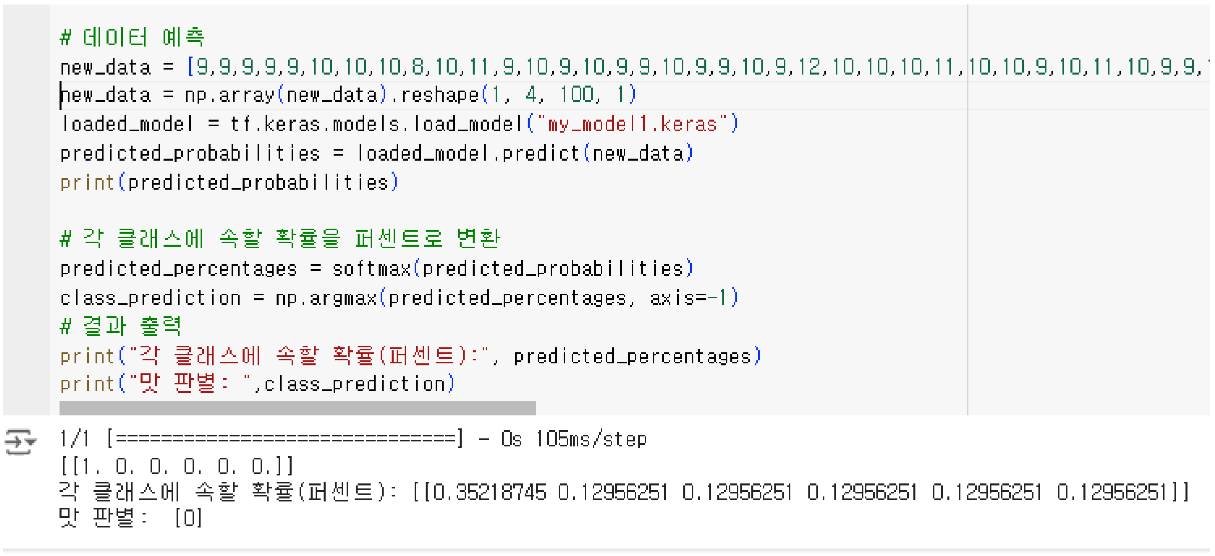
검증 결과

0: unknown (알수없음) 1: sweetness(단맛) 2: salty(짠맛) 3: sour(신맛) 4: bitter(쓴맛) 5: palatability(감칠맛)

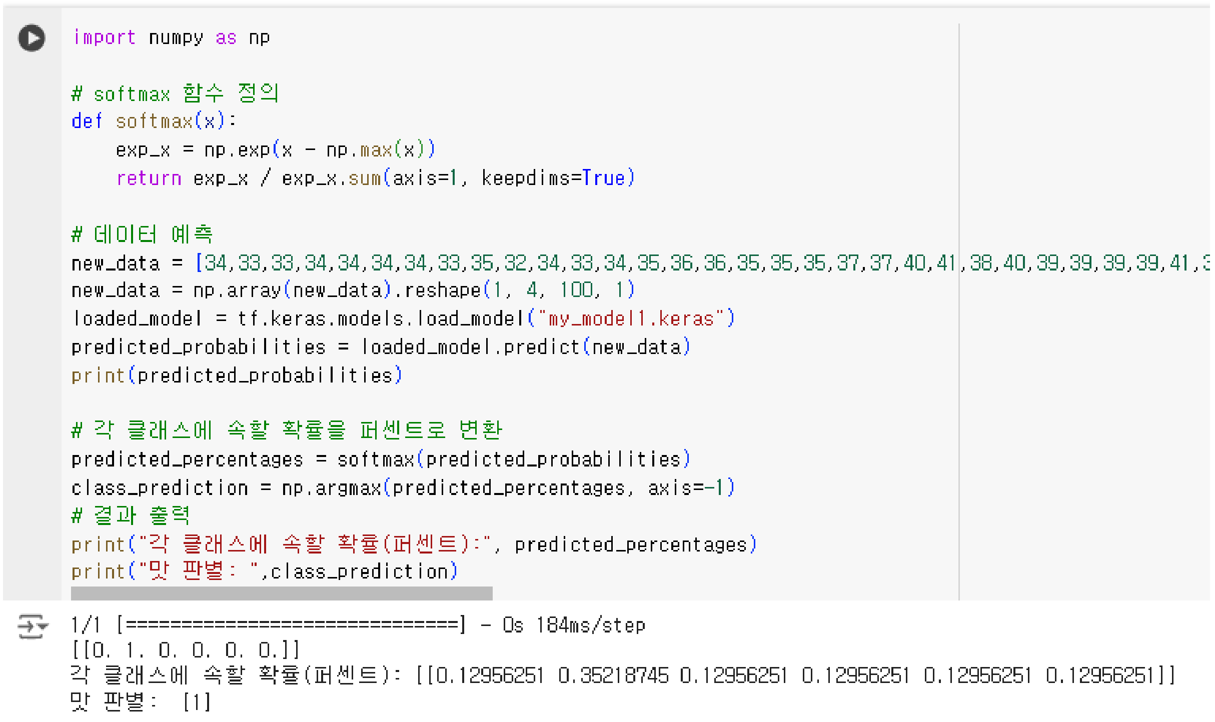
unknown: stainless (스테인리스 팬)



```
# 데이터 예측
new_data = [9,9,9,9,9,10,10,10,8,10,11,9,10,9,10,9,9,10,9,9,10,9,12,10,10,10,11,10,10,9,10,11,10,9,9,
new_data = np.array(new_data).reshape(1, 4, 100, 1)
loaded_model = tf.keras.models.load_model("my_model1.keras")
predicted_probabilities = loaded_model.predict(new_data)
print(predicted_probabilities)

# 각 클래스에 속할 확률을 퍼센트로 변환
predicted_percentages = softmax(predicted_probabilities)
class_prediction = np.argmax(predicted_percentages, axis=-1)
# 결과 출력
print("각 클래스에 속할 확률(퍼센트):", predicted_percentages)
print("맛 판별: ",class_prediction)
```

```
1/1 [==============================] - 0s 105ms/step
[[1. 0. 0. 0. 0. 0.]]
각 클래스에 속할 확률(퍼센트): [[0.35218745 0.12956251 0.12956251 0.12956251 0.12956251 0.12956251]]
맛 판별:  [0]
```

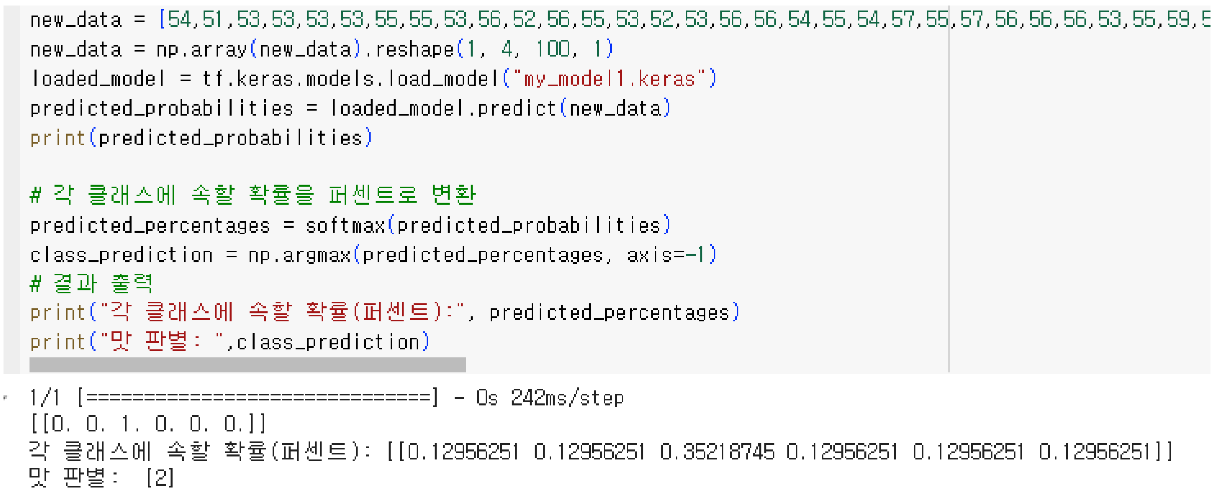
sweetness: chocolate(초콜릿)



```
import numpy as np

# softmax 함수 정의
def softmax(x):
    exp_x = np.exp(x - np.max(x))
    return exp_x / exp_x.sum(axis=1, keepdims=True)

# 데이터 예측
new_data = [34,33,33,34,34,34,34,33,35,32,34,33,34,35,36,36,35,35,35,37,37,40,41,38,40,39,39,39,39,41,3
new_data = np.array(new_data).reshape(1, 4, 100, 1)
loaded_model = tf.keras.models.load_model("my_model1.keras")
predicted_probabilities = loaded_model.predict(new_data)
print(predicted_probabilities)

# 각 클래스에 속할 확률을 퍼센트로 변환
predicted_percentages = softmax(predicted_probabilities)
class_prediction = np.argmax(predicted_percentages, axis=-1)
# 결과 출력
print("각 클래스에 속할 확률(퍼센트):", predicted_percentages)
print("맛 판별: ",class_prediction)
```

```
1/1 [==============================] - 0s 184ms/step
[[0. 1. 0. 0. 0. 0.]]
각 클래스에 속할 확률(퍼센트): [[0.12956251 0.35218745 0.12956251 0.12956251 0.12956251 0.12956251]]
맛 판별:  [1]
```

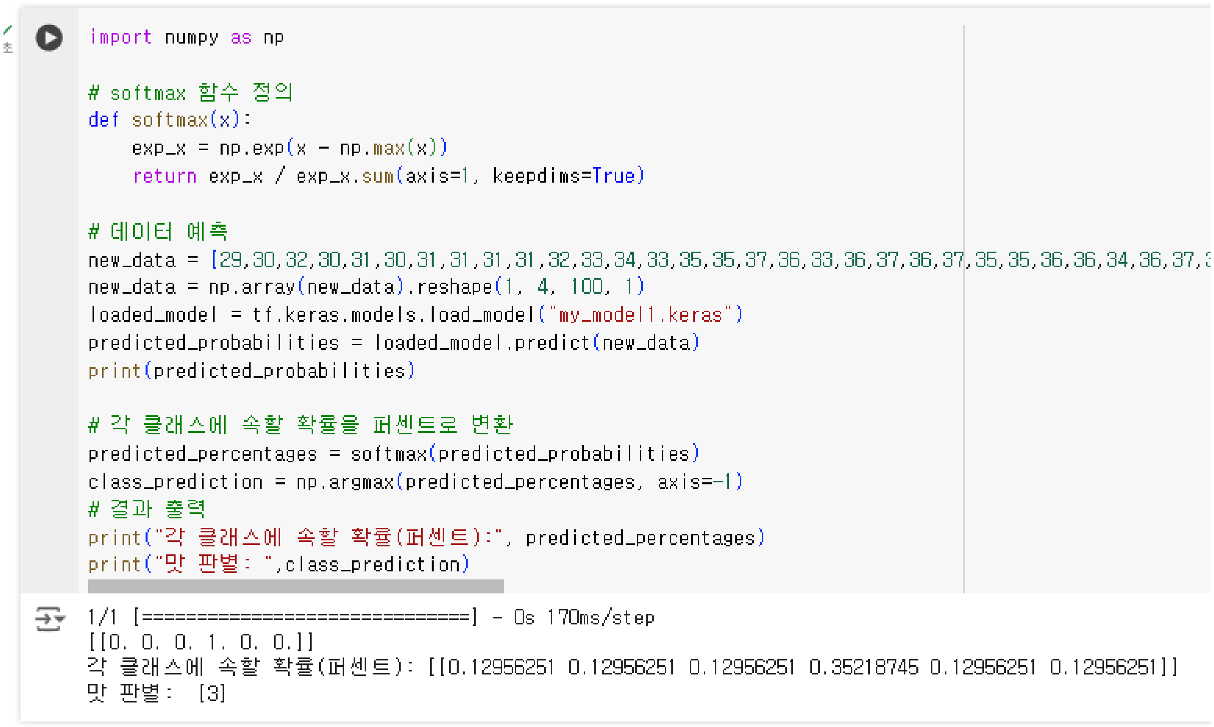
salty: soy sauce(간장)



```
new_data = [54,51,53,53,53,53,55,55,53,56,52,56,55,53,52,53,56,56,54,55,54,57,55,57,56,56,56,53,55,59,5
new_data = np.array(new_data).reshape(1, 4, 100, 1)
loaded_model = tf.keras.models.load_model("my_model1.keras")
predicted_probabilities = loaded_model.predict(new_data)
print(predicted_probabilities)

# 각 클래스에 속할 확률을 퍼센트로 변환
predicted_percentages = softmax(predicted_probabilities)
class_prediction = np.argmax(predicted_percentages, axis=-1)
# 결과 출력
print("각 클래스에 속할 확률(퍼센트):", predicted_percentages)
print("맛 판별: ",class_prediction)
```

```
1/1 [==============================] - 0s 242ms/step
[[0. 0. 1. 0. 0. 0.]]
각 클래스에 속할 확률(퍼센트): [[0.12956251 0.12956251 0.35218745 0.12956251 0.12956251 0.12956251]]
맛 판별:  [2]
```

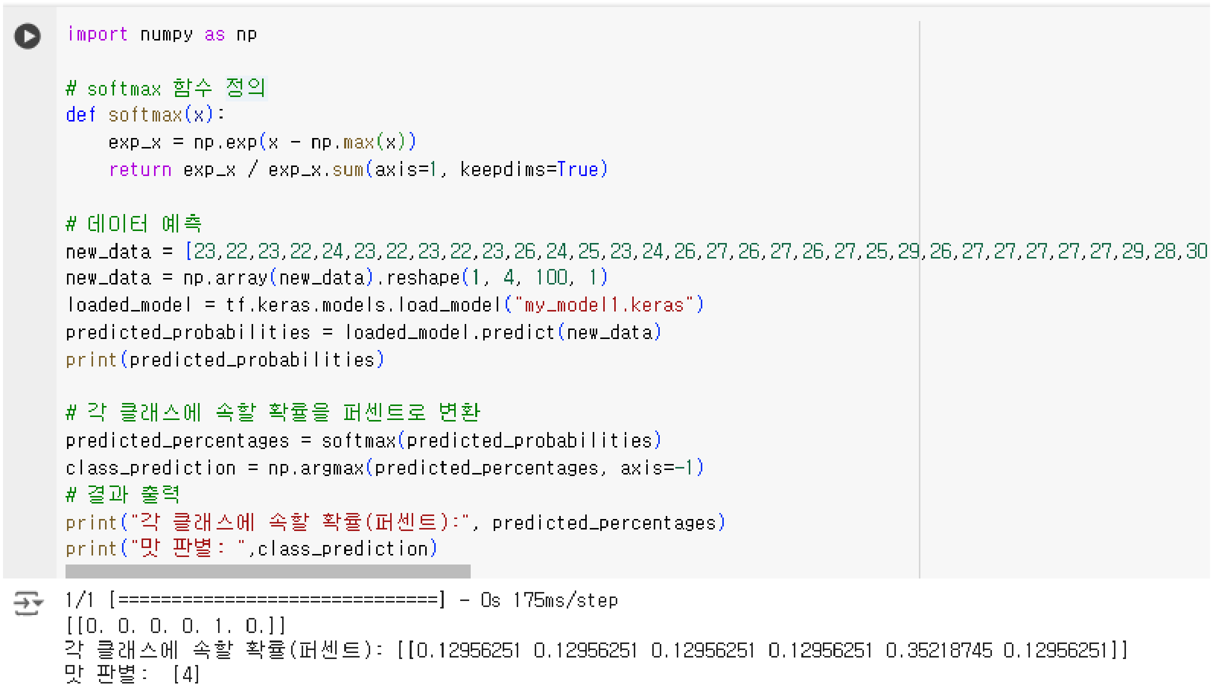
sour: sourmix(샤워믹스)



```
import numpy as np

# softmax 함수 정의
def softmax(x):
    exp_x = np.exp(x - np.max(x))
    return exp_x / exp_x.sum(axis=1, keepdims=True)

# 데이터 예측
new_data = [29,30,32,30,31,30,31,31,31,31,32,33,34,33,35,35,37,36,33,36,37,36,37,35,35,36,36,34,36,37,3
new_data = np.array(new_data).reshape(1, 4, 100, 1)
loaded_model = tf.keras.models.load_model("my_model1.keras")
predicted_probabilities = loaded_model.predict(new_data)
print(predicted_probabilities)

# 각 클래스에 속할 확률을 퍼센트로 변환
predicted_percentages = softmax(predicted_probabilities)
class_prediction = np.argmax(predicted_percentages, axis=-1)
# 결과 출력
print("각 클래스에 속할 확률(퍼센트):", predicted_percentages)
print("맛 판별: ",class_prediction)
```

```
1/1 [==============================] - 0s 170ms/step
[[0. 0. 0. 1. 0. 0.]]
각 클래스에 속할 확률(퍼센트): [[0.12956251 0.12956251 0.12956251 0.35218745 0.12956251 0.12956251]]
맛 판별:  [3]
```

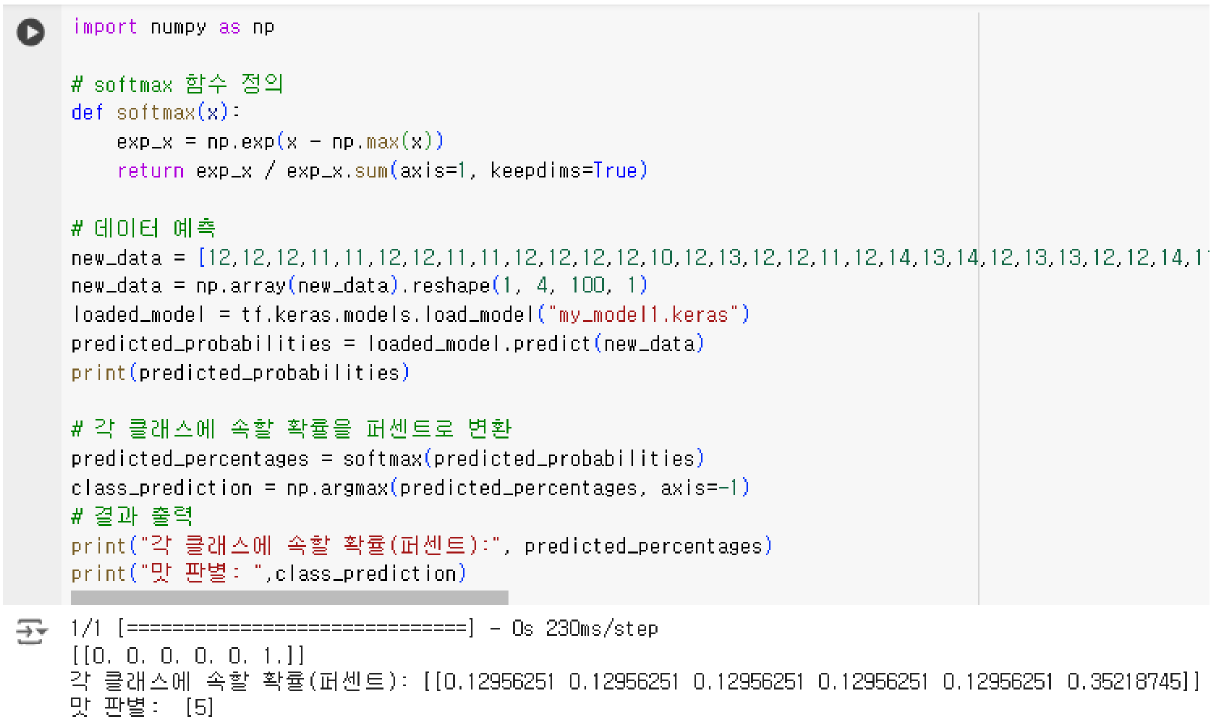
bitter: Dark chocolate(다크초콜릿)



```
import numpy as np

# softmax 함수 정의
def softmax(x):
    exp_x = np.exp(x - np.max(x))
    return exp_x / exp_x.sum(axis=1, keepdims=True)

# 데이터 예측
new_data = [23,22,23,22,24,23,22,23,22,23,26,24,25,23,24,26,27,26,27,26,27,25,29,26,27,27,27,27,27,29,28,30
new_data = np.array(new_data).reshape(1, 4, 100, 1)
loaded_model = tf.keras.models.load_model("my_model1.keras")
predicted_probabilities = loaded_model.predict(new_data)
print(predicted_probabilities)

# 각 클래스에 속할 확률을 퍼센트로 변환
predicted_percentages = softmax(predicted_probabilities)
class_prediction = np.argmax(predicted_percentages, axis=-1)
# 결과 출력
print("각 클래스에 속할 확률(퍼센트):", predicted_percentages)
print("맛 판별: ",class_prediction)
```

```
1/1 [==============================] - 0s 175ms/step
[[0. 0. 0. 0. 1. 0.]]
각 클래스에 속할 확률(퍼센트): [[0.12956251 0.12956251 0.12956251 0.12956251 0.35218745 0.12956251]]
맛 판별:  [4]
```

palatability: miwon(미원)



```
import numpy as np

# softmax 함수 정의
def softmax(x):
    exp_x = np.exp(x - np.max(x))
    return exp_x / exp_x.sum(axis=1, keepdims=True)

# 데이터 예측
new_data = [12,12,12,11,11,12,12,11,11,12,12,12,12,10,12,13,12,12,11,12,14,13,14,12,13,13,12,12,14,1
new_data = np.array(new_data).reshape(1, 4, 100, 1)
loaded_model = tf.keras.models.load_model("my_model1.keras")
predicted_probabilities = loaded_model.predict(new_data)
print(predicted_probabilities)

# 각 클래스에 속할 확률을 퍼센트로 변환
predicted_percentages = softmax(predicted_probabilities)
class_prediction = np.argmax(predicted_percentages, axis=-1)
# 결과 출력
print("각 클래스에 속할 확률(퍼센트):", predicted_percentages)
print("맛 판별: ",class_prediction)
```

```
1/1 [==============================] - 0s 230ms/step
[[0. 0. 0. 0. 0. 1.]]
각 클래스에 속할 확률(퍼센트): [[0.12956251 0.12956251 0.12956251 0.12956251 0.12956251 0.35218745]]
맛 판별:  [5]
```