1일차 실습

Python 기초

간단한 계산

덧셈

```
1 + 2
```

뺄셈

```
5 - 3
```

곱셈

```
7 * 3
```

나눗셈

```
8 / 2
```

우선 순위

곱셈과 나눗셈을 덧셈과 뺄셈보다 먼저한다.

```
1 + 2 * 3
```

괄호

괄호 안을 먼저 계산한다.

```
(1 + 2) * 3
```

괄호는 둥근 괄호만 사용한다. 괄호를 여러 개 겹쳐 쓸 경우 안쪽 괄호부터 계산을 한다.

```
(1 + (3 - 2)) * 4
```

주석

`#` 뒤에 쓴 내용은 무시된다. `#`를 이용해서 코드에 주석을 달 수 있다.

```
1 + 1 # 일 더하기 일
```

변수

프로그래밍에서는 계산이나 처리의 대상을 값이라 한다. 변수는 이 값을 담는 그릇에 해당한다. 파이썬에서는 =을 이용해 변수를 할당한다. =은 수학에서와 달리 "같다"를 나타내지 않는다. 아래 식은 변수 a에 값 1을 할당한다.

```
a = 1
```

이제 a는 1을 가리킨다.

a + 3을 하면 1 + 3을 하는 것과 같다.

```
a + 3
```

재할당

한 번 값을 할당한 변수에 다른 값을 할당할 수도 있다.

```
a = 5
```

재할당

`=`은 항상 오른쪽부터 계산한다. 다음과 같이 `a = a + 1`이라고 하면 `a + 1`을 먼저 계산한다. 현재 `a`는 `5`를 가리키므로, `a + 1`은 `6`이 된다. 다음으로 이 결과를 `a`에 할당한다.

```
a = a + 1
```

따라서 `a`의 값은 `6`이 된다.

변수 이름

변수 이름에는 알파벳만이 아니라 한글 등 다른 문자와 숫자, 밑줄 `_`을 쓸 수 있다.

단, 변수 이름 첫 자리에는 숫자를 쓸 수 없다(예: `1변수`).

문자열

문자 데이터. 작은 따옴표(`'`) 또는 큰 따옴표(`"`)로 감싸서 표현

```
'hello'
```

리스트

여러 개의 값을 순서대로 포함하는 자료 구조

```
x = [1, 2, 3]
```

인덱싱을 통해 특정한 값만 가리킬 수 있음

```
x[0]
```

Python에서는 인덱스가 0번 부터 시작

함수

함수는 일종의 명령으로서 일정한 인자(argument)를 넘겨받아 실행하고, 그 결과를 반환한다.

함수의 인자는 `f(x)`와 같이 괄호 안에 쓴다.

인자가 여러 개일 때는 `f(x, y)`와 같이 쉼표로 구분한다.

아래 `min` 함수는 두 개의 값을 받아 그 중에 더 작은 값을 반환한다.

```
min(1, 2)
```

모듈 불러오기

`import` 문을 통해 모듈을 불러올 수 있다

`random.randint`는 `random` 모듈을 `randint` 함수를 가리킨다.

```
import random
random.randint(1, 6)
```

`randint` 함수는 최솟값과 최댓값을 받아 그 사이의 정수 하나를 무작위로 반환한다.

모듈에서 특정 함수만 불러오기

```
from random import randint
randint(1, 6)
```

모듈을 별칭으로 불러오기

```
import random as rd
rd.randint(1, 6)
```

텐서플로 기초

딥러닝 프레임워크

텐서플로 TensorFlow: 구글이 지원, 다양한 기능 제공, 기업에서 널리 사용 중

파이토치 PyTorch: 페이스북이 지원, 빠른 구현이 쉬움, 학계에서 인기가 커지는 중

텐서: 행렬의 일반화. 다차원 배열

임포트

```
import tensorflow as tf
```

로지스틱 회귀분석 모형

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Dense(1, activation='sigmoid')
])
```

파라미터

```
model.weights
```

텐서 만들기

$$x = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}$$

```
x = tf.convert_to_tensor([
    [1, 0],
    [0, 1],
    [1, 1],
], dtype=tf.float32)
```

예측

모형에 데이터 입력

```
prediction = model(x)
```

예측 결과 확인

```
prediction.numpy()
```

MNIST 손글씨 데이터

```
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

전처리

```
x_train, x_test = x_train / 255.0, x_test / 255.0
```

데이터 보기

이미지

```
import matplotlib.pyplot as plt
plt.imshow(x_train[0])
```

답 보기

```
y_train[0]
```

모형 정의

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(10)
])
```

예측

```
predictions = model(x_train[:1]).numpy()
```

로짓을 확률로 변환

```
tf.nn.softmax(predictions).numpy()
```

손실

```
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
```

```
loss_fn(y_train[:1], predictions).numpy()
```

모형 학습 방법 정의

```
model.compile(optimizer='adam',
              loss=loss_fn,
              metrics=['accuracy'])
```

학습

```
model.fit(x_train, y_train, epochs=5)
```

평가

```
model.evaluate(x_test,  y_test, verbose=2)
```

확률을 출력하는 모형으로 수정

모형 정의

```
probability_model = tf.keras.Sequential([
  model,
  tf.keras.layers.Softmax()
])
```

확률 출력

```
probability_model(x_test[:5])
```