BAGIAN 1: PRIMITIVE DATA TYPES (Soal 1-5)

Soal 1: Dalam machine learning, learning_rate = 0.01 (float). Tentukan tipe data dari 0.01 dan hitung learning_rate * 100

Output:

```
<class 'float'>
1.0
```

In [1]:
```
# Answer 1
learning_rate = 0.01
print(type(learning_rate))
print(learning_rate * 100)
```

```
<class 'float'>
1.0
```

Soal 2: Model deep learning memiliki epoch = 50 (int) dan batch_size = 32 (int). Hitung total_samples = epoch * batch_size

Output:

```
Total samples: 1600
```

In [3]:
```
# Answer 2
epoch = 50
batch_size = 32
# Hint: Kalikan epoch dan batch_size
total_samples = epoch * batch_size
print(f"Total samples: {total_samples}")
```

```
Total samples: 1600
```

Soal 3: Akurasi model = 0.9567. Konversi ke persentase dengan format persentase.

Output:

```
Akurasi: 95.67%
```

In [ ]:
```
# Answer 3
accuracy = 0.9567 * 100

print(f"Akurasi: {accuracy}%")
```

```
Akurasi: 95.67%
```

Soal 4: Training loss = 0.234 dan validation_loss = 0.312. Hitung selisih dan tentukan apakah model overfitting (diff > 0.05)

Output:

```
Overfitting: True
```

In [8]:
```
# Answer 4
training_loss = 0.234
validation_loss = 0.312
# Hint: Hitung selisih dari validation_loss dan training_loss
diff = validation_loss - training_loss
# Hint: Bandingkan diff dengan 0.05
is_overfitting = diff > 0.05
print(f"Overfitting: {is_overfitting}")
```

```
Overfitting: True
```

Soal 5: Konversi string model_name = "ResNet50" menjadi panjangnya dan tampilkan hasilnya.

Output:

```
ResNet50 memiliki 8 karakter
```

In [32]:
```
# Answer 5
model_name = "ResNet50"

print(f"{model_name} memiliki {len(model_name)} karakter")
```

```
ResNet50 memiliki 8 karakter
```

BAGIAN 2: STRING OPERATIONS (Soal 6-9)

Soal 6: Model AI memiliki nama framework = "TensorFlow". Ubah menjadi huruf kecil dan ambil 4 karakter pertama

Output:

```
tens
```

In [31]:
```
# Answer 6
framework = "TensorFlow"
lowercase_framework = framework.lower()  # Ubah ke huruf kecil
# Hint: Ambil 4 karakter pertama menggunakan indexing
print(lowercase_framework[:4])
```

```
tens
```

Soal 7: Gabungkan layer_1 = "Convolutional" dan layer_2 = "MaxPooling" dengan separator " -> "

Output:

```
Convolutional -> MaxPooling
```

In [30]:
```
# Answer 7

layer_1 = "Convolutional"
layer_2 = "MaxPooling"


print(f"{layer_1} -> {layer_2}")
```

```
Convolutional -> MaxPooling
```

Soal 8: Split string layers = "CNN,RNN,Transformer" menjadi list dan tampilkan elemen pertama

Output:

```
CNN
```

In [33]:
```
# Answer 8
layers = "CNN,RNN,Transformer"
# Hint: Gunakan method split() dengan separator yang tepat
layer_list = layers.split(",")
print(layer_list[0])  # Akses elemen pertama
```

```
CNN
```

Soal 9: Dari list algorithms = ["Gradient Descent", "Adam", "SGD"], gabungkan menjadi string dengan separator ", "

Output:

```
Gradient Descent, Adam, SGD
```

In [34]:
```
# Answer 9

list_algorithms = ["Gradien Desent", "Adam", "SGD"]

print(", ".join(list_algorithms))
print()
```

```
Gradien Desent, Adam, SGD
```

BAGIAN 3: COMPOSITE DATA TYPES - LIST & DICT (Soal 10-14)

Soal 10: Buat list neurons = [64, 128, 256, 512] dan tambahkan 1024 di akhir, lalu tampilkan panjangnya

Output:

```
Total layers: 5
```

In [27]:
```python
# Answer 10
neurons = [64, 128, 256, 512]
# Hint: Gunakan method untuk menambah elemen di akhir
neurons.append(1024)
print(f"Total layers: {len(neurons)}")
```

```
Total layers: 5
```

Soal 11: Dari list predictions = [0.9, 0.2, 0.8, 0.1], akses elemen pertama dan terakhir

Output:

```
First: 0.9, Last: 0.1
```

In [ ]:
```python
# Answer 11
list_prediction = [0.9, 0.2, 0.8, 0.1]
print(f"First: {list_prediction[0]}, Last: {list_prediction[-1]}")

print()
```

```
First: 0.9, Last: 0.1
```

Soal 12: Buat dictionary model_config = {'name': 'VGG16', 'layers': 16, 'pretrained': True} dan akses nilai 'layers'

Output:

```
Number of layers: 16
```

In [38]:
```python
# Answer 12
# Deklarasikan dictionary model_config
# Hint: Akses nilai dengan menggunakan key dalam square bracket
# Output yang diinginkan: "Number of layers: 16"
model_config = {
    'name' : 'VGG16',
    'layers' : 16,
    'pretrained' : True
}

print("Number of layers:", model_config['layers'])
```

```
Number of layers: 16
```

Soal 13: Update dictionary hyperparameters = {'lr': 0.001, 'momentum': 0.9} dengan menambah 'weight_decay': 0.0001

Output:

```
{'lr': 0.001, 'momentum': 0.9, 'weight_decay': 0.0001}
```

In [40]:
```
# Answer 13
hyperparameters = {'lr': 0.001, 'momentum': 0.9}
# Hint: Tambah key dan value baru ke dictionary
hyperparameters['weight_decay'] = 0.0001
print(hyperparameters)
```

```
{'lr': 0.001, 'momentum': 0.9, 'weight_decay': 0.0001}
```

Soal 14: Dari dictionary results = {'train_acc': 0.95, 'val_acc': 0.92, 'test_acc': 0.91}, tampilkan semua keys

Output:

```
dict_keys(['train_acc', 'val_acc', 'test_acc'])
```

In [41]:
```
# Answer 14
results = {'train_acc': 0.95, 'val_acc': 0.92, 'test_acc': 0.91}
# Hint: Gunakan method untuk menampilkan semua keys dari dictionary
print(results.keys())
```

```
dict_keys(['train_acc', 'val_acc', 'test_acc'])
```

BAGIAN 4: LOOPS - FOR & WHILE (Soal 15-18)

Soal 15: Tampilkan epochs 1-5 dengan format "Epoch X/5: Training..." menggunakan for loop

Output:

```
Epoch 1/5: Training...
Epoch 2/5: Training...
Epoch 3/5: Training...
Epoch 4/5: Training...
Epoch 5/5: Training...
```

In [53]:
```
# Answer 15
# Hint: Gunakan for loop dengan range untuk iterasi dari 1 hingga 5
for i in range (5):
    print(f"Epoch {i+1}/5: Training...")
# ..dst
```

```
Epoch 1/5: Training...
Epoch 2/5: Training...
Epoch 3/5: Training...
Epoch 4/5: Training...
Epoch 5/5: Training...
```

Soal 16: Dari list batch_losses = [0.5, 0.4, 0.3, 0.2], tampilkan setiap loss dengan indeksnya menggunakan for loop

Output:

```
Batch 0: loss = 0.5
Batch 1: loss = 0.4
Batch 2: loss = 0.3
Batch 3: loss = 0.2
```

In [54]:
```python
# Answer 16
batch_losses = [0.5, 0.4, 0.3, 0.2]
# Hint: Gunakan enumerate() untuk mendapat index dan value sekaligus
for i, loss in enumerate(batch_losses):
    print(f"Batch {i}: loss = {loss}")
```

```
Batch 0: loss = 0.5
Batch 1: loss = 0.4
Batch 2: loss = 0.3
Batch 3: loss = 0.2
```

Soal 17: Gunakan while loop untuk menampilkan learning rate yang berkurang: mulai dari lr=0.1, kurangi 0.02 setiap iterasi hingga < 0.04

Output:

```
Current LR: 0.1
Current LR: 0.08
Current LR: 0.06
Current LR: 0.04
```

In [66]:
```python
# Answer 17
# Deklarasikan lr = 0.1
# Hint: Gunakan while loop dengan kondisi yang tepat
# Dalam loop: tampilkan lr, lalu kurangi sebesar 0.02
# Output yang diinginkan:
# Current LR: 0.1
# Current LR: 0.08
# ..dst
Lr = 0.10
while Lr >= 0.02:
    print(f"Current LR:{Lr:.2f}")
    Lr = Lr - 0.020
```

```
Current LR:0.10
Current LR:0.08
Current LR:0.06
Current LR:0.04
```

Soal 18: Gunakan break untuk menghentikan loop ketika model_accuracy = 0.95 tercapai dari list accuracies = [0.70, 0.80, 0.90, 0.95, 0.97]

Output:

```
Accuracy: 0.7
Accuracy: 0.8
```

```
Accuracy: 0.9
Accuracy: 0.95
Target reached! Breaking...
```

In [67]:
```python
# Answer 18
accuracies = [0.70, 0.80, 0.90, 0.95, 0.97]
for acc in accuracies:
    print(f"Accuracy: {acc}")
    if acc == 0.95:  # Cek apakah acc == 0.95
        print("Target reached! Breaking...")
        break  # Gunakan break
```

```
Accuracy: 0.7
Accuracy: 0.8
Accuracy: 0.9
Accuracy: 0.95
Target reached! Breaking...
```

BAGIAN 5: FUNCTIONS & PROCEDURES (Soal 19-21)

Soal 19: Buat fungsi calculate_accuracy(true_labels, predictions) yang menghitung akurasi. Test dengan true_labels=[1,0,1,1], predictions=[1,0,0,1]

Output:

```
Accuracy: 0.75
```

In [71]:
```python
# Answer 19
# Hint: Buat fungsi yang membandingkan setiap elemen dari dua list
# Hitung jumlah yang sama, lalu bagi dengan total untuk mendapat akurasi
# Output yang diinginkan: "Accuracy: 0.75"

def calculate_accuracy(true_labels, predictions):
    correct = 0
    for i in range(len(true_labels)):
        if true_labels[i] == predictions[i]:
            correct += 1

    accuracy = correct / len(true_labels)
    return accuracy

true_labels = [1, 0, 1, 1]
predictions = [1, 0, 0, 1]
print(f"Accuracy: {calculate_accuracy(true_labels, predictions)}")
```

```
Accuracy: 0.75
```

Soal 20: Buat fungsi normalize_data(value, min_val, max_val) untuk normalisasi min-max. Test dengan value=75, min_val=0, max_val=100

Output:

```
Normalized value: 0.75
```

In [73]:
```python
# Answer 20
def normalize_data(value, min_val, max_val):
    # Hint: Formula normalisasi min-max adalah (value - min) / (max - min)
    return (value - min_val) / (max_val - min_val)

normalized = normalize_data(75, 0, 100)
print(f"Normalized value: {normalized}")
```

```
Normalized value: 0.75
```

Soal 21: Buat fungsi apply_activation(x, activation_type='relu') yang menerapkan aktivasi. Test dengan x=-5 dan x=3

Output:

```
ReLU(-5) = 0
ReLU(3) = 3
```

In [75]:
```python
# Answer 21
# Deklarasikan fungsi dengan default parameter activation_type='relu'
# Hint: ReLU mengembalikan nilai x jika positif, 0 jika negatif
# Juga support sigmoid activation (sudah diberikan)
# Output yang diinginkan: "ReLU(-5) = 0" dan "ReLU(3) = 3"

def apply_activation(x, activation_type='relu'):
    if activation_type == 'relu':
        if x >= 0:
            return x
        else:
            return 0

print(f"ReLU(-5) = {apply_activation(-5)}")
print(f"ReLU(3) = {apply_activation(3)}")
```

```
ReLU(-5) = 0
ReLU(3) = 3
```

BAGIAN 6: RECURSIVE FUNCTIONS (Soal 22-23)

Soal 22: Buat fungsi rekursif fibonacci(n) untuk menghitung nilai Fibonacci. Test dengan n=6

Output:

```
Fibonacci(6) = 8
```

In [77]:
```python
# Answer 22
# Hint: Buat fungsi rekursif dengan base case (n <= 1)
# Recursive case: fib(n-1) + fib(n-2)
# Output yang diinginkan: "Fibonacci(6) = 8"
```

```
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)

print(f"Fibonacci(6) = {fibonacci(6)}")
```

```
Fibonacci(6) = 8
```

Soal 23: Buat fungsi rekursif power(base, exponent) untuk menghitung base^exponent. Test dengan base=2, exponent=10

Output:

```
2^10 = 1024
```

In [79]:
```
# Answer 23
# Hint: Buat fungsi rekursif dengan base case (exponent == 0 return 1)
# Recursive case: base * power(base, exponent-1)
# Output yang diinginkan: "2^10 = 1024"

def power(base, exponent):
    if exponent <= 0:
        return 1
    else:
        return base * power(base, exponent-1)

print(f"2^10 = {power(2, 10)}")
```

```
2^10 = 1024
```

BAGIAN 7: CLASSES & OBJECTS (Soal 24-25)

Soal 24: Buat class NeuralNetwork dengan attributes (name, layers, activation). Buat instance dan tampilkan info

Output:

```
Model: CNN, Layers: 5, Activation: ReLU
```

In [90]:
```
# Answer 24
# Hint: Buat class dengan __init__ yang menerima name, layers, activation
# Buat method display_info() yang menampilkan informasi model
# Output yang diinginkan: "Model: CNN, Layers: 5, Activation: ReLU"

class NeuralNetwork:
    def __init__(self, name, layers, activation):
        self.name = name
        self.layers = layers
```

```
        self.activation = activation

    def display_info(self):
        print(f"Model: {self.name}, Layers: {self.layers}, Activation: {self.activa

model = NeuralNetwork("CNN", 5, "ReLU")
model.display_info()
```

```
Model: CNN, Layers: 5, Activation: ReLU
```

Soal 25: Buat class DataPreprocessor dengan method fit(data) dan transform(value) untuk min-max normalization. Test dengan data=[10,20,30]

Output:

```
Normalized 20: 0.5
```

In [91]:
```
# Answer 25
# Hint: Buat class dengan __init__ yang initialize min_val dan max_val
# fit() method: set min_val dan max_val dari data
# transform() method: normalisasi nilai menggunakan formula (value - min) / (max -
# Output yang diinginkan: "Normalized 20: 0.5"

class DataPreprocessor:
    def __init__(self):
        self.min_val = None
        self.max_val = None

    def fit(self, data):
        self.min_val = min(data)
        self.max_val = max(data)

    def transform(self, value):
        return (value - self.min_val) / (self.max_val - self.min_val)

preprocessor = DataPreprocessor()
preprocessor.fit([10, 20, 30])
print(f"Normalized 20: {preprocessor.transform(20)}")
```

```
Normalized 20: 0.5
```

In [ ]:
```

```