AI VIETNAM
@aivietnam.edu.vn

Module 01 - Exercise

Activation Functions

Code

Nguyen Quoc Thai
MSc in Computer Science

Objectives

Python Basic

- Introduction to Python
- Flow Control: IF-ELSE
- Flow Control: For, While Loop
- Python Functions

Practice

- Calculate F-Score Metric
- Calculate Activation Functions
- Calculate Loss Functions
- Estimate Trigonometric Functions

Condition is True

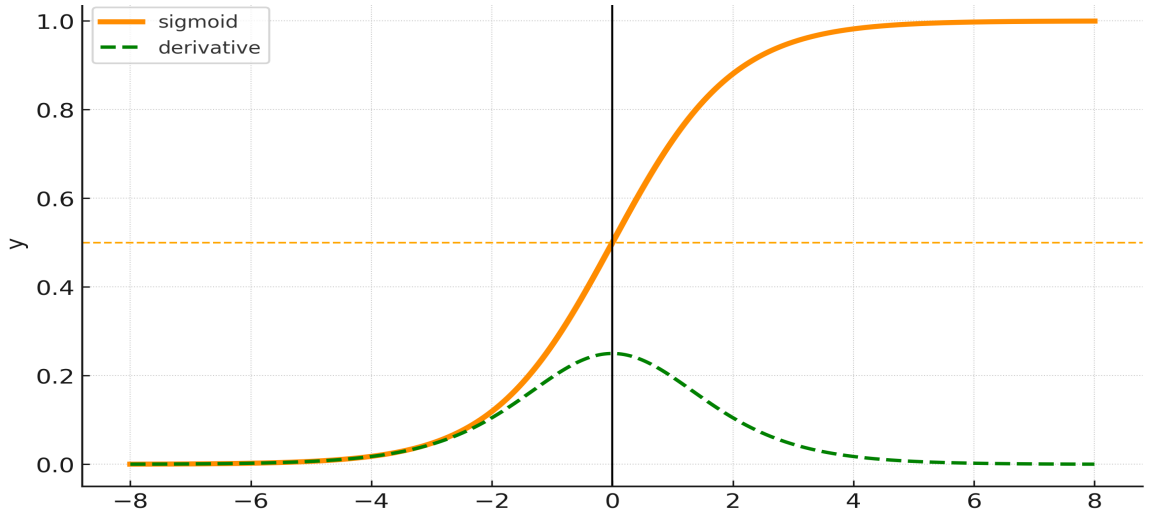
```
number = 10
if number > 0:
        # code

else:
        # code
# code after if
```

Condition is False

```
number = -5
if number > 0:
        # code

else:
        # code
# code after if
```



Outline

SECTION 1

Python Basic

SECTION 2

Practice

Condition is True

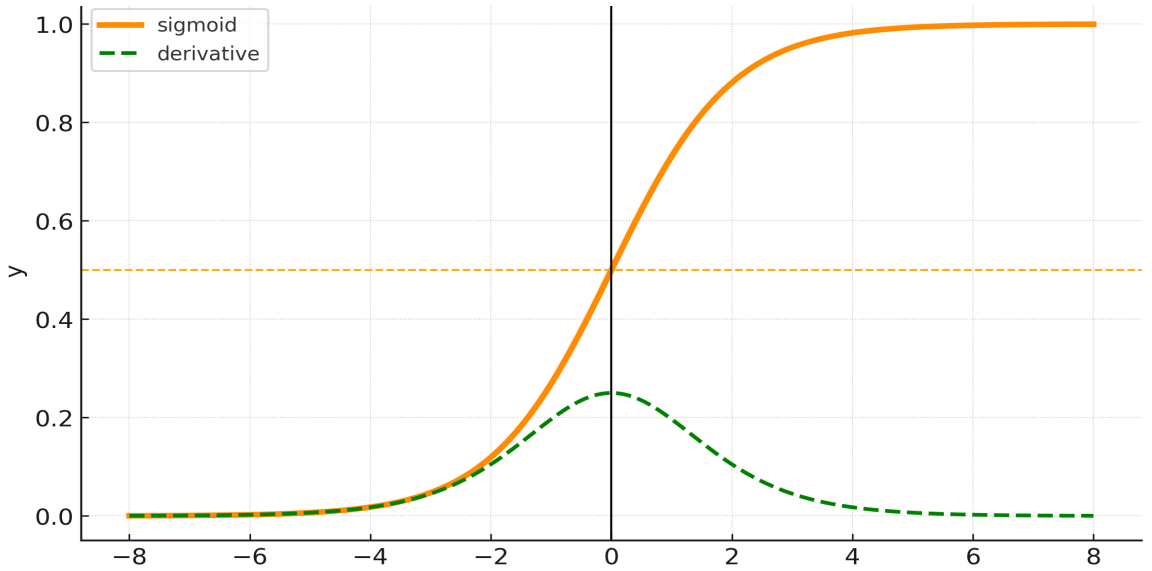
```
number = 10
if number > 0:
        # code

else:
        # code
# code after if
```

Condition is False

```
number = -5
if number > 0:
        # code

else:
        # code
# code after if
```



Python Basic

Introduction to Python

- **Python**
  - Easy to learn and use
  - Supports procedural, object-oriented or functional programming
  - Perfect for complex and quick projects

- **"Hello World"**

```
>>> print("Hello World")
Hello World
```

```
1 print("Hello World!")
```

✓ 0.0s

```
Hello World!
```

https://www.python.org/

Python Basic

Python Variables

- A variable is a container (storage area) to hold data
- Use the assignment operator "=" to assign a value to a variable
- Basic data types: Boolean, str, int, float, complex

```
1 var1 = 'AI Viet Nam'
2 var2 = 3
3 var3 = 15.5
4 var4 = True
5 print(var1, var2, var3, var4)
```

✓ 0.0s

```
AI Viet Nam 3 15.5 True
```

Python Basic

Python Variables

- Type conversion is the process of converting data of one type to another. For example: converting int data to str
- Type conversion: implicit (automatic) and explicit (manual)

```
1  # automatic
2  print(type(var2))
3  print(type(var3))
4  total = var2 + var3
5  print(total, type(total))
```

✓ 0.0s

```
<class 'int'>
<class 'float'>
18.5 <class 'float'>
```

```
1  # manual
2  var3 = int(var3)
3  print(var3, type(var3))
```

✓ 0.0s

```
15 <class 'int'>
```

Python Basic

Basic Input and Output

- Python Output: Use the print() function

```
1 print('AI Viet Nam')
2 print("AI Viet Nam")
3 print("'AI Viet Nam' is awesome.")
4 name = "AI Viet Nam"
5 print(name)
6 print("{} is awesome.".format(name))
7 print(f"{name} is awesome.")
```

✓ 0.0s

```
AI Viet Nam
AI Viet Nam
'AI Viet Nam' is awesome.
AI Viet Nam
AI Viet Nam is awesome.
AI Viet Nam is awesome.
```

- Python Input: Use the input() function

```
1 num = input('Enter a number: ')
2 print('You Entered:', num)
3 print('Data type of num:', type(num))
4
5 num = int(input('Enter a number: '))
6 print('You Entered:', num)
7 print('Data type of num:', type(num))
```

✓ 10.1s

```
You Entered: 7
Data type of num: <class 'str'>
You Entered: 8
Data type of num: <class 'int'>
```

Python Basic

Operators

- Arithmetic: +, - , *, //, %, **
- Assignment: +, +=, -=, *=, /=, %=, **=
- Comparison: ==, !=, >, <, >=, <=
- Logical: and, or, not

```
1  # logical
2  print(True and True)      # True
3  print(True and False)     # False
4  print(True or False)      # True
5  print(not True)           # False
```

✓ 0.0s

```
True
False
True
False
```

```
1  a = 7
2  b = 2
3  # addition
4  print ('Sum: ', a + b)
5
6  # assign the sum of a and b to a
7  a += b       # a = a + b
8  print(a)
9
10 # equal to operator
11 print('a == b =', a == b)
```

✓ 0.0s

```
Sum:  9
9
a == b = False
```

Python Basic

Python if…else Statement

- An if statement executes a block of code only when the specified condition is met.
- If, if…else, if…elif…else

Condition is True

```
number = 10
if number > 0:
        # code

else:
        # code
# code after if
```

Condition is False

```
number = -5
if number > 0:
        # code

else:
        # code
# code after if
```

```
1 num = 10
2 if num > 10:
3     num += 5
4 else:
5     num -= 5
6 print(num)
```

✓ 0.0s

```
5
```

Python Basic

For/While Loop

- A for loop to iterate over sequences such as lists, strings, dictionaries, etc.
- A while loop to repeat a block of code until a certain condition is met.

```python
languages = ['Swift', 'Python', 'Go']
for lang in languages:
    print(lang)

print(list(range(1, 3)))
for i in range(1, 3):
    print(i)
```

✓ 0.0s

```
Swift
Python
Go
[1, 2]
1
2
```

```python
total = 0
i = 0
while i <= 3:
    i += 1
    if i % 2 == 0:
        continue
    total += i
print(total)
```

✓ 0.0s

```
4
```

Python Basic

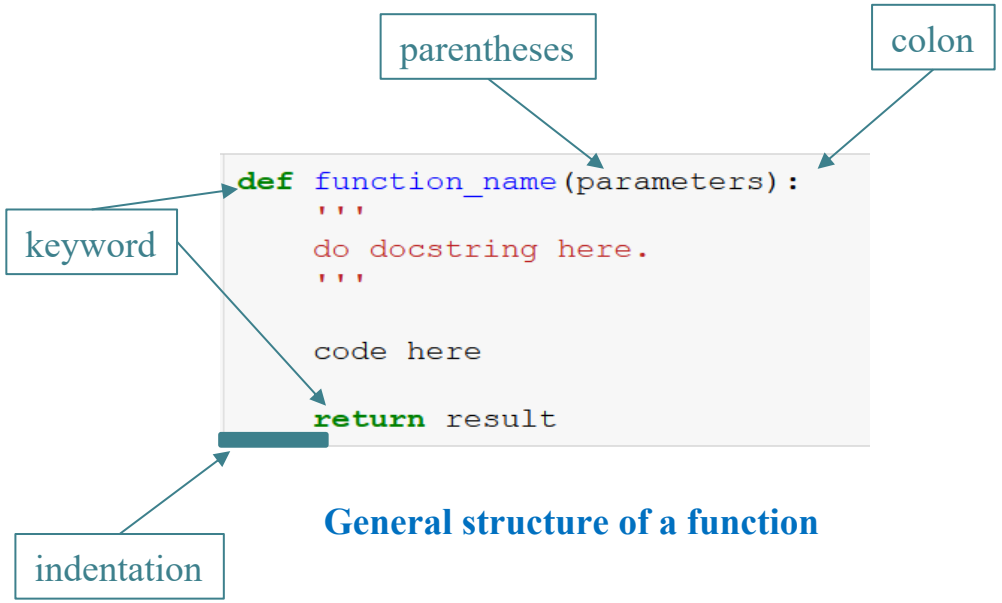
Function

Built-in Functions

print(parameters)

type(parameter)

User-defined Functions



General structure of a function

Outline

SECTION 1

Python Basic

SECTION 2

Practice

Condition is True

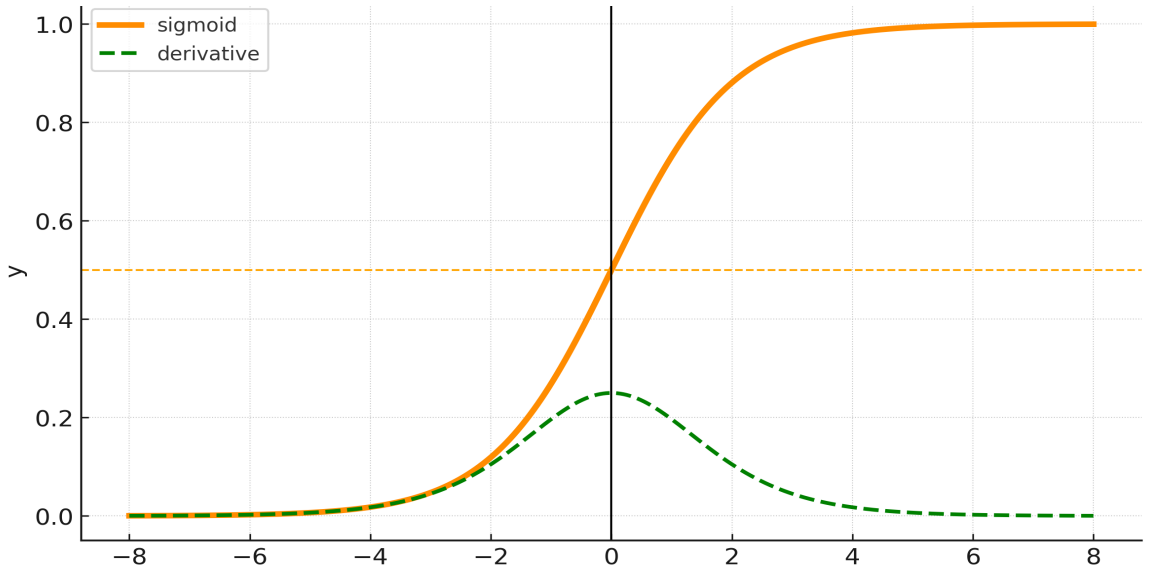
```
number = 10
if number > 0:
        # code

else:
        # code
# code after if
```

Condition is False

```
number = -5
if number > 0:
        # code

else:
        # code
# code after if
```



Practice

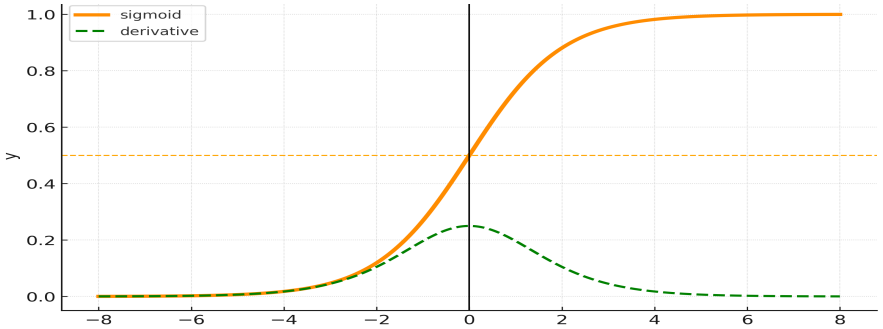
Exercise 1

Calculate F-Score Metric

	Predicted: Dog	Predicted: Cat
Actual: Dog	✅ TP = 1	❌ FN = 0
Actual: Cat	❌ FP = 0	✅ TN = 0

Exercise 2

Calculate Activation Functions



Exercise 3

Calculate Loss Functions

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \qquad \text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Exercise 4

Calculate Trigonometric Functions

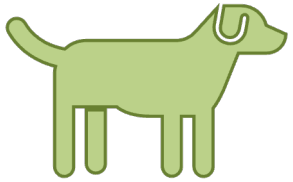
$$sin(x) \approx \sum_{i=0}^{\infty}(-1)^i \frac{x^{(2i+1)}}{(2i+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!}$$

F1-Score

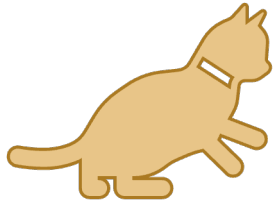
1 Classification Model

- An example of a confusion matrix for a binary classification task with cats and dogs.

DOGs



CATs



- TP (True Positive): Model predicts Dog and it is a Dog
- FP (False Positive): Model predicts Dog, but it is a Cat.
- FN (False Negative): Model predicts Cat, but it is a Dog.
- TN (True Negative): Model predicts Cat, and it is a Cat.

	Predicted: Dog	Predicted: Cat
Actual: Dog	✅ TP = 1	❌ FN = 0
Actual: Cat	❌ FP = 0	✅ TN = 0

F1-Score

1 F1-Score

- F1-score is a metric used to evaluate the accuracy of a classification model
- It is the harmonic mean of:
  Precision (How many predicted positives are actually correct?)
  Recall (How many actual positives were correctly predicted?)

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{all detections}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{all ground truths}}$$

$$F_1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

	Predicted: Dog	Predicted: Cat
Actual: Dog	✅ TP = 1	❌ FN = 0
Actual: Cat	❌ FP = 0	✅ TN = 0

F1-Score

1 F1-Score

- **Exercise 1: Write a function to calculate F1-Score based on TP, FP, FN**

INPUT → **PROGRAM** → OUTPUT

3 values:
TP, FP, FN

3 values:
Precision, Recall, F1-Score

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F}_1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1-Score

1 F1-Score

- **Exercise 1: Write a function to calculate F1-Score based on TP, FP, FN**
- **Test case:**

INPUT → **PROGRAM** → OUTPUT

3 values:
TP, FP, FN

3 values:
Precision, Recall, F1-Score

	TP	FP	FN
1	2	1	1
2	1	3	2

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

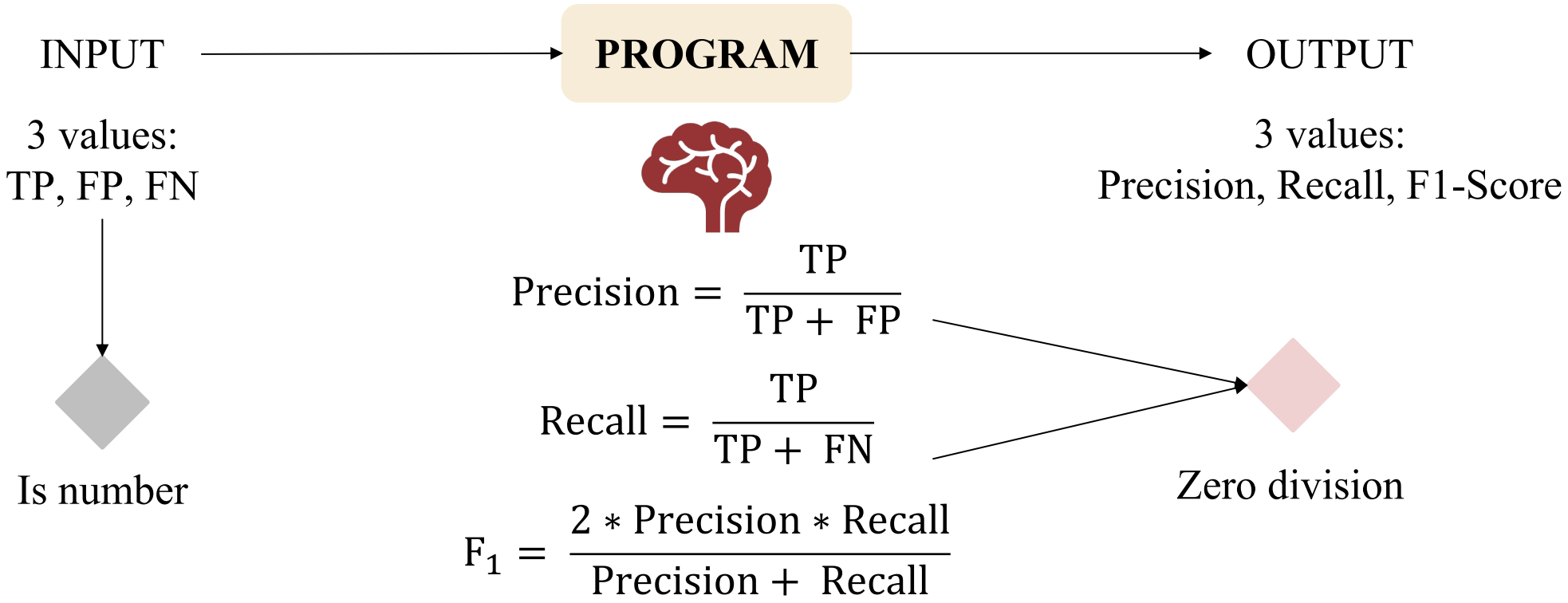
$$\text{F}_1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

	P	R	F1
1	0.67	0.67	0.67
2	0.25	0.33	0.28

F1-Score

1 F1-Score

- **Conditions**



F1-Score

1 F1-Score

- **Exercise 1: Write a function to calculate F1-Score based on TP, FP, FN**

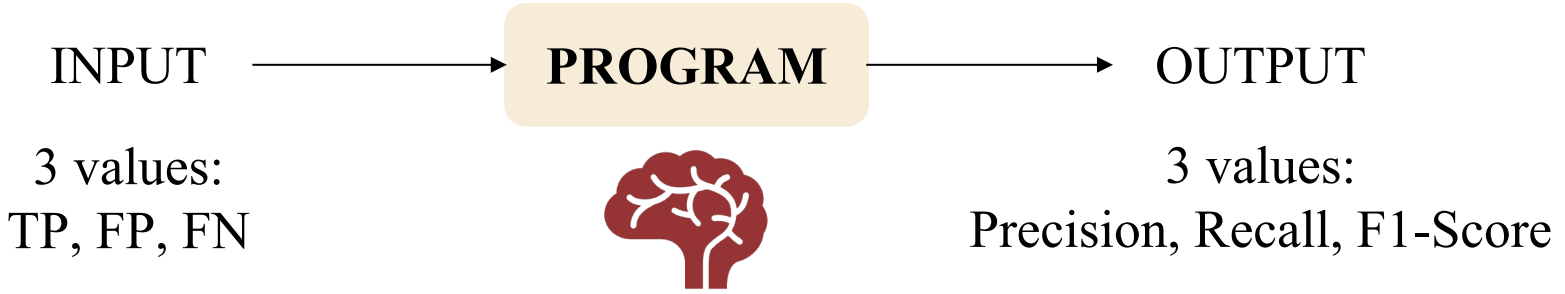
Write a Function

Calculate F1-Score

INPUT:
3 parameters: tp, tp, fn

OUTPUT:
Precision, Recall , F1-score

Condition:
tp, tp, fn: int | zero division



$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$F_1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1-Score

1 F1-Score

- Pseudocode

Write a Function
Calculate F1-Score
INPUT: 3 parameters: tp, tp, fn
OUTPUT: Precision, Recall , F1-score
Condition: tp, tp, fn: int | zero division

```
FUNCTION calc_f1_score(tp, fp, fn
    # check data type
    # check zero-division

    precision = tp/(tp+fp)
    recall = tp/(tp+fn)
    f1_score = 2*(precision*recall)/(precision+recall)

    return precision, recall, f1_score

ENDFUNCTION
```

F1-Score

1 F1-Score

Write a Function

Calculate F1-Score

INPUT:
3 parameters: tp, tp, fn

OUTPUT:
Precision, Recall , F1-score

Condition:
tp, tp, fn: int | zero division

```python
def evaluate_f1_components(tp, fp, fn):
    # Kiểm tra kiểu dữ liệu
    if not isinstance(tp, int):
        print('tp must be int')
        return None

    if not isinstance(fp, int):
        print('fp must be int')
        return None

    if not isinstance(fn, int):
        print('fn must be int')
        return None
```

F1-Score

1 F1-Score

Write a Function

Calculate F1-Score

INPUT:
3 parameters: tp, tp, fn

OUTPUT:
Precision, Recall , F1-score

Condition:
tp, tp, fn: int | zero division

```python
    # Kiểm tra giá trị không âm
    if tp < 0 or fp < 0 or fn < 0:
        print('tp, fp, and fn must be non-negative integers')
        return None

    # Tính precision và recall với xử lý chia cho 0
    precision = tp / (tp + fp) if (tp + fp) > 0 else 0.0
    recall = tp / (tp + fn) if (tp + fn) > 0 else 0.0

    # Tính F1 Score
    f1_score = 2*(precision*recall)/(
        precision+recall) if (precision+recall) > 0 else 0.0

    return precision, recall, f1_score
```

Practice

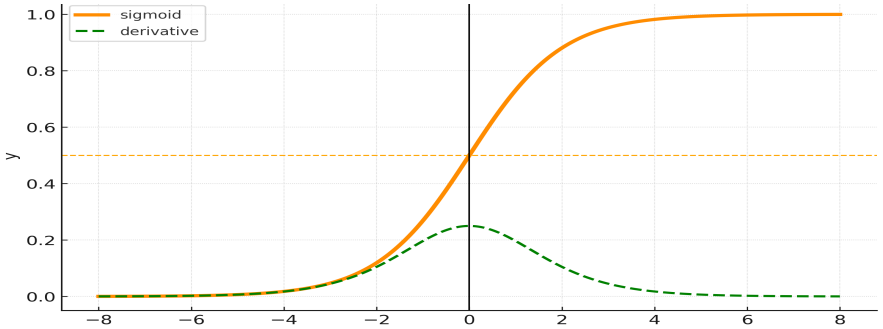
Exercise 1

Calculate F-Score Metric

	Predicted: Dog	Predicted: Cat
Actual: Dog	✅ TP = 1	❌ FN = 0
Actual: Cat	❌ FP = 0	✅ TN = 0

Exercise 2

Calculate Activation Functions



Exercise 3

Calculate Loss Functions

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \qquad \text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Exercise 4

Calculate Trigonometric Functions

$$sin(x) \approx \sum_{i=0}^{\infty}(-1)^i \frac{x^{(2i+1)}}{(2i+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!}$$

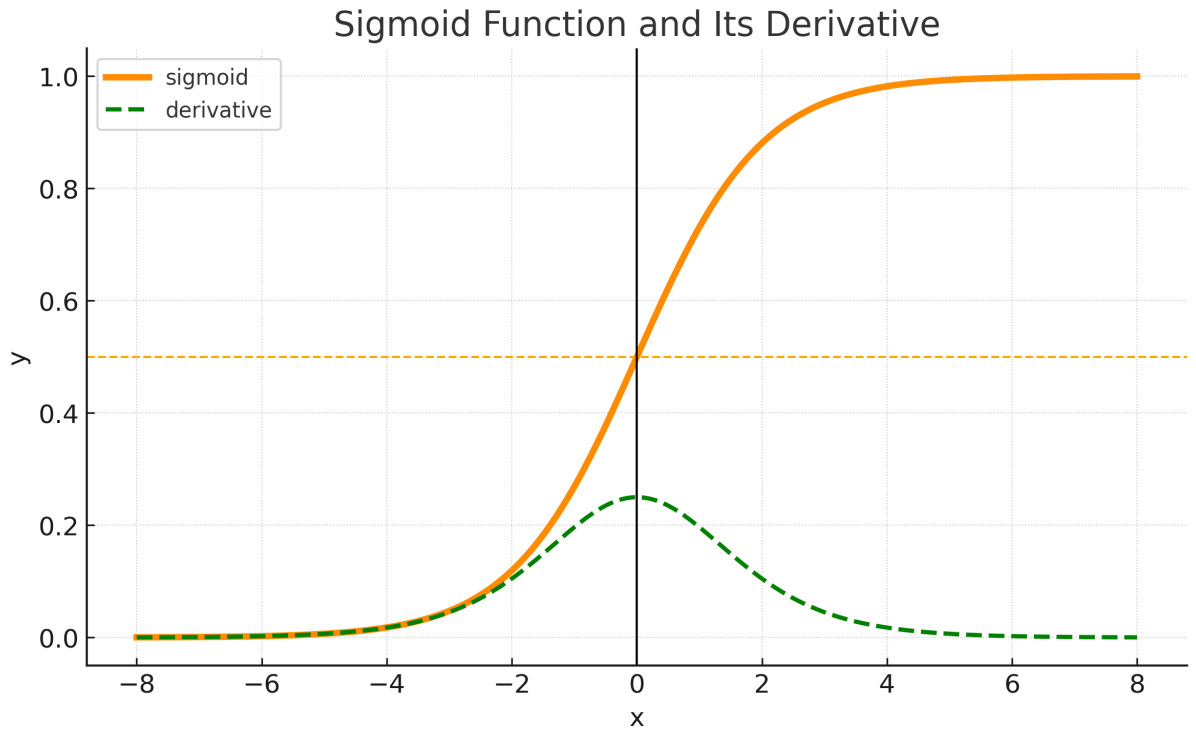
Activation Functions

2 Sigmoid Function

$$\boldsymbol{sigmoid(x) = \frac{1}{1 + e^{-x}}}$$

$$sigmoid(3) = \frac{1}{1 + e^{-3}} = 0.95$$

	1	5	-4	3
Sigmoid(x)	0.731	0.993	0.017	0.94

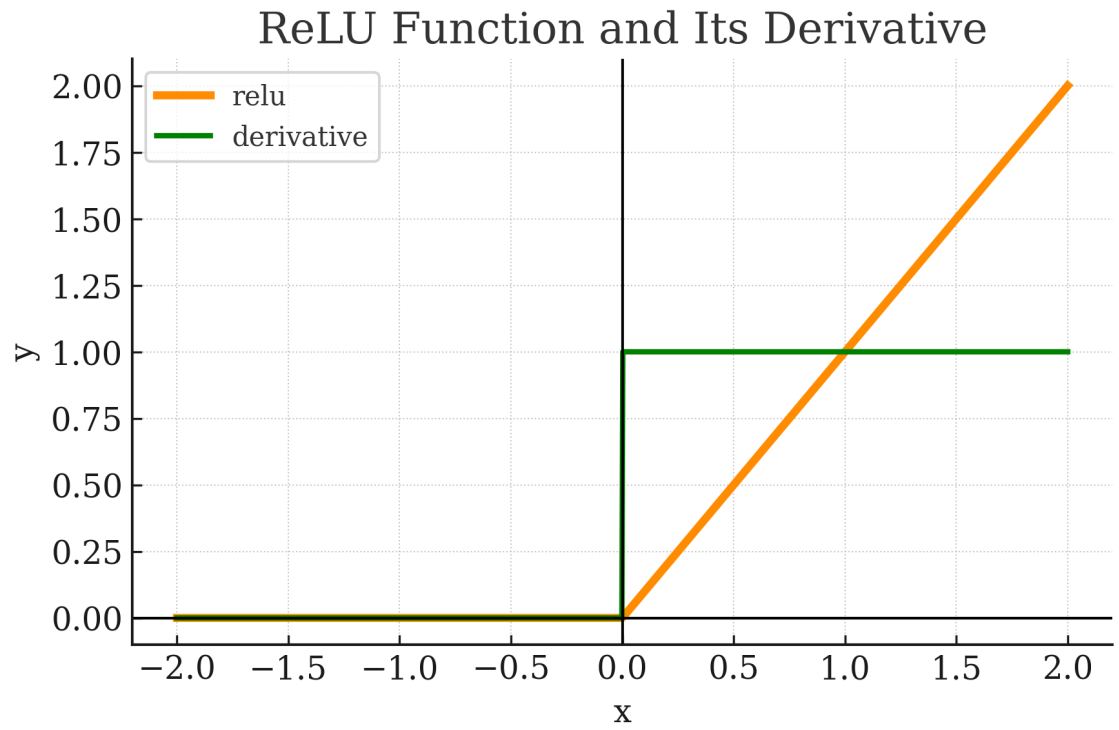


Activation Functions

2 Relu Function

$$relu(x) = \begin{cases} 0 & if\ x \leq 0 \\ x & if\ x > 0 \end{cases}$$

	1	5	-4	3
relu(x)	1	5	0	3



Activation Functions

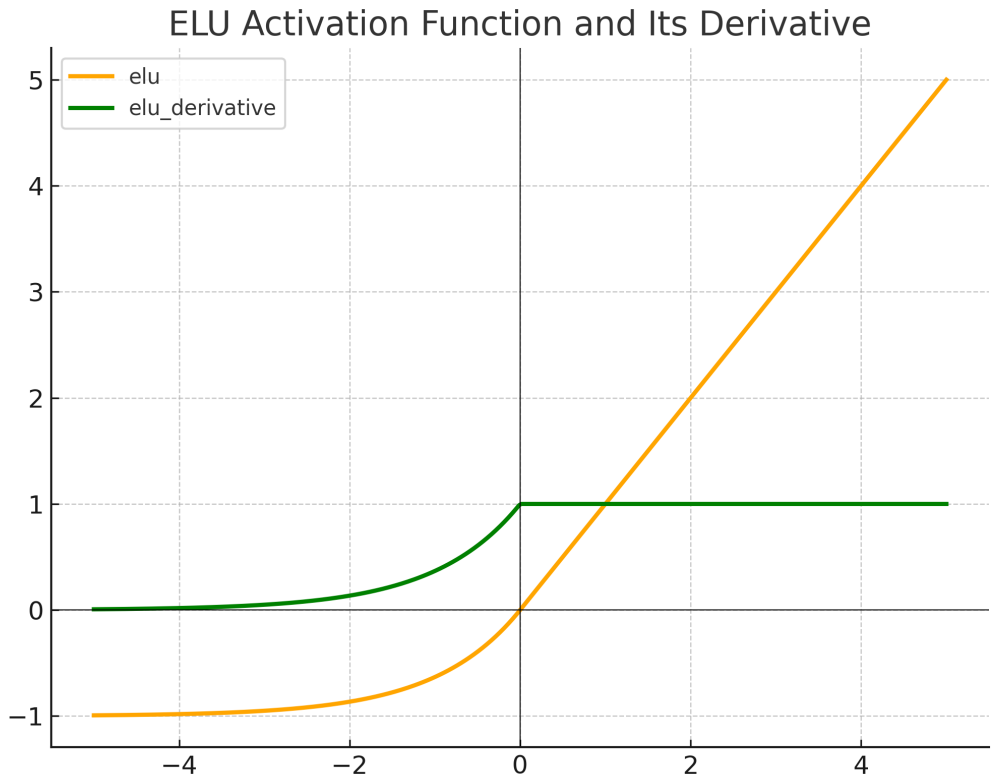
2 ELU Function

$$ELU(x) = \begin{cases} \alpha(e^x - 1) & if\ x \leq 0 \\ x & if\ x > 0 \end{cases}$$

$$\alpha = 0.01$$

$$ELU(-4) = 0.01(e^{-4} - 1) = -0.0098$$

	1	5	-4	3
ELU(x)	1	5	-0.0098	3



Activation Functions

2 Activation Functions

- **Exercise 2: Write a function to calculate activate functions**

INPUT → **PROGRAM** → OUTPUT

2 values:
x and activation name

x is a number
name = {sigmoid, relu, elu}

1 value:
F(x) with F: sigmoid, relu, ELU

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

$$relu(x) = \begin{cases} 0 & if\ x \leq 0 \\ x & if\ x > 0 \end{cases}$$

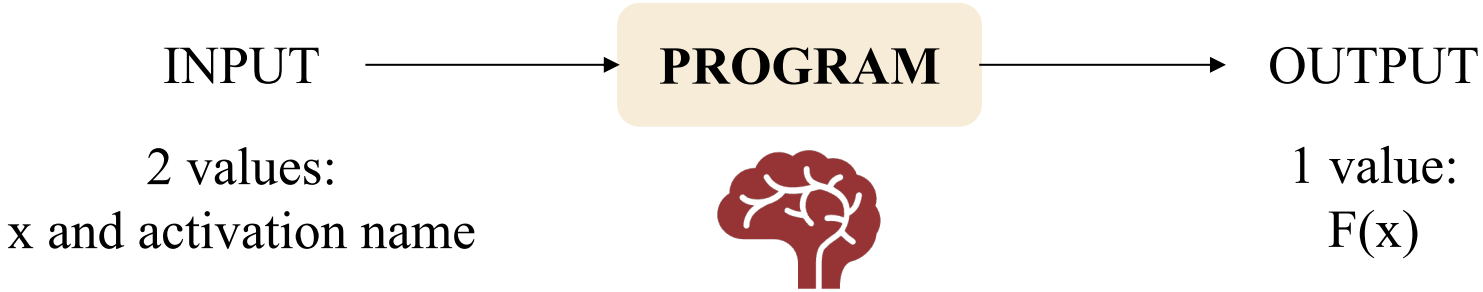
$$ELU(x) = \begin{cases} \alpha(e^x - 1) & if\ x \leq 0 \\ x & if\ x > 0 \end{cases}$$

Activation Functions

2 Activation Functions

- **Exercise 2: Write a function to calculate activate functions**

Write a Function
Calculate activation function
INPUT: 2 paras: x, function name
OUTPUT: 1 value:
Condition 1: x: float
Condition 2: activation name in {sigmoid, relu, elu}



$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

$$relu(x) = \begin{cases} 0 & if\ x \leq 0 \\ x & if\ x > 0 \end{cases}$$

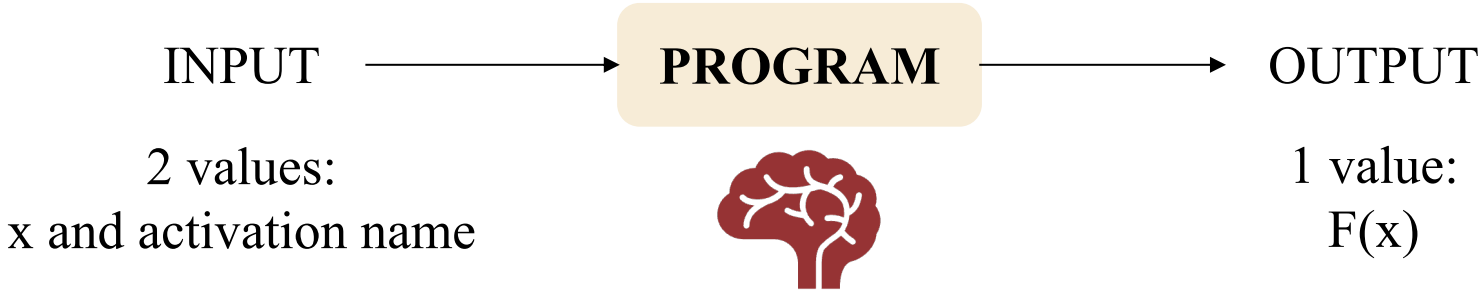
$$ELU(x) = \begin{cases} \alpha(e^{x} - 1) & if\ x \leq 0 \\ x & if\ x > 0 \end{cases}$$

Activation Functions

2 Activation Functions

- **Exercise 2: Write a function to calculate activate functions**
- **Test case:**

Write a Function
Calculate activation function
INPUT: 2 paras: x, function name
OUTPUT: 1 value:
Condition 1: x: float
Condition 2:activation name in {sigmoid, relu, elu}



I		O
1	sigmoid	0.731
5	relu	5
-4	elu	-0.0098
2	tanh	---

Activation Functions

2 Activation Functions

Write a Function

Calculate activation function

INPUT:
2 paras: x, function name

OUTPUT:
1 value:

Condition 1:
x: float

Condition 2: activation name in {sigmoid, relu, elu}

```python
import math
def is_number(n):
    try:
        float(n)   # Type-casting the string to `float`.
                   # If string is not a valid `float`,
                   # it'll raise `ValueError` exception
    except ValueError:
        return False
    return True
assert is_number(3) == 1.0
assert is_number('-2a') == 0.0
print(is_number(1))
print(is_number('n'))
```

```
True
False
```

Activation Functions

2 Activation Functions

Write a Function
Calculate activation function
INPUT: 2 paras: x, function name
OUTPUT: 1 value:
Condition 1: x: float
Condition 2: activation name in {sigmoid, relu, elu}

```python
import math

def calc_sig(x):
    """
    Tính hàm sigmoid: σ(x) = 1 / (1 + e^(-x))
    """
    return 1./(1+math.e**(-x))

assert round(calc_sig(3), 2)==0.95

calc_sig(1)
```

✓ 0.0s

```
0.7310585786300049
```

Activation Functions

2 Activation Functions

Write a Function

Calculate activation function

INPUT:
2 paras: x, function name

OUTPUT:
1 value:

Condition 1:
x: float

Condition 2: activation name in {sigmoid, relu, elu}

```python
def calc_relu(x):
    """
    Tính hàm ReLU:
    ReLU(x) = max(0, x)
    """
    if x<=0:
        result = 0.0
    else:
        result = x
    return float(result)

calc_relu(5)
```

✓ 0.0s

```
5.0
```

Activation Functions

2 Activation Functions

Write a Function

Calculate activation function

INPUT:
2 paras: x, function name

OUTPUT:
1 value:

Condition 1:
x: float

Condition 2: activation name in {sigmoid, relu, elu}

```python
import math

def calc_elu(x):
    """
    Tính hàm ELU (Exponential Linear Unit):
    ELU(x) = x                    nếu x >= 0
           = alpha * (e^x - 1)    nếu x < 0
    """
    alpha = 0.01
    result = None
    if x < 0:
        result = alpha*(math.e**x - 1)
    else:
        result = x
    return result
assert round(calc_elu(1))==1

calc_elu(-4)
```

✓ 0.0s

```
-0.009816843611112657
```

Activation Functions

2 Activation Functions

Write a Function

Calculate activation function

INPUT:
2 paras: x, function name

OUTPUT:
1 value:

Condition 1:
x: float

Condition 2: activation name in {sigmoid, relu, elu}

```python
def calc_activation_func(x, act_name):
    """
    Tính hàm kích hoạt cho x dựa trên act_name:
    'relu', 'sigmoid', hoặc 'elu'.
    """
    result = None
    if act_name == 'relu':
        result = calc_relu(x)

    elif act_name == 'sigmoid':
        result = calc_sig(x)

    elif act_name == 'elu':
        result = calc_elu(x)

    else:
        result = None

    return result

assert calc_activation_func(x = 1, act_name='relu') == 1
calc_activation_func(1, "sigmoid")
```

✓ 0.0s

```
0.7310585786300049
```

Activation Functions

2 Activation Functions

Write a Function

Calculate activation function

INPUT:
2 paras: x, function name

OUTPUT:
1 value:

Condition 1:
x: float

Condition 2: activation name in {sigmoid, relu, elu}

```python
def interactive_activation_function():
    x = input('Input x = ')
    if not is_number(x):
        print('x must be a number')
        return # exit()

    act_name = input('Input activation function (sigmoid|relu|elu): ')
    x = float(x)
    result = calc_activation_func(x, act_name)
    if result is None:
        print(f'{act_name} is not supportted')
    else:
        print(f'{act_name}: f({x}) = {result}')
```

✓ 0.0s

```python
interactive_activation_function()
```

```
Input x = 3
Input activation Function (sigmoid|relu|elu): sigmoid
sigmoid: f(3.0) = 0.9525741268224331
```

Practice

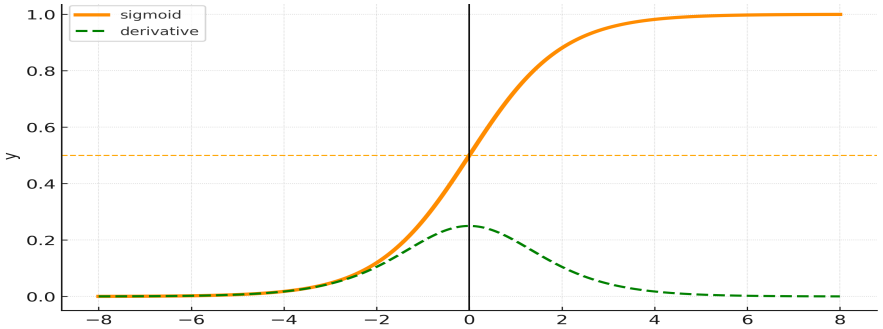
Exercise 1

Calculate F-Score Metric

	Predicted: Dog	Predicted: Cat
Actual: Dog	✅ TP = 1	❌ FN = 0
Actual: Cat	❌ FP = 0	✅ TN = 0

Exercise 2

Calculate Activation Functions



Exercise 3

Calculate Loss Functions

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \qquad \text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Exercise 4

Calculate Trigonometric Functions

$$sin(x) \approx \sum_{i=0}^{\infty}(-1)^i \frac{x^{(2i+1)}}{(2i+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!}$$

Loss Functions

3 Loss Functions

- **Exercise 3: Write a function to calculate loss functions**

INPUT → **PROGRAM** → OUTPUT

2 values:
n and loss name

1 value: loss

n is int number
name = {mae, mse, rmse}
$y_i, \hat{y}_i$: random value
with $i \in \{0, \dots, n\}$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{MSE}$$

Loss Functions

3 Loss Functions

- **Exercise 3: Write a function to calculate loss functions**

INPUT → **PROGRAM** → OUTPUT

2 values:
n and loss name

n is int numner
name = {mae, mse, rmse}
$y_i, \hat{y}_i$: random value
with $i \in \{0, \dots, n\}$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{MSE}$$

1 value: loss

n = 3	y_i	$\hat{y}_i$
loop 0	0	1
loop 1	2	1
loop 2	3	2

MAE = 1
MSE = 1
RMSE = 1

Loss Functions

3 Loss Functions

- **Exercise 3: Write a function to calculate loss functions**

Write a Function
Calculate loss function
INPUT: 2 paras: n, loss name
OUTPUT: 1 value: loss
Condition 1: n: int
Condition 2: loss name in {mae, mse, rmse}

INPUT → **PROGRAM** → OUTPUT

2 values:
n and loss name

1 value:
F(x)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{MSE}$$

Loss Functions

3 Loss Functions

Write a Function

Calculate loss function

INPUT:
2 paras: n, loss name

OUTPUT:
1 value: loss

Condition 1:
n: int

Condition 2: loss name in {mae, mse, rmse}

```python
def calc_ae(y, y_hat):
    """
    Tính sai số tuyệt đối (Absolute Error)
    giữa giá trị thực tế và giá trị dự đoán.

    Tham số:
    y (float hoặc int): Giá trị thực tế (ground truth).
    y_hat (float hoặc int): Giá trị dự đoán.

    Trả về:
    float: Giá trị tuyệt đối của hiệu giữa y và y_hat.
    """
    return abs(y - y_hat)


y = 1
y_hat = 6
assert calc_ae(y, y_hat)==5

y = 2
y_hat = 9
print(calc_ae(y, y_hat))
```

✓ 0.0s

Loss Functions

3 Loss Functions

Write a Function

Calculate loss function

INPUT:
2 paras: n, loss name

OUTPUT:
1 value: loss

Condition 1:
n: int

Condition 2: loss name in {mae, mse, rmse}

```python
def calc_se(y, y_hat):
    """
    Tính sai số bình phương (Squared Error)
    giữa giá trị thực tế và giá trị dự đoán.

    Tham số:
    y (float hoặc int): Giá trị thực tế (ground truth).
    y_hat (float hoặc int): Giá trị dự đoán.

    Trả về:
    float: Bình phương của hiệu giữa y và y_hat.
    """
    return (y-y_hat)**2
y = 4
y_hat = 2
assert calc_se(y, y_hat) == 4

print(calc_se(2, 1))
```

✓ 0.0s

```
1
```

Loss Functions

3 Loss Functions

Write a Function

Calculate loss function

INPUT:
2 paras: n, loss name

OUTPUT:
1 value: loss

Condition 1:
n: int

Condition 2: loss name in {mae, mse, rmse}

```python
import random

def cal_activation_function():
    num_samples = input('Input number of samples (integer number) which are generated: ')
    if not num_samples.isnumeric():#Hàm isnumeric() trong Python trả về true nếu một chuỗi dạng Unicode chỉ chứa các ký tự số,
    #nếu không là false.
        print("number of samples must be an integer number")
        return # exit()
    loss_name = input('Input loss name: ')
```

Loss Functions

3 Loss Functions

Write a Function

Calculate loss function

INPUT:
2 paras: n, loss name

OUTPUT:
1 value: loss

Condition 1:
n: int

Condition 2: loss name in {mae, mse, rmse}

```
    # giả sử người dùng luôn nhập đúng MAE, MSE hoặc RMSE
    final_loss = 0
    num_samples = int(num_samples)
    for i in range(num_samples):
        pred_sample = random.uniform(0,10)
        target_sample = random.uniform(0,10)

        if loss_name == 'MAE':
            loss = calc_ae(pred_sample, target_sample)
        elif loss_name == 'MSE' or loss_name == 'RMSE':
            loss = calc_se(pred_sample, target_sample)
        # hoặc trả về thông báo loss không có
        final_loss += loss
        print(f'loss_name: {loss_name}, sample: {i}: pred: {round(pred_sample,
        2)} target: {round(target_sample,2)} loss: {round(loss,2)}')

    final_loss /= num_samples
    if loss_name == 'RMSE':
        final_loss = math.sqrt(final_loss)
    print(f'final {loss_name}: {final_loss}')
    return final_loss
```

Loss Functions

3 Loss Functions

Write a Function

Calculate loss function

INPUT:
2 paras: n, loss name

OUTPUT:
1 value: loss

Condition 1:
n: int

Condition 2: loss name in {mae, mse, rmse}

```
1  final_loss = cal_activation_function()
2  final_loss
```

✓ 3.6s

```
loss_name: MAE, sample: 0: pred: 5.92 target: 6.96 loss: 1.05
loss_name: MAE, sample: 1: pred: 0.1 target: 5.74 loss: 5.64
loss_name: MAE, sample: 2: pred: 0.94 target: 2.1 loss: 1.16
final MAE: 2.6156580640656397

2.6156580640656397
```


QUIZ TIME

Practice

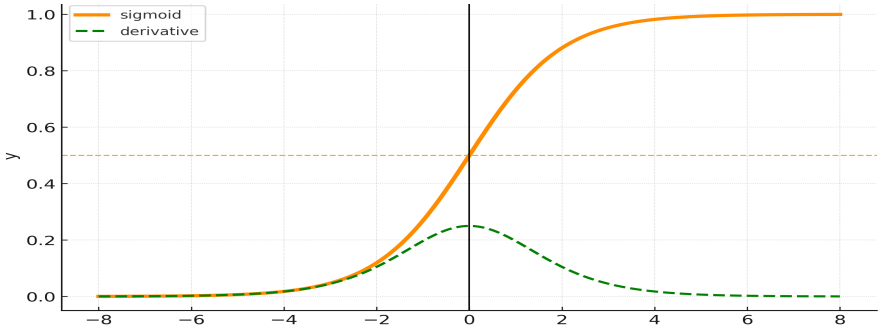
Exercise 1

Calculate F-Score Metric

	Predicted: Dog	Predicted: Cat
Actual: Dog	✅ TP = 1	❌ FN = 0
Actual: Cat	❌ FP = 0	✅ TN = 0

Exercise 2

Calculate Activation Functions



Exercise 3

Calculate Loss Functions

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \qquad \text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Exercise 4

Calculate Trigonometric Functions

$$sin(x) \approx \sum_{i=0}^{\infty}(-1)^i \frac{x^{(2i+1)}}{(2i+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!}$$

Trigonometric Functions

4 Trigonometric Functions

- **Exercise 4: Write a function to estimate trigonometric functions**

$$sin(x) \approx \sum_{i=0}^{\infty} (-1)^i \frac{x^{(2i+1)}}{(2i+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots$$

$$cos(x) \approx \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \frac{x^{10}}{10!} + \cdots$$

$$sinh(x) \approx \sum_{i=0}^{\infty} \frac{x^{(2i+1)}}{(2i+1)!} = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \frac{x^9}{9!} + \cdots$$

$$cosh(x) \approx \sum_{i=0}^{\infty} \frac{x^{2i}}{(2i)!} = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \frac{x^8}{8!} + \frac{x^{10}}{10!} + \cdots$$

Trigonometric Functions

4 Trigonometric Functions

- **Exercise 4: Write a function to estimate sin function**

x: number
n: represents the number of terms used from the infinite series to approximate sin(x)

$$sin(x) \approx \sum_{i=0}^{\infty} (-1)^i \frac{x^{(2i+1)}}{(2i+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots$$

$$\sin(x) \approx \sum_{i=0}^{n} (-1)^i \frac{x^{(2i+1)}}{(2i+1)!}$$

Trigonometric Functions

4 Trigonometric Functions

- **Exercise 4: Write a function to estimate sin function**

Write a Function
Calculate sin function
INPUT: 2 paras: x, n
OUTPUT: 1 value: sin(x)
Function: factorial_fcn()

$$sin(x) \approx \sum_{i=0}^{\infty} (-1)^i \frac{x^{(2i+1)}}{(2i+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots$$

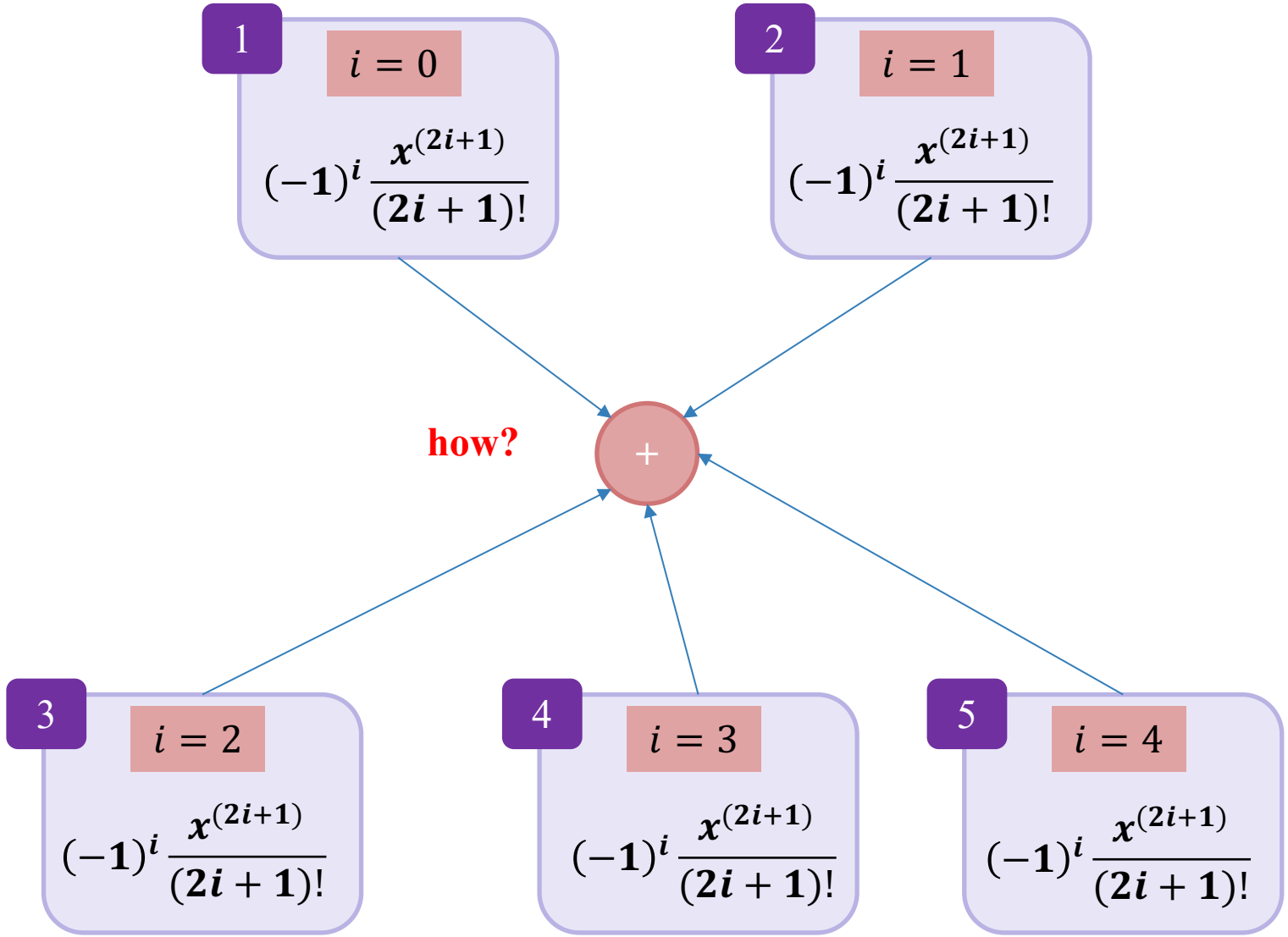
$$\sin(x) \approx \sum_{i=0}^{n} (-1)^i \frac{x^{(2i+1)}}{(2i+1)!}$$

Trigonometric Functions

4 Trigonometric Functions

$$\sin(x) \approx \sum_{i=0}^{n} (-1)^i \frac{x^{(2i+1)}}{(2i+1)!}$$

```
n = 4
for i in range(n+1):
    …
```



Trigonometric Functions

4 Trigonometric Functions

$$\sin(x) \approx \sum_{i=0}^{n} (-1)^i \frac{x^{(2i+1)}}{(2i+1)!}$$

$$\sin(x) \approx \sum_{i=0}^{n} (-1)^i \frac{x^{(2i+1)}}{(2i+1)!}$$

```
FUNCTION factorial_fcn(value)
    result = 1
    FOR i start at 1 TO value
          result *= i
    RETURN result
ENDFUNCTION
```

```
FUNCTION factorial_fcn(value)
    result = 1
    FOR i start at 0 TO value -1
          result *= (i+1)
    RETURN result
ENDFUNCTION
```

```
FOR i start at 0 TO n
    coef = (-1)**I
    num = x**(2*i+1)
    denom = factorial_fnc(2*i+1)
    result = coef*(num/denom) + result
    RETURN result
```

Trigonometric Functions

4 Trigonometric Functions

$$\sin(x) \approx \sum_{i=0}^{n} (-1)^i \frac{x^{(2i+1)}}{(2i+1)!}$$

```python
def factorial_fcn(x):
    """
    Compute the factorial of a non-negative integer x.

    Parameters:
    x (int): The input integer (x >= 0)

    Returns:
    int: The factorial of x (i.e., x!)
    """
    res = 1
    for i in range(x):
        res *= (i + 1)
    return res

factorial_fcn(x=4)
```

✓ 0.0s

Trigonometric Functions

4 Trigonometric Functions

$$\sin(x) \approx \sum_{i=0}^{n} (-1)^i \frac{x^{(2i+1)}}{(2i+1)!}$$

```python
def approx_sin(x, n):
    """
    Approximate the sine of x using the Taylor series expansion.

    Parameters:
    x (float): The input angle in radians.
    n (int): Number of terms in the Taylor series expansion.

    Returns:
    float: Approximate value of sin(x) using n+1 terms.
    """
    sin_approx = 0
    for i in range(n + 1):
        coef = (-1)**i
        num = x**(2 * i + 1)
        denom = factorial_fcn(2 * i + 1)
        sin_approx += coef * (num / denom)

    return sin_approx
```

✓ 0.0s

```python
print(round(approx_sin(x=3.14, n=10), 4))
```

✓ 0.0s

```
0.0016
```

Trigonometric Functions

4 Trigonometric Functions

$$cos(x) \approx \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!}$$

```python
def approx_cos(x, n):
    """
    Approximate the cosine of x using the Taylor series expansion.
    Parameters:
    x (float): The input angle in radians.
    n (int): Number of terms in the Taylor series expansion.
    Returns:
    float: Approximate value of cos(x) using n+1 terms.
    """
    cos_approx = 0
    for i in range(n + 1):
        coef = (-1) ** i
        num = x ** (2 * i)
        denom = factorial_fcn(2 * i)
        cos_approx += coef * (num / denom)

    return cos_approx

# Test
assert round(approx_cos(x=1, n=10), 2) == 0.54
```

✓ 0.0s

```python
print(round(approx_cos(x=3.14, n=10), 2))
```

✓ 0.0s

```
-1.0
```

Trigonometric Functions

4 Trigonometric Functions

$$sinh(x) \approx \sum_{i=0}^{\infty} \frac{x^{(2i+1)}}{(2i+1)!}$$

```python
def approx_sinh(x, n):
    """
    Approximate the hyperbolic sine of x using the Taylor series expansion.
    Parameters:
    x (float): The input value.
    n (int): Number of terms in the Taylor series expansion.
    Returns:
    float: Approximate value of sinh(x) using n+1 terms.
    """

    sinh_approx = 0
    for i in range(n + 1):
        num = x ** (2 * i + 1)
        denom = factorial_fcn(2 * i + 1)
        sinh_approx += num / denom
    return sinh_approx

# Test
assert round(approx_sinh(x=1, n=10), 2) == 1.18
```

✓ 0.0s

```python
print(round(approx_sinh(x=3.14, n=10), 2))
```

✓ 0.0s

```
11.53
```

Trigonometric Functions

4 Trigonometric Functions

$$cosh(x) \approx \sum_{i=0}^{\infty} \frac{x^{2i}}{(2i)!}$$

```python
def approx_cosh(x, n):
    """
    Approximate the hyperbolic cosine of x using the Taylor series.
    Parameters:
    x (float): The input value.
    n (int): Number of terms in the Taylor series expansion.
    Returns:
    float: Approximate value of cosh(x) using n+1 terms.
    """
    cosh_approx = 0
    for i in range(n + 1):
        num = x ** (2 * i)
        denom = factorial_fcn(2 * i)
        cosh_approx += num / denom

    return cosh_approx

# Test
assert round(approx_cosh(x=1, n=10), 2) == 1.54
```

✓ 0.0s

```python
print(round(approx_cosh(x=3.14, n=10), 2))
```

✓ 0.0s

```
11.57
```

Practice

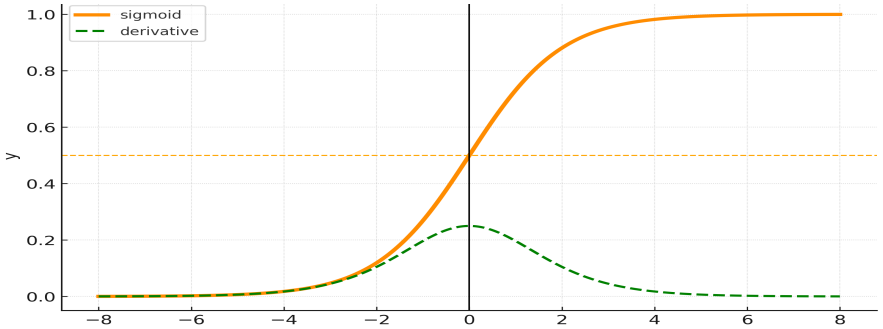
Exercise 1

Calculate F-Score Metric

	Predicted: Dog	Predicted: Cat
Actual: Dog	✅ TP = 1	❌ FN = 0
Actual: Cat	❌ FP = 0	✅ TN = 0

Exercise 2

Calculate Activation Functions



Exercise 3

Calculate Loss Functions

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \qquad \text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Exercise 4

Calculate Trigonometric Functions

$$sin(x) \approx \sum_{i=0}^{\infty}(-1)^i \frac{x^{(2i+1)}}{(2i+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!}$$

Objectives

Python Basic

- Introduction to Python
- Flow Control: IF-ELSE
- Flow Control: For, While Loop
- Python Functions

Practice

- Calculate F-Score Metric
- Calculate Activation Functions
- Calculate Loss Functions
- Estimate Trigonometric Functions

Condition is True

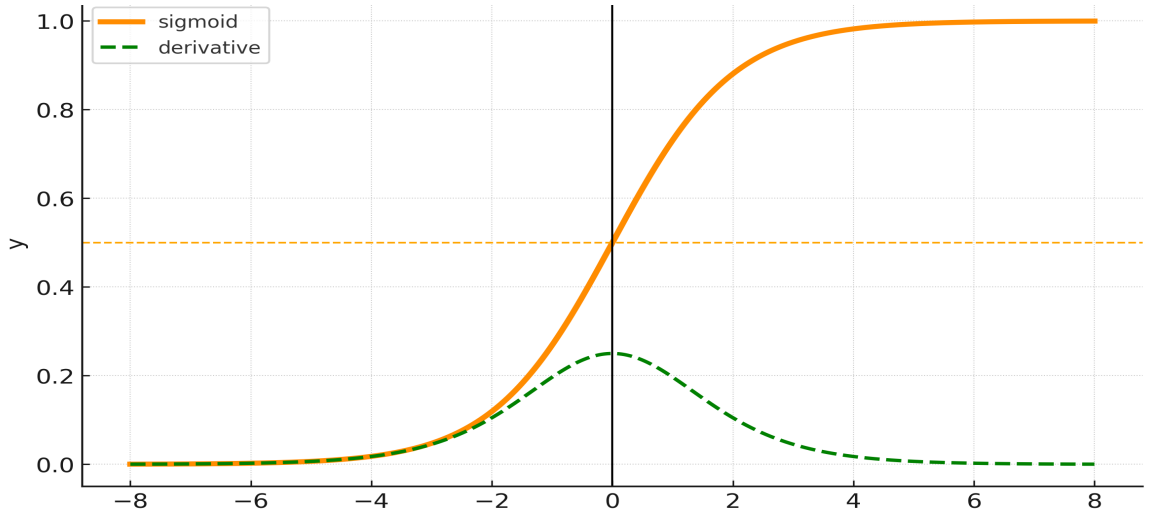
```
number = 10
if number > 0:
        # code

else:
        # code
# code after if
```

Condition is False

```
number = -5
if number > 0:
        # code

else:
        # code
# code after if
```



Thanks!

Any questions?