AI VIETNAM
All-in-One Course

Python Basics
(TA Session)

Nguyễn Thọ Anh Khoa
Ph.D. Student

Year 2024

Objectives

- ✔ Introduction of print() function in Python
- ✔ Application of for loop in Python
- ✔ Introduction basic steps to program with Python
- ✔ Application of if, else, and string in Python
- ✔ Implementation of functions that evaluate classification and regression tasks
- ✔ Implementation of functions that approximate trigonometric functions

```
    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)
        #else : catch error
        final_loss += loss
        print(f'loss_name: {loss_name}, sample: {i}: pred: {pred_sample}

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

Outline

- Python print() function
- Exercise1: Viết function đánh giá classification model bằng f1-score
- Exercise2: Viết function tính kết quả theo các activation function
- Exercise3: Viết function chọn regression loss function để tính loss
- Exercise4: Viết các function để ước lượng các hàm số
- Exercise5: Viết function thực hiện hàm MD_nRE

Python print() function

Python print() function

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

- **objects:** Các objects muốn in ra. * thể hiện có thể in được nhiều hơn 1 object
- **sep (optional):** Khi in nhiều object, các ký tự mong muốn dùng để phân chia các objects. Default là ‘ ’
- **end (optional):** Các ký tự muốn in ở cuối cùng (cuối line). Default là ‘\n’
- **file (optional):** Phải là object với write method. Default là sys.stdout
- **flush (optional):** nếu True output là flushed hoặc buffered nếu False
- **Note:** Các arguments **sep, end, file, flush** là keyword arguments
- **Note:** Các objects sẽ được convert sang string type

AI VIETNAM
All-in-One Course

Python print() function

```
 99  # print 1 object
100  print("Xin Chào AI VN")
101
102  b = 5
103  # print 2 object
104  print("a =", b)
105
106  # print 4 object
107  c = 999.9
108  print('a =', b, "c =", c)
```

```
Xin Chào AI VN
a = 5
a = 5 c = 999.9
```

- sep = ' ' : Giữa các object có một space
- end = '\n' : Xuống hàng ở mỗi phần cuối của object cuối cùng
- file = sys.stdout : output được print trên screen
- flush The stream is not forcibly flushed.

```
113  a = 'home'
114  b = 'Khoa'
115  c = 'AI_2022'
116  d = 'Exercise1'
117  print("path = ", a, b, c, d, sep='/')
```

```
path = /home/Khoa/AI_2022/Exercise1
```

- sep = '/' : Giữa các object có một ký tự '/' '
- end = '\n' : Xuống hàng ở mỗi phần cuối của object cuối cùng

```
121  print("line1", end='-')
122  print("line2", end='-')
123  print("line3")
```

```
line1-line2-line3
```

- sep = ' ' : Giữa các object có một space
- end = '-' : Thêm ký tự '-' vào phần cuối cùng của object cuối cùng

```
127  sourceFile = open('python.txt', 'w')
128  print('Content in txt file', file = sourceFile)
129  sourceFile.close()
```

Python print() function

- **f-strings in python**

```
131  var_name = 'vname'
132  f"This is an f-string {var_name} and {var_name}."
```

```
This is an f-string vname and vname.
```

Đặt biến vào trong giữa {}. Tại runtime tất cả các biến này sẽ được thay bằng giá trị tương ứng

```
135  num1 = 4
136  num2 = 5
137  print(f"The product of {num1} and {num2} is {num1 * num2}.")
```

```
The product of 4 and 5 is 20.
```

Có thể là variable hoặc expression

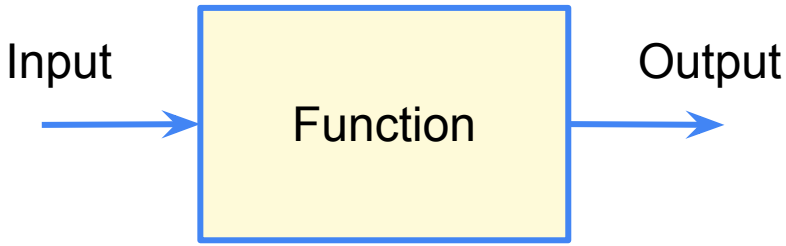
```
142  def mul(num1,num2):
143      return num1*num2
144  num1 = 4
145  num2 = 5
146  print(f"The product of {num1} and {num2} is {mul(num1,num2)}.")
```

```
The product of 4 and 5 is 20.
```

Có thể là variable hoặc call function

Exercise 1: Viết function đánh giá classification model bằng f1-score

Exercise 1: Viết function đánh giá classification model bằng f1-score



Hiểu yêu cầu đề bài

Xác Định Input và Output

Điều kiện ràng buộc

Pseudocode

Flowchart

Exercise 1: Viết function đánh giá classification model bằng f1-score

Viết function thực hiện đánh giá classification model bằng F1-Score.

- $Precision = \frac{TP}{TP+FP} = \frac{TP}{all\ detections}$
- $Recall = \frac{TP}{TP+FN} = \frac{TP}{all\ ground\ truths}$
- $F1\_score = \frac{2*Precision*Recall}{Precision+Recall}$

- Input: function nhận 3 giá trị **tp, fp, fn**
- Output: print ra kết quả của **Precision, Recall, và F1-score**

NOTE: Đề bài yêu cầu các điều kiện sau

- Phải **kiểm tra giá trị nhận vào tp, fp, fn là type int**, nếu là type khác thì print ví dụ check fn là float, print **'fn must be int'** và thoát hàm hoặc dừng chương trình.
- Yêu cầu **tp, fp, fn phải đều lớn hơn 0**, nếu không thì print **'tp and fp and fn must be greater than zero'** và thoát hàm hoặc dừng chương trình

```
# Examples
exercise1(tp=2, fp=3, fn=4)
>> precision is  0.4
recall is  0.3333333333333333
f1-score is  0.3636363636363636

exercise1(tp='a', fp=3, fn=4)
>> tp must be int


exercise1(tp=2, fp='a', fn=4)
>> fp must be int


exercise1(tp=2, fp=3, fn='a')
>> tp must be int

exercise1(tp=2, fp=3, fn=0)
>> tp and fp and fn must be greater than zero

exercise1(tp=2.1, fp=3, fn=0)
>> tp must be int
```

Exercise 1: Viết function đánh giá classification model bằng f1-score

- **Hiểu yêu cầu đề bài**

Viết 1 Functions

Tính F1-Score

Hàm nhận 3 parameters tp, tp, và fn

Print kết quả Precision, Recall , F1-score

Viết function thực hiện đánh giá classification model bằng F1-Score.

- $Precision = \frac{TP}{TP+FP} = \frac{TP}{all\ detections}$
- $Recall = \frac{TP}{TP+FN} = \frac{TP}{all\ ground\ truths}$
- $F1\_score = \frac{2*Precision*Recall}{Precision+Recall}$

- Input: function nhận 3 giá trị **tp, fp, fn**
- Output: print ra kết quả của **Precision, Recall, và F1-score**

Thông tin thêm:
- Các ràng buộc và xử lý lỗi
- tp, fp, fn thuộc type int
- tp, fp, fn > 0

NOTE: Đề bài yêu cầu các điều kiện sau

- Phải **kiểm tra giá trị nhận vào tp, fp, fn là type int**, nếu là type khác thì print ví dụ check fn là float, print **'fn must be int'** và thoát hàm hoặc dừng chương trình.
- Yêu cầu **tp, fp, fn phải đều lớn hơn 0**, nếu không thì print **'tp and fp and fn must be greater than zero'** và thoát hàm hoặc dừng chương trình

Exercise 1: Viết function đánh giá classification model bằng f1-score

- **Pseudocode**

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

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

$$\bullet\ F1-score\ = \frac{2*Precision*Recall}{Precision+Recall}$$

```
FUNCTION calc_f1_score(tp, fp, fn)
    precision = tp/(tp+fp)
    recall = tp/(tp+fn)
    f1_score = 2*(precision*recall)/(precision+recall)

    PRINT precision
    PRINT recall
    PRINT f1_score

ENDFUNCTION
```

Exercise 1: Viết function đánh giá classification model bằng f1-score

- **Pseudocode**

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

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

$$\bullet F1\_score = \frac{2*Precision*Recall}{Precision+Recall}$$

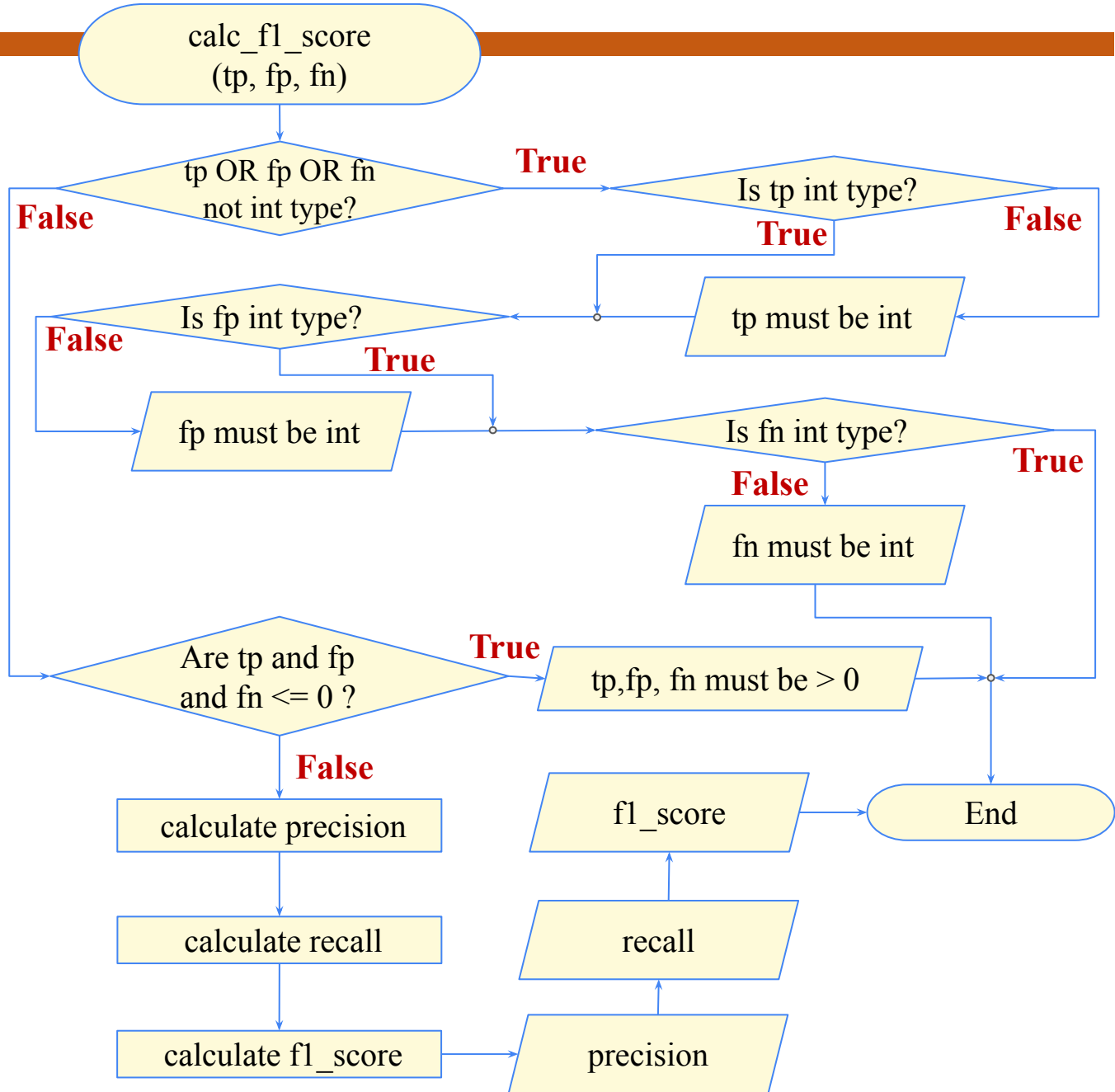
NOTE: Đề bài yêu cầu các điều kiện sau

- Phải **kiểm tra giá trị nhận vào tp, fp, fn là type int**, nếu là type khác thì print ví dụ check fn là float, print **'fn must be int'** và thoát hàm hoặc dừng chương trình.
- Yêu cầu **tp, fp, fn phải đều lớn hơn 0**, nếu không thì print **'tp and fp and fn must be greater than zero'** và thoát hàm hoặc dừng chương trình

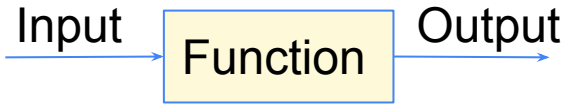
```
FUNCTION calc_f1_score(tp, fp, fn)
    IF type of tp is not int type OR type of fp is not int type OR
type of fn is not int type THEN
        IF type of tp is not int type THEN
            PRINT "tp must be int"
        IF type of fp is not int type THEN
            PRINT "fp must be int"
        IF type of fn is not int type THEN
            PRINT "fn must be int"
        RETURN
    IF (tp <= 0) and not (fp <= 0) and not (fn <= 0) THEN
        PRINT "tp and fp and fn must be greater than zero"
        RETURN
    ENDIF

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

    PRINT precision
    PRINT recall
    PRINT f1_score

ENDFUNCTION
```

Exercise 1: Viết function đánh giá classification model bằng f1-score

```
FUNCTION calc_f1_score(tp, fp, fn)
    IF type of tp is not int type OR type of fp is not int type OR
type of fn is not int type THEN
      IF type of tp is not int type THEN
          PRINT "tp must be int"
      IF type of fp is not int type THEN
          PRINT "fp must be int"
      IF type of fn is not int type THEN
          PRINT "fn must be int"
      RETURN
    IF (tp <= 0) and not (fp <= 0) and not (fn <= 0) THEN
        PRINT "tp and fp and fn must be greater than zero"
        RETURN
     ENDIF

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

    PRINT precision
    PRINT recall
    PRINT f1_score

ENDFUNCTION
```



Exercise 1: Viết function đánh giá classification model bằng f1-score



Hiểu yêu cầu đề bài

Xác Định Input và Output

Điều kiện ràng buộc

Pseudocode

Flowchart

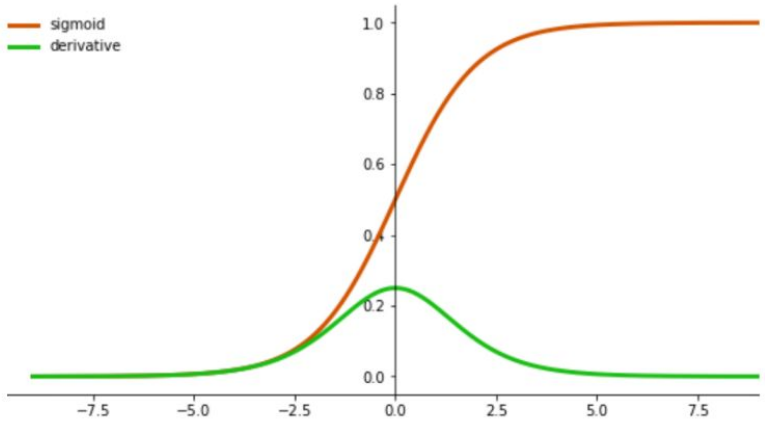
Exercise 2: Viết function tính kết quả theo các activation function

Exercise 2: Viết function tính kết quả theo các activation function

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

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

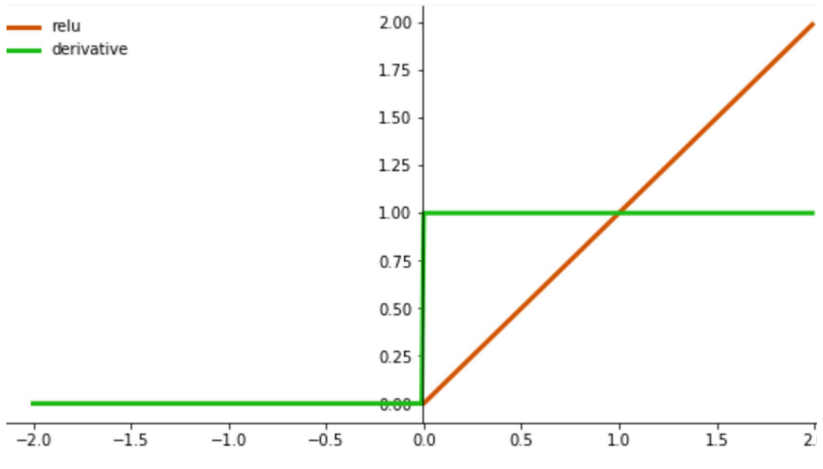
$$ELU(x) = \begin{cases} \alpha(e^{x} - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$



data = | 1 | 5 | -4 | 3 | -2 |

data_a = sigmoid(data)

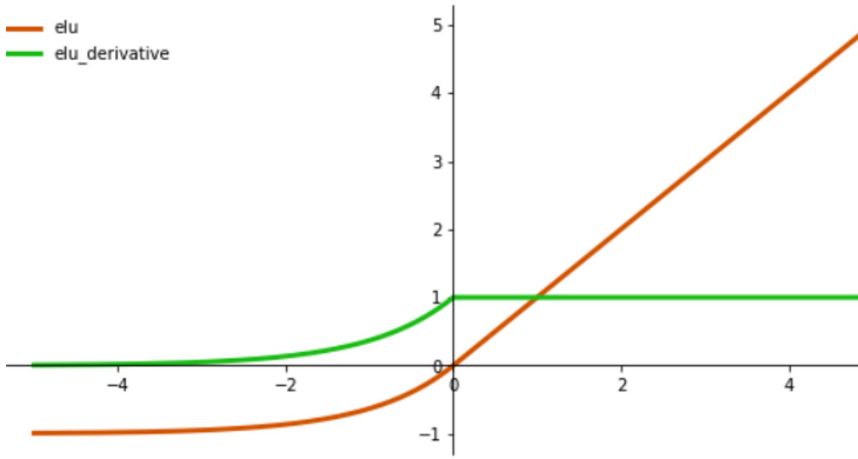
data_a = | 0.731 | 0.993 | 0.017 | 0.95 | 0.119 |



data = | 1 | 5 | -4 | 3 | -2 |

data_a = ReLU(data)

data_a = | 1 | 5 | 0 | 3 | 0 |



data = | 1 | 5 | -4 | 3 | -2 |

data_a = ELU(data)

data_a = | 1 | 5 | -0.098 | 3 | -0.086 |

Exercise 2: Viết function tính kết quả theo các activation function

• Input:
  – Người dùng nhập giá trị **x**
  – Người dùng nhập tên **activation function chỉ có 3 loại (sigmoid, relu, elu)**

• Output: Kết quả **f(x)** (x khi đi qua activation function tương ứng theo activation function name). Ví dụ **nhập x=3, activation_function = 'relu'. Output: print 'relu: f(3)=3'**

NOTE: Lưu ý các điều kiện sau:
• Dùng function **is_number** được cung cấp sẵn để **kiểm tra x có hợp lệ hay không** (vd: x='10', is_number(x) sẽ trả về True ngược lại là False). Nếu **không hợp lệ print 'x must be a number' và dừng chương trình.**

• Kiểm tra **activation function name có hợp lệ hay không nằm trong 3 tên (sigmoid, relu, elu). Nếu không print 'ten_function_user is not supported'** (vd người dùng nhập 'belu' thì print 'belu is not supported')

• Convert **x** sang **float** type

• Thực hiện theo công thức với activation name tương ứng. Print ra kết quả

• Dùng math.e để lấy số e

• $\alpha = 0.01$

```python
# Given
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
```

Code Listing 3: Cho trước hàm is_number

```
exercise3()
>> Input x = 1.5
Input activation Function (binary|sigmoid|elu): sigmoid
sigmoid: f(1.5) = 0.8175744761936437

exercise3()
>> Input x = abc
x must be a number

exercise3()
>> Input x = 1.5
Input activation Function (binary|sigmoid|elu): relu
relu is not supportted
```

Exercise 2: Viết function tính kết quả theo các activation function

- **Hiểu yêu cầu đề bài**

Viết 1 Functions

Tính 1 trong 3 activation function từ yêu cầu từ user

Hàm nhận 2 input từ user x và tên function

Print tên function và kết quả với x

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

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

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

• Input:
  – Người dùng nhập giá trị **x**
  – Người dùng nhập tên **activation function chỉ có 3 loại (sigmoid, relu, elu)**

• Output: Kết quả **f(x)** (x khi đi qua activation function tương ứng theo activation function name). Ví dụ **nhập x=3, activation_function = 'relu'. Output: print 'relu: f(3)=3'**

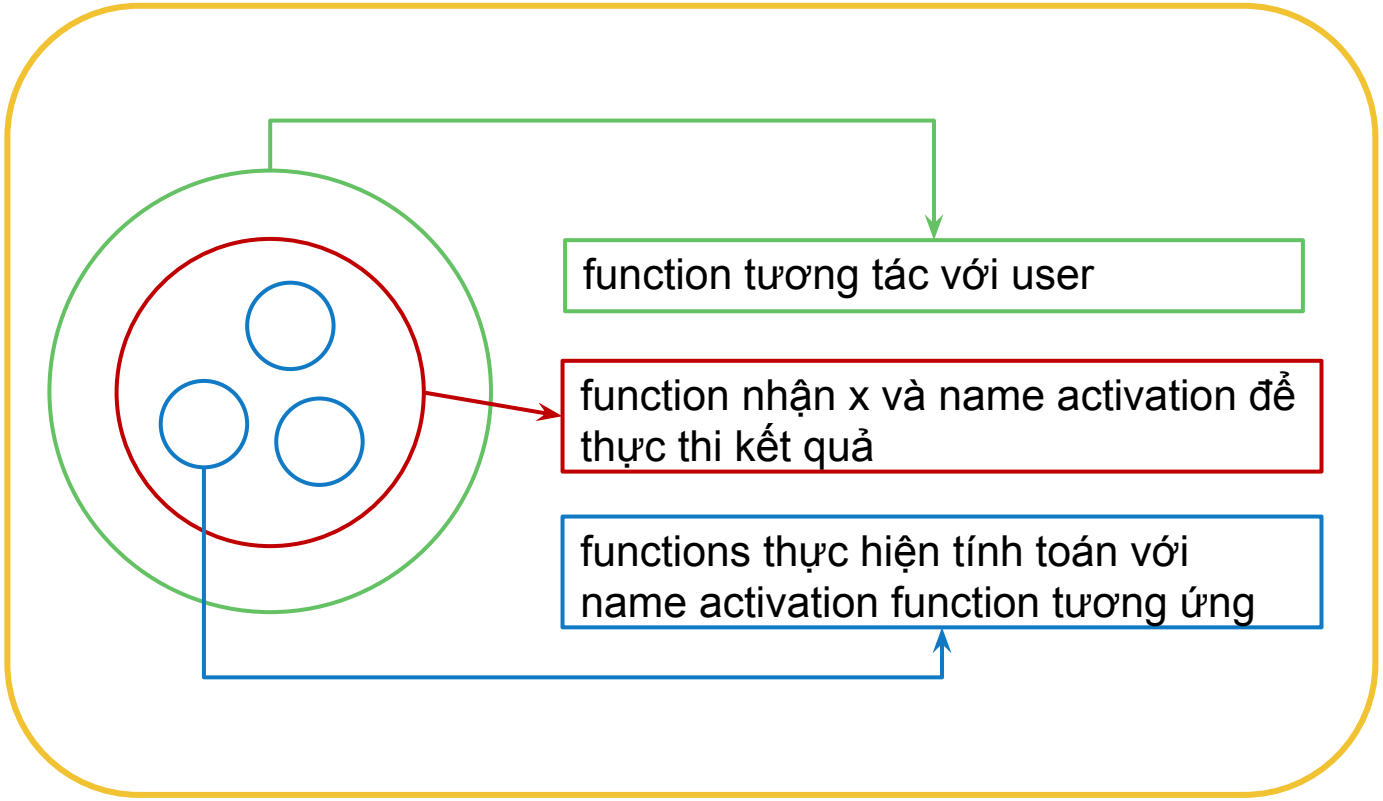
Thông tin thêm:
- Chỉ nhận x là con số
- Chỉ nhận 1 trong 3 tên activation function mà hàm hỗ trợ
- x khi tính toán nên ở dạng float
- Dùng math module để lấy số e
- Dùng $\alpha = 0.01$

• Convert **x** sang **float** type
• Dùng math.e để lấy số e
• α = 0.01

• Dùng function **is_number** được cung cấp sẵn để **kiểm tra x có hợp lệ hay không** (vd: x='10', is_number(x) sẽ trả về True ngược lại là False). Nếu **không hợp lệ print 'x must be a number' và dừng chương trình.**

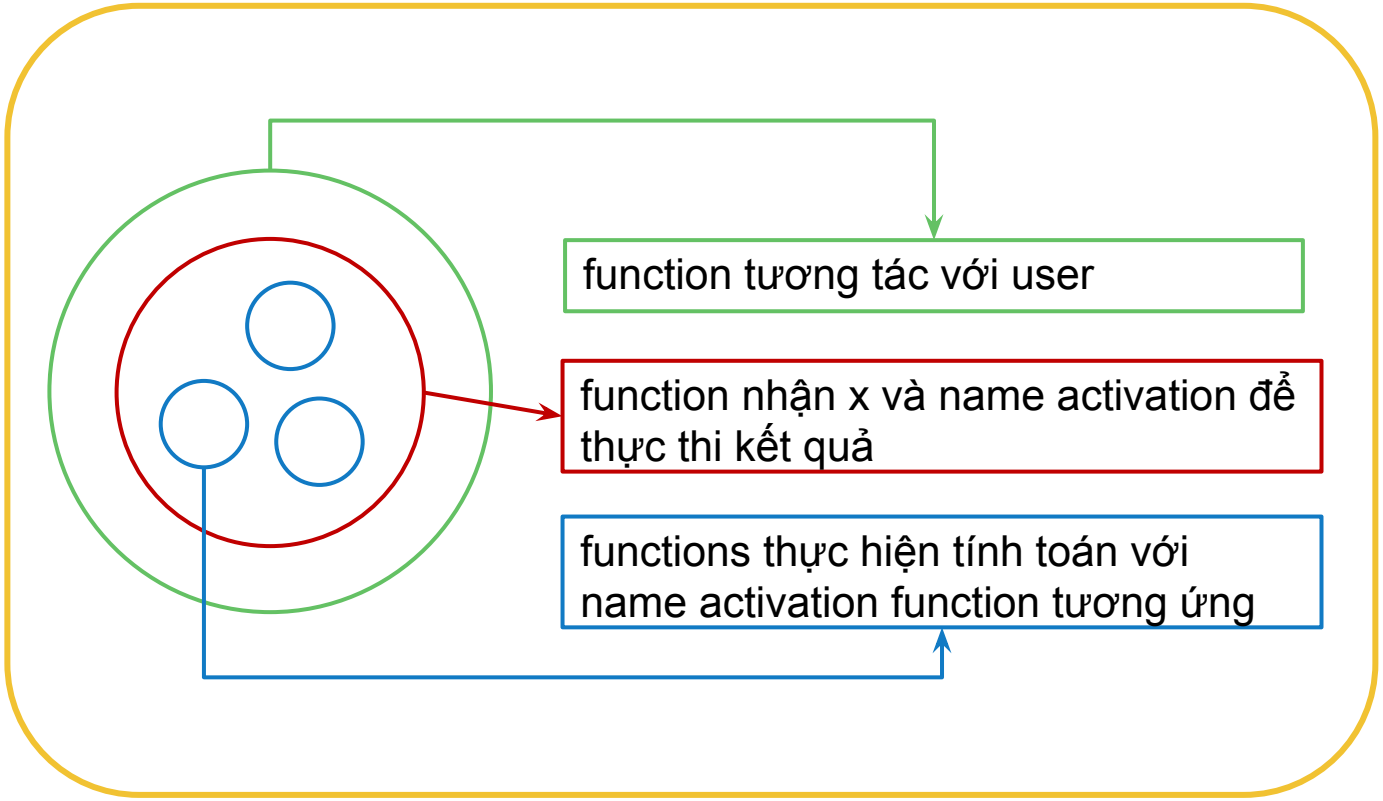
• Kiểm tra **activation function name có hợp lệ hay không nằm trong 3 tên (sigmoid, relu, elu). Nếu không print 'ten_function_user is not supported'** (vd người dùng nhập 'belu' thì print 'belu is not supported')

Exercise 2: Viết function tính kết quả theo các activation function



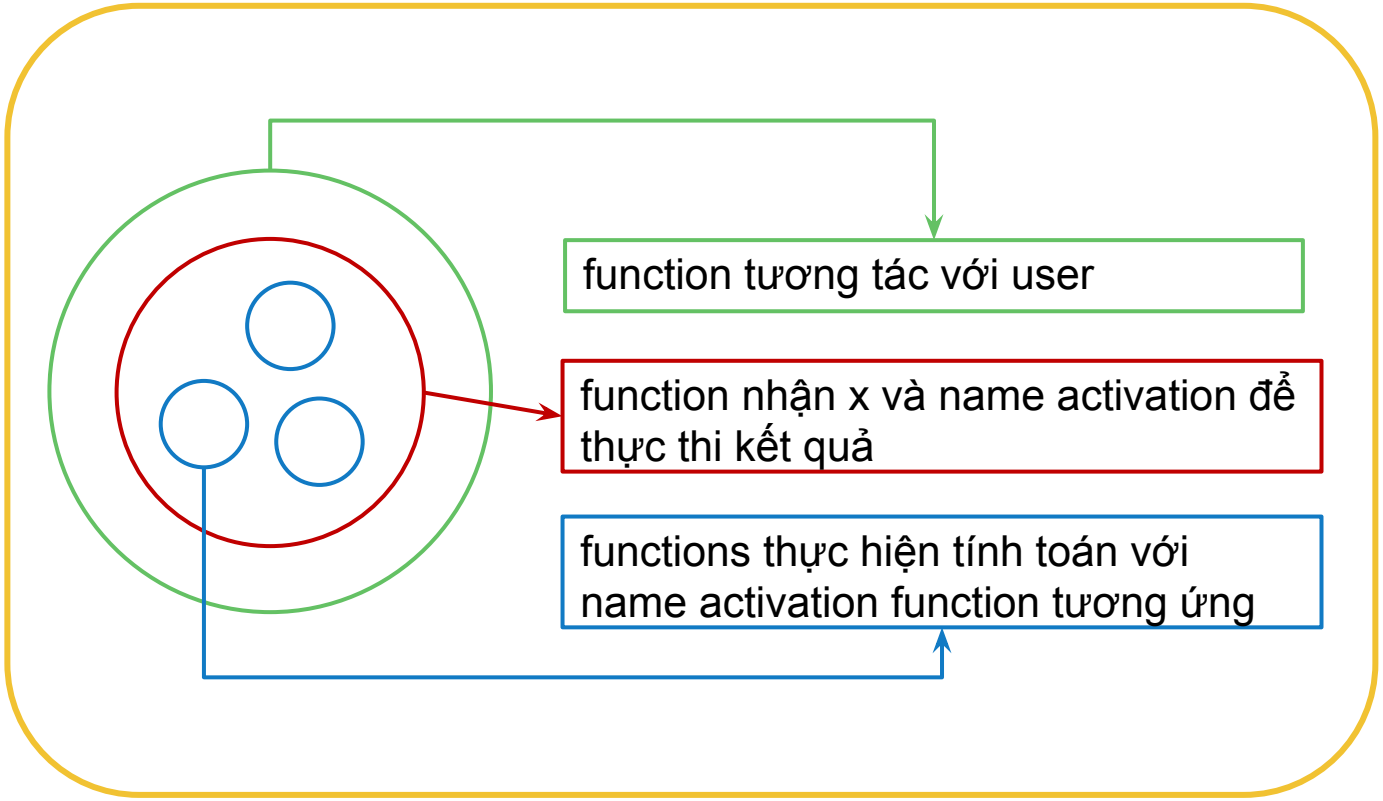

```
FUNCTION exercise2()
    x = input('Input x = ')
    IF not is_number(x) THEN
        PRINT "x must be a number"
        RETURN
    ENDIF

    act_name = input('activation function name: ')
    cast x to float
    result = calc_activation_func(x, act_name)

    IF act_name is invalid (result is None) THEN
        PRINT act_name + "is not supported"
    ELSE
        PRINT act_name + string(x)  + string(result)
    ENDIF

ENDFUNCTION
```

Exercise 2: Viết function tính kết quả theo các activation function




```
FUNCTION calc_activation_func(x, act_name)
    result = None
    IF act_name == sigmoid THEN
        result  = calc_sig(x)
    ELSE IF act_name == 'relu' THEN
        result  = calc_relu(x)
    ELSE IF act_name == 'elu' THEN
        result  = calc_elu(x)
    RETURN result
ENDFUNCTION
```

Exercise 2: Viết function tính kết quả theo các activation function




```
FUNCTION calc_sig(x)
    result = 1./(1+math.e**(-x))
    RETURN result
ENDFUNCTION
```

```
FUNCTION calc_relu(x)
    IF x < 0 THEN
        result = 0.
    ELSE
        result = x
    ENDIF
    RETURN result
ENDFUNCTION
```

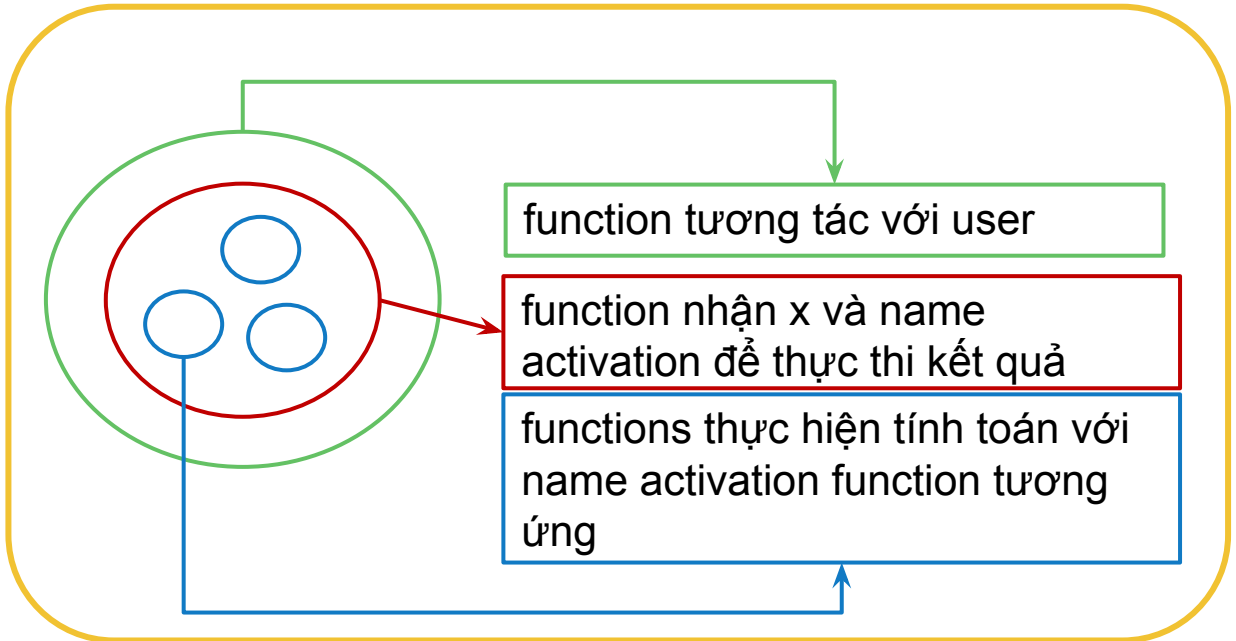
```
FUNCTION calc_elu(x)
    alpha = 0.01
    IF x < 0 THEN
        result = alpha*(math.e**x - 1)
    ELSE
        result = x
    ENDIF
    RETURN result
ENDFUNCTION
```

```
FUNCTION calc_sig(x)
    result = 1./(1+math.e**(-x))
    RETURN result
ENDFUNCTION
```

```
FUNCTION calc_elu(x)
    alpha = 0.01
    IF x < 0 THEN
        result = alpha*(math.e**x - 1)
    ELSE
        result = x
    ENDIF
    RETURN result
ENDFUNCTION
```

```
FUNCTION calc_activation_func(x, act_name)
    result = None
    IF act_name == sigmoid THEN
        result  = calc_sig(x)
    ELSE IF act_name == 'relu' THEN
        result  = calc_relu(x)
    ELSE IF act_name == 'elu' THEN
        result  = calc_elu(x)
    RETURN result
ENDFUNCTION
```

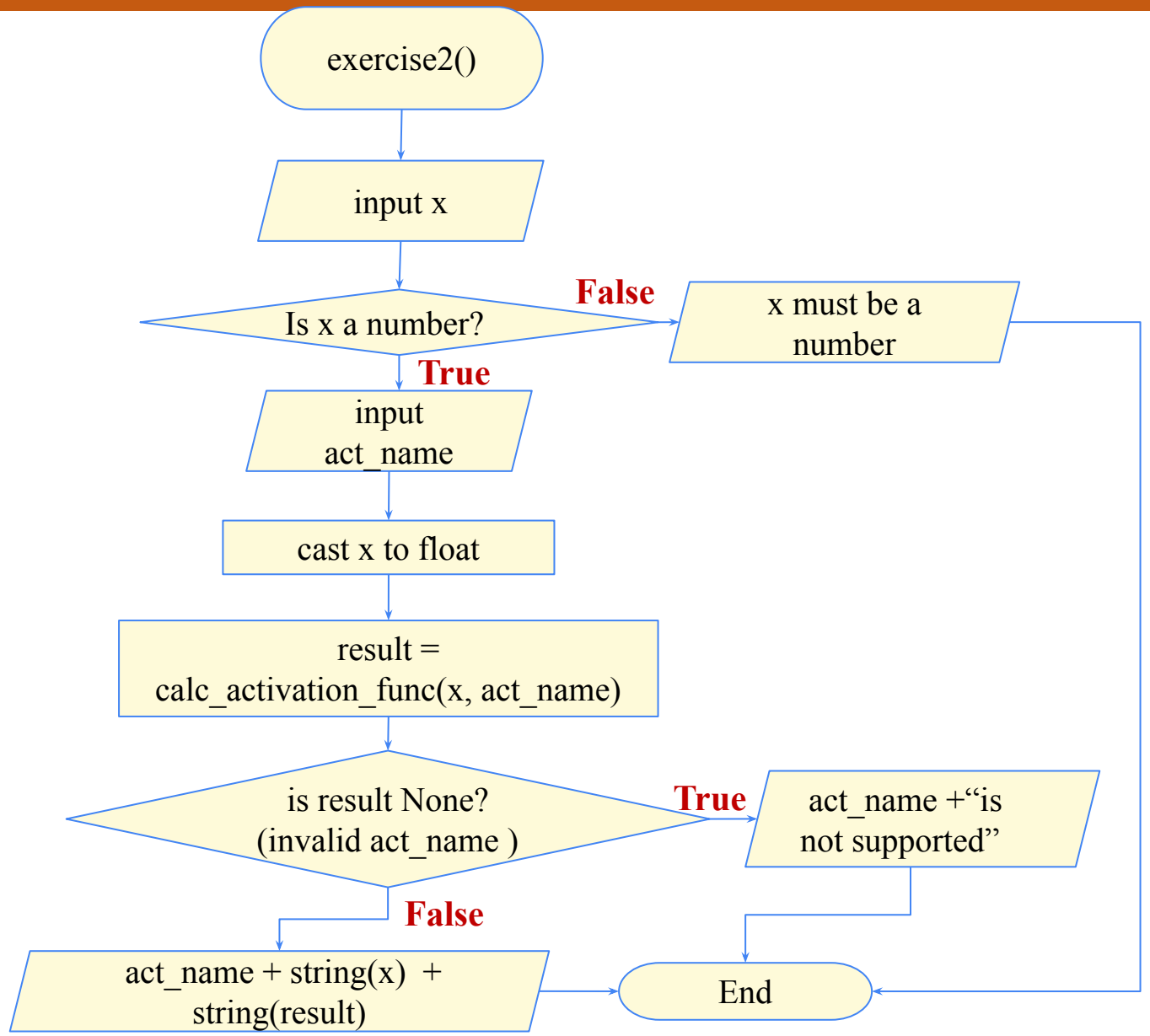
```
FUNCTION calc_relu(x)
    IF x < 0 THEN
        result = 0.
    ELSE
        result = x
    ENDIF
    RETURN result
ENDFUNCTION
```




```
FUNCTION exercise2()
    x = input('Input x = ')
    IF not is_number(x) THEN
        PRINT "x must be a number"
        RETURN
    ENDIF

    act_name = input('activation function name: ')
    cast x to float
    result = calc_activation_func(x, act_name)

    IF act_name is valid (result is None) THEN
        PRINT act_name + "is not supported"
    ELSE
        PRINT act_name + string(x)  + string(result)
    ENDIF

ENDFUNCTION
```

Exercise 2: Viết function tính kết quả theo các activation function

```
FUNCTION exercise2()
    x = input('Input x = ')
    IF not is_number(x) THEN
        PRINT "x must be a number"
        RETURN
    ENDIF

    act_name = input('activation function name: ')
    cast x to float
    result = calc_activation_func(x, act_name)

    IF act_name is valid (result is None) THEN
        PRINT act_name + "is not supported"
    ELSE
        PRINT act_name + string(x)  + string(result)
    ENDIF

ENDFUNCTION
```



Exercise 3: Viết function chọn regression loss function để tính loss

Exercise 3: Viết function tính loss theo regression loss function

4. **Viết function lựa chọn regression loss function để tính loss: :**

- 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 = $\sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$
- **n** chính là **số lượng samples (num_samples)**, với **i** là mỗi sample cụ thể. Ở đây các bạn có thể hiểu là cứ mỗi **i** thì sẽ **có 1 cặp** y_i là **target** và $\widehat{y_i}$ là predict.

- Input:
  - Người dùng **nhập số lượng sample (num_samples) được tạo ra (chỉ nhận integer numbers)**
  - Người dùng **nhập loss name (MAE, MSE, RMSE-(optional)) chỉ cần MAE và MSE, bạn nào muốn làm thêm RMSE đều được.**

- Output: Print ra **loss name, sample, predict, target, loss**

– **loss name:** là loss mà người dùng chọn

– **sample:** là thứ tự sample được tạo ra (ví dụ num_samples=5, thì sẽ có 5 samples và mỗi sample là sample-0, sample-1, sample-2, sample-3, sample-4)

– **predict:** là số mà model dự đoán (chỉ cần dùng random tạo random một số trong range [0,10))

– **target:** là số target mà mong muốn mode dự đoán đúng (chỉ cần dùng random tạo random một số trong range [0,10))

– **loss:** là kết quả khi đưa predict và target vào hàm loss

– **note:** ví dụ num_sample=5 thì sẽ có 5 cặp predict và target

Exercise 3: Viết function tính loss theo regression loss function

- Dùng **.isnumeric() method** để kiểm tra **num_samples** có hợp lệ hay không (vd: x='10', num_samples.isnumeric() sẽ trả về True ngược lại là False). **Không hợp lệ print 'number of samples must be an integer number'** và dừng chương trình.
- **Dùng vòng lặp for, lặp lại num_samples lần. Mỗi lần dùng random modules tạo một con số ngẫu nhiên trong range [0.0, 10.0) cho predict và target.** Sau đó đưa predict và target vào loss function và print ra kết quả mỗi lần lặp.
- **Dùng random.uniform(0,10)** để tạo ra một số ngẫu nhiên trong range [0,10)
- **Giả xử người dùng luôn nhập đúng loss name MSE, MAE, và RMSE (đơn giảng bước này để các bạn không cần check tên hợp lệ)**
- Dùng abs() để tính trị tuyệt đối ví dụ abs(-3) sẽ trả về 3
- Dùng math.sqrt() để tính căn bậc 2

```
exercise4()
>> Input number of samples (integer number) which are generated: 5
Input loss name: RMSE
loss name: RMSE, sample: 0, pred: 6.659262156575629, target: 4.5905830130732355,
    loss: 4.279433398761796
loss name: RMSE, sample: 1, pred: 4.592264312227207, target: 8.447168720237958,
    loss: 14.860287994900718
loss name: RMSE, sample: 2, pred: 8.701801828625959, target: 9.280646891626386,
    loss: 0.3350616069599687
loss name: RMSE, sample: 3, pred: 4.799972972282257, target: 9.877147335937869,
    loss: 25.777699518961764
loss name: RMSE, sample: 4, pred: 0.20159822778697878, target: 5.540221923628147,
     loss: 28.50090296579681
final RMSE: 3.8406610234536727


exercise4()
>> Input number of samples (integer number) which are generated: aa
number of samples must be an integer number
```

Exercise 3: Viết function tính loss theo regression loss function

- **Hiểu yêu cầu đề bài**

Viết 1 Functions

Tính 1 trong 3 loss function từ yêu cầu từ user

Hàm nhận 2 input từ user num_samples và tên function

Print loss name, sample, predict, target, loss, final loss

Thông tin thêm:
- **Dùng .isnumeric() kiểm tra num_samples**
- **predict và target ($\hat{y}$, y) được tạo ra random trong mỗi lần lặp**
- **Random trong range [0,10)**

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

Công thứ tính theo series cần vòng lặp n lần

- RMSE $= \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$
- **n** chính là **số lượng samples (num_samples)**, với **i** là mỗi sample cụ thể. Ở đây các bạn có thể hiểu là cứ mỗi **i** thì sẽ **có 1 cặp** y_i là **target** và $\widehat{y_i}$ là predict.

- Input:
  - Người dùng **nhập số lượng sample (num_samples) được tạo ra (chỉ nhận integer numbers)**
  - Người dùng **nhập loss name (MAE, MSE, RMSE-(optional)) chỉ cần MAE và MSE, bạn nào muốn làm thêm RMSE đều được.**

- Output: Print ra **loss name, sample, predict, target, loss và final loss**

- Dùng **.isnumeric() method** để kiểm tra **num_samples** có hợp lệ hay không (vd: x='10', num_samples.isnumeric() sẽ trả về True ngược lại là False). **Không hợp lệ print 'number of samples must be an integer number'** và dừng chương trình.
- **Dùng vòng lặp for, lặp lại num_samples lần. Mỗi lần dùng random modules tạo một con số ngẫu nhiên trong range [0.0, 10.0) cho predict và target.** Sau đó đưa predict và target vào loss function và print ra kết quả mỗi lần lặp.
- **Dùng random.uniform(0,10)** để tạo ra một số ngẫu nhiên trong range [0,10)
- **Giả xử người dùng luôn nhập đúng loss name MSE, MAE, và RMSE (đơn giảng bước này để các bạn không cần check tên hợp lệ)**

Exercise 3: Viết function tính loss theo regression loss function

- $\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$
- $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- $\text{RMSE} = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$
- **n** chính là **số lượng samples (num_samples)**, với **i** là mỗi sample cụ thể. Ở đây các bạn có thể hiểu là cứ mỗi **i** thì sẽ **có 1 cặp** y_i **là target và** $\hat{y}$ **là predict**.

$$\sqrt{a} + \sqrt{b} \neq \sqrt{a+b}$$

```
exercise4()
>> Input number of samples (integer number) which are generated: 5
Input loss name: RMSE
loss name: RMSE, sample: 0, pred: 6.659262156575629, target: 4.5905830130732355,
    loss: 4.279433398761796
loss name: RMSE, sample: 1, pred: 4.592264312227207, target: 8.447168720237958,
    loss: 14.860287994900718
loss name: RMSE, sample: 2, pred: 8.701801828625959, target: 9.280646891626386,
    loss: 0.3350616069599687
loss name: RMSE, sample: 3, pred: 4.799972972282257, target: 9.877147335937869,
    loss: 25.777699518961764
loss name: RMSE, sample: 4, pred: 0.20159822778697878, target: 5.540221923628147,
    loss: 28.50090296579681
final RMSE: 3.8406610234536727


exercise4()
>> Input number of samples (integer number) which are generated: aa
number of samples must be an integer number
```

loss của Squared Error

Final loss của RMSE

Exercise 3: Viết function tính loss theo regression loss function

❖ Quiz: matching the functions and corresponding codes

```python
import math

def function1(y, y_hat):
    return abs(y-y_hat)

def function2(y, y_hat):
    return (y-y_hat)**2

def function3(y, y_hat):
    return math.sqrt( (y-y_hat)**2 )
```

a $(y - \hat{y})^2$

b $|y - \hat{y}|$

c $\sqrt{(y - \hat{y})^2}$

```python
import random

y = random.random()
y_hat = random.random()
```

(?, ?)

What are the range of the two variables?

```python
import random

y = random.uniform(0,10)
y_hat = random.uniform(0,10)
```

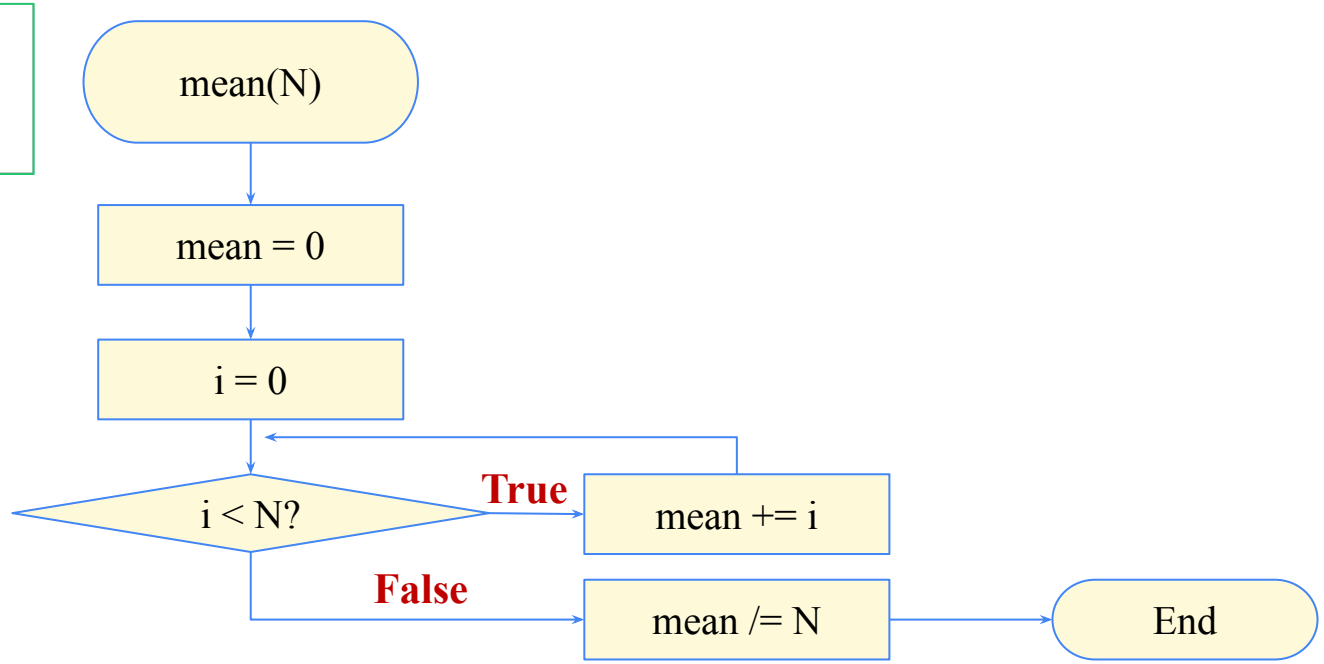
(?, ?)

Exercise 3: Viết function tính loss theo regression loss function

- 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 $= \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$
- **n** chính là **số lượng samples (num_samples)**, với **i** là mỗi sample cụ thể. Ở đây các bạn có thể hiểu là cứ mỗi **i** thì sẽ **có 1 cặp** y_i **là target và** $\hat{y}$ **là predict.**

$$mean = \frac{1}{N}\sum_{i=0}^{N-1} i\,,\ N > 0$$

```
FUNCTION mean(N)
    mean = 0
    FOR i start at 0 TO N - 1
        mean += i
    ENDFOR
    mean  /= N
    RETURN mean
ENDFUNCTION
```



Exercise 3: Viết function tính loss theo regression loss function

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

```
total_error = 0
FOR each iteration FROM 0 TO 1
    y = random number
    y_hat = random number
    absolute_error = absolute(y - y_hat)
    total_error += absolute_error
ENDFOR
mean_absolute_error = total_error / iterations
PRINT mean_absolute_error
```

Function()

```
FUNCTION mean_absolute_error(iterations)
    total_error = 0
    FOR each iteration FROM 0 TO iterations - 1
        y = random number
        y_hat = random number
        absolute_error = absolute(y - y_hat)
        total_error += absolute_error
    ENDFOR
    mean_absolute_error = total_error / iterations
    RETURN mean_absolute_error
ENDFUNCTION
```

```python
import random
import math

total_error = 0
for _ in range(2):
    y = random.random()
    y_hat = random.random()
    absolute_error = abs(y-y_hat)
    total_error = total_error + absolute_error

mean_absolute_error = total_error/2
print(mean_absolute_error)
```

❖ Quiz: matching again!

1

```python
import random
import math

total_error = 0
for _ in range(2):
    y = random.random()
    y_hat = random.random()
    absolute_error = abs(y-y_hat)
    total_error = total_error + absolute_error

mean_absolute_error = total_error/2
print(mean_absolute_error)
```

Given $n = 2$

a

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

b

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

c

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

2

```python
import random
import math

total_error = 0
for _ in range(2):
    y = random.random()
    y_hat = random.random()
    absolute_error = (y-y_hat)**2
    total_error = total_error + absolute_error

mean_squared_error = total_error/2
print(mean_squared_error)
```

3

```python
import random
import math

total_error = 0
for _ in range(2):
    y = random.random()
    y_hat = random.random()
    absolute_error = (y-y_hat)**2
    total_error = total_error + absolute_error

root_mean_squared_error = math.sqrt(total_error/2)
print(root_mean_squared_error)
```

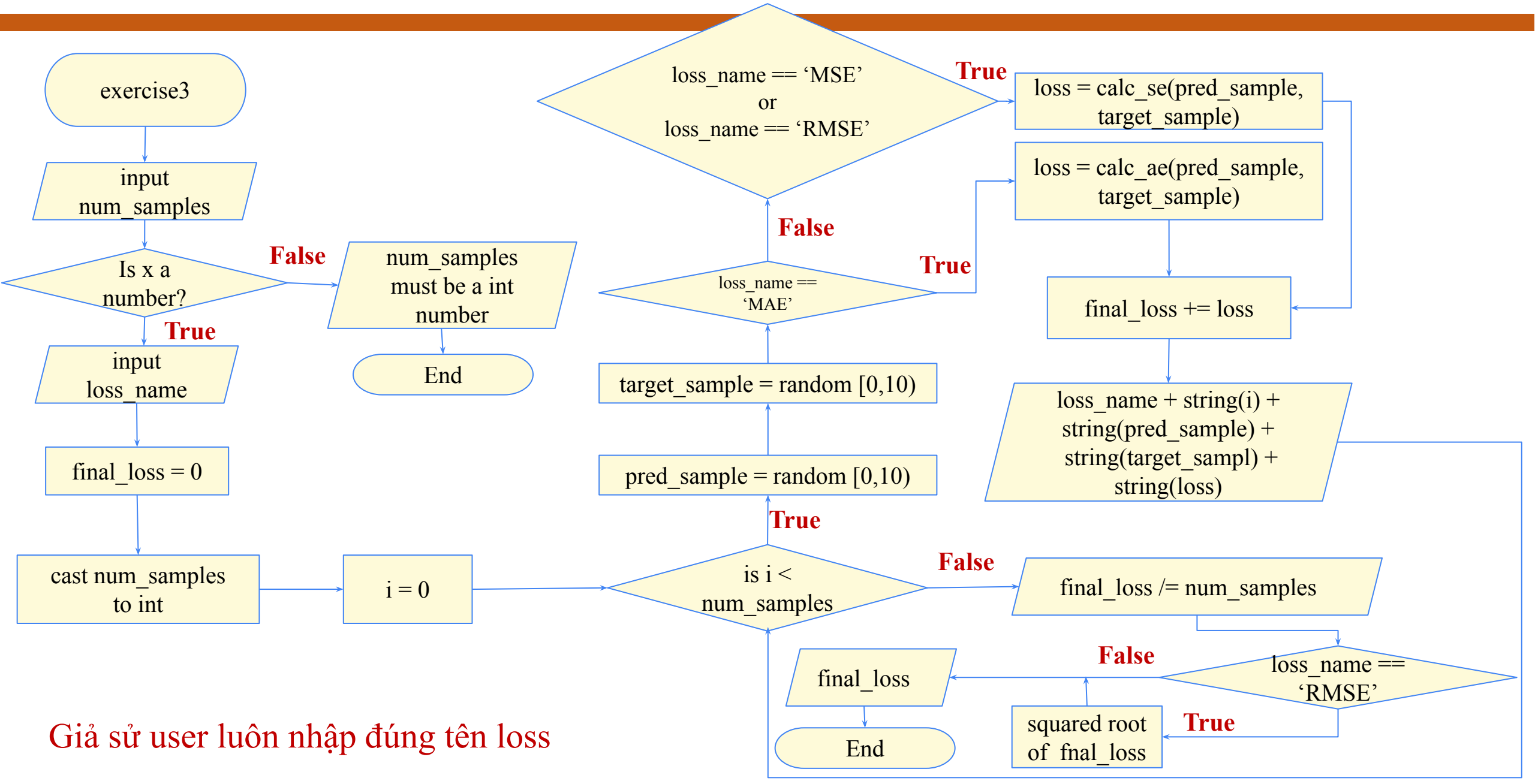
Pseudocode

```
FUNCTION calc_ae(y, y_hat)
    result = absolute(y - y_hat)
    RETURN result
ENDFUNCTION
```

```
FUNCTION calc_se(y, y_hat)
    result = (y - y_hat)**2
    RETURN result
ENDFUNCTION
```

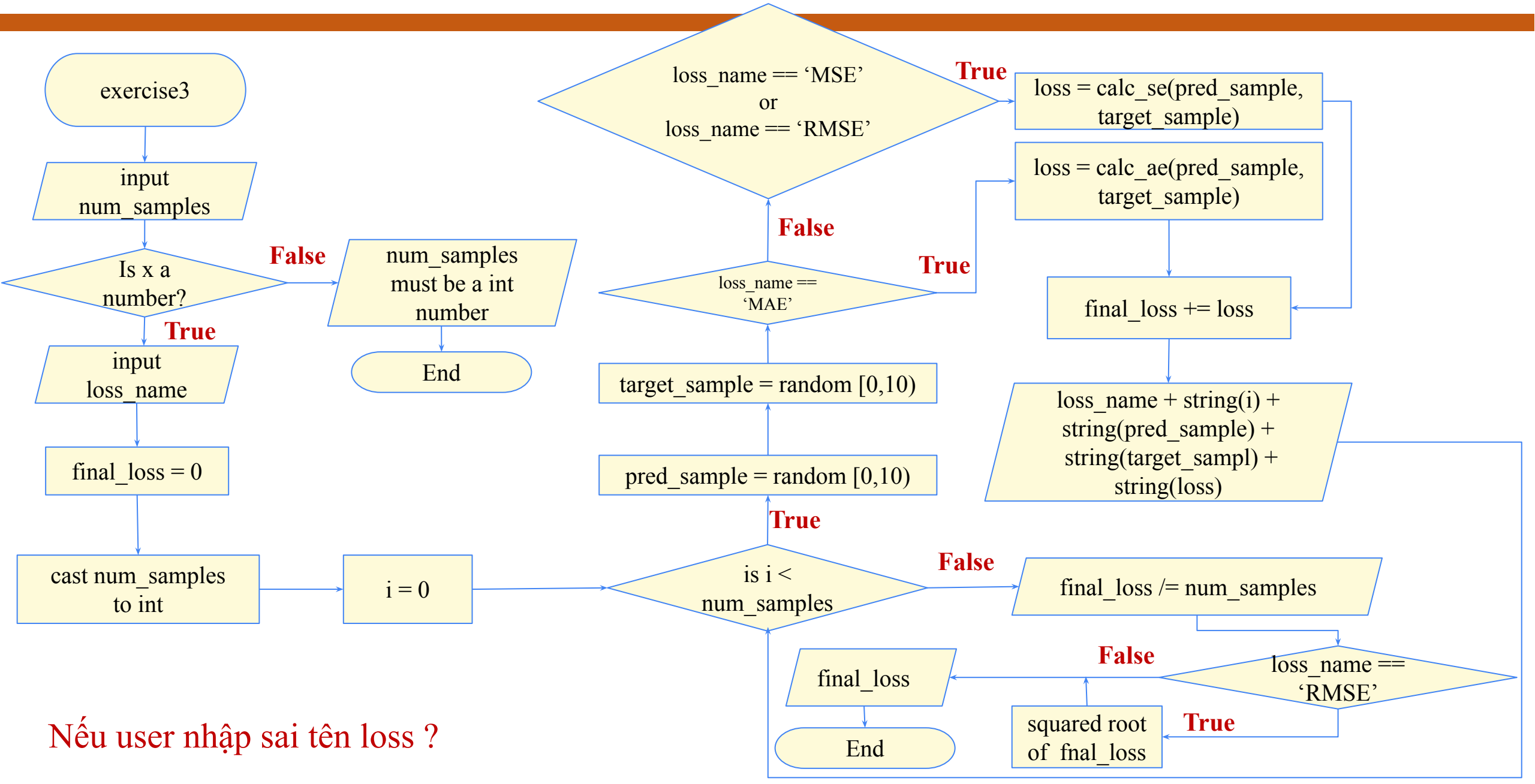
```
FUNCTION exercise3()
    num_samples = input('Input num_samples = ')
    IF not samples .isnumeric() THEN
        PRINT "number of samples must be an integer number"
        RETURN
    ENDIF
    loss_name = input('Input loss name:  ')

    final_loss = 0
    cast num_samples to int
    FOR i start at 0 TO num_samples - 1
        pred_sample = random FROM 0 TO less than 10
        target_sample = random FROM 0 TO less than 10

        IF loss_name == 'MAE' THEN
            loss = calc_ae(sample , target_sample)
        ELSE IF loss_name == 'MSE' OR loss_name == 'RMSE' THEN
            loss = calc_se(sample , target_sample)
        ENDIF
        final_loss += loss
        PRINT loss_name: {loss_name}, sample: {i}: pred: {pred_sample} target: {target_sample} loss: {loss}
    ENDFOR
    final_loss /= num_samples
    IF loss_name == 'RMSE' THEN
            final_loss = square root of final_loss
    ENDIF
    PRINT string(final_loss )
ENDFUNCTION
```

Exercise 3: Viết function tính loss theo regression loss function



Giả sử user luôn nhập đúng tên loss

Exercise 3: Viết function tính loss theo regression loss function



Nếu user nhập sai tên loss ?


QUIZ
QUIZ
QUIZ
QUIZ

Exercise 4: Viết các function để ước lượng các hàm số

Exercise4: Viết các function để ước lượng các hàm số

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

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

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

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

- Sử dụng loop
- Viết 1 hàm tính giai thừa
- Viết 1 hàm thực hiện tính toán và gọi hàm tính giai thừa

Exercise 4: Viết các function để ước lượng các hàm số

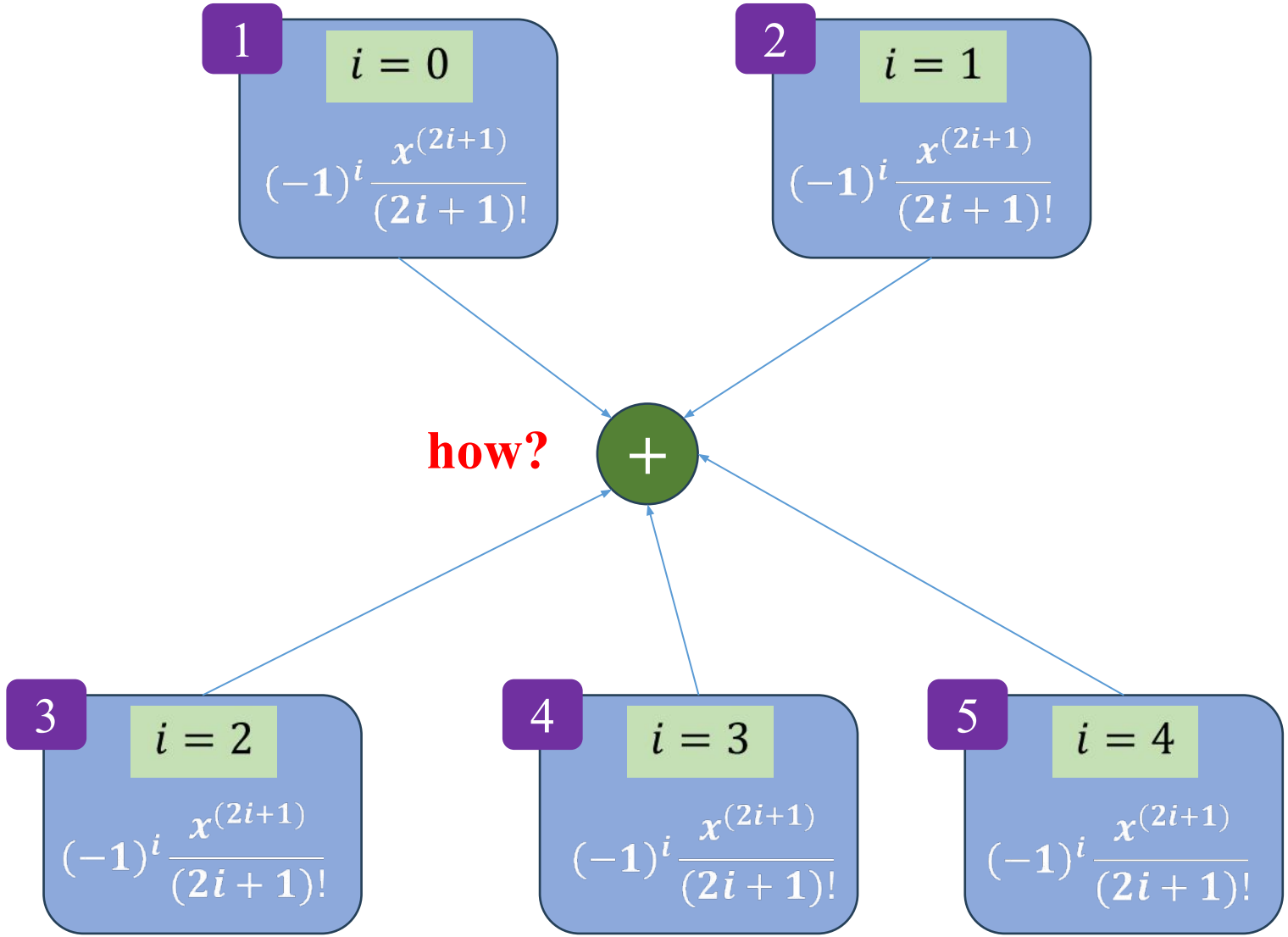
❖ sin(x)

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

practically

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

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



Exercise 4: Viết các function để ước lượng các hàm số

❖ sin(x)

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

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

```
def factorial(value):
    result = 1
    for i in range(1, value+1):
        result = result*i
    return result
```

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

```
def factorial(value):
    result = 1
    for i in range(value):
        result = result*(i+1)
    return result
```

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

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

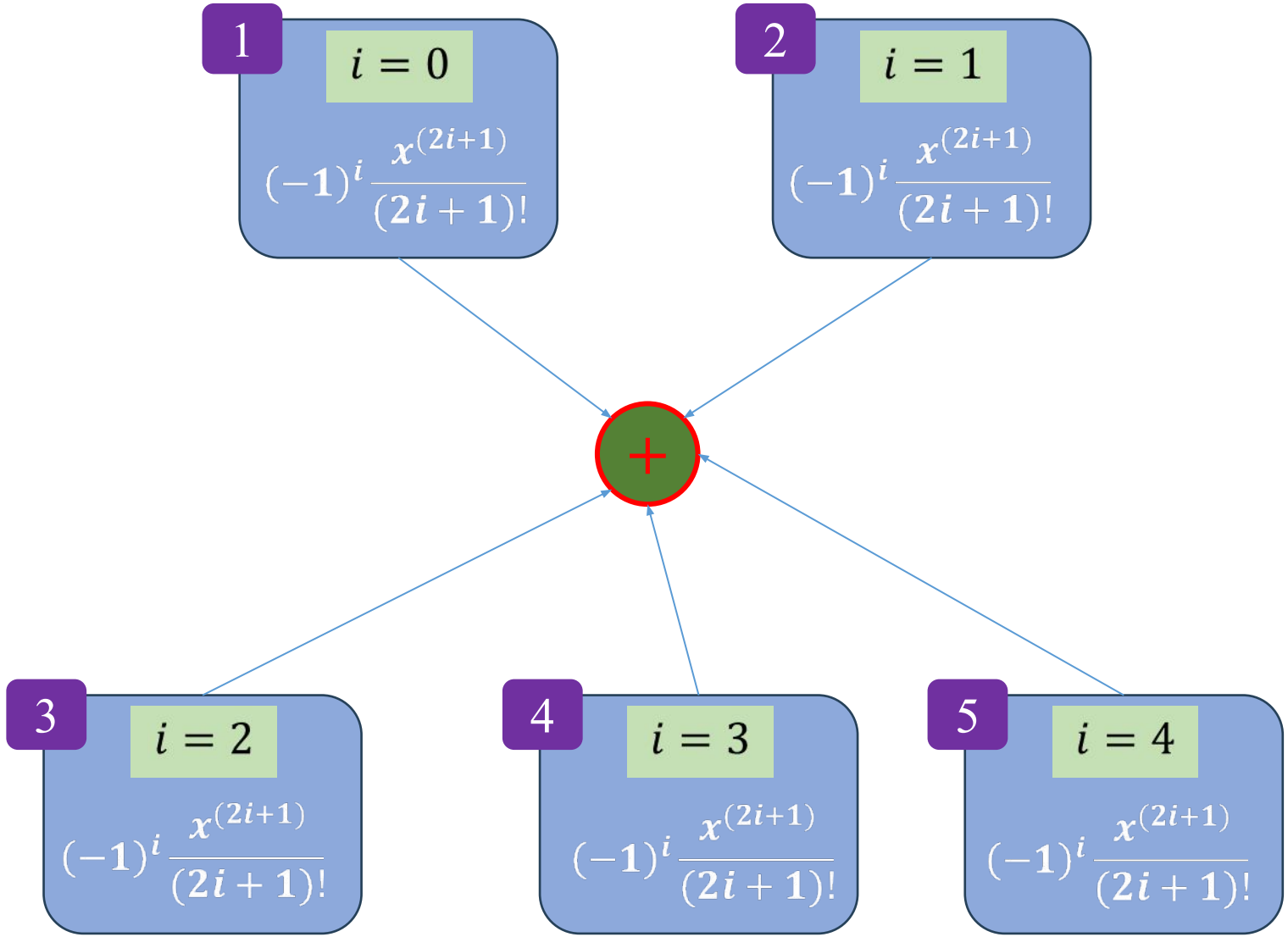
```
for i in range(n+1):
    coef = (-1)**i
    num = x**(2*i+1)
    denom = factorial_fcn(2*i+1)
    coef   * (num/denom)
```

Exercise 4: Viết các function để ước lượng các hàm số

❖ sin(x)

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

```
def factorial_fcn(value):
    res = 1
    for i in range(value):
        res *= (i+1)
    return res

def approx_sin(x, n):
    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
```



Exercise 4: Viết các function để ước lượng các hàm số

Summary

coef

num

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

denom

- Sử dụng loop
- Viết 1 hàm tính giai thừa
- Viết 1 hàm thực thiện tính toán và gọi hàm tính giai thừa

```
FUNCTION approx_sin(x, n)
    sin_approx = 0
    FOR i start at 0 TO n
        coef = (-1)**i
        num = x**(2*i+1)
        denom = factorial_fcn(2*i+1)
        sin_approx += coef * ((num)/(denom))
    ENDFOR
    RETURN sin_approx
ENDFUNCTION
```

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

```python
def factorial_fcn(value):
    res = 1
    for i in range(value):
        res *= (i+1)
    return res

def approx_sin(x, n):
    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
```

Exercise 4: Viết các function để ước lượng các hàm số

❖ cos(x)

coef

num

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

denom

- Sử dụng loop
- Viết 1 hàm tính giai thừa
- Viết 1 hàm thực thiện tính toán và gọi hàm tính giai thừa

```
FUNCTION approx_cos(x, n)
    cos_approx = 0
    FOR i FROM 0 TO n
        coef = (-1)**i
        num = x**(2*i)
        denom = factorial_fcn(2*i)
        cos_approx += coef * (num / denom)
    ENDFOR
    RETURN cos_approx
ENDFUNCTION
```

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

```python
def factorial_fcn(x):
    res = 1
    for i in range(x):
        res *= (i+1)
    return res

def approx_cos(x, n):
    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
```

Exercise 4: Viết các function để ước lượng các hàm số

❖ sinh(x)

num

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

denom

- Sử dụng loop
- Viết 1 hàm tính giai thừa
- Viết 1 hàm thực thiện tính toán và gọi hàm tính giai thừa

```
FUNCTION approx_sinh(x, n)
    sinh_approx = 0
    FOR i FROM 0 TO n
        coef = 1
        num = x**(2*i+1)
        denom = factorial_fcn(2*i+1)
        sinh_approx += coef * (num / denom)
    ENDFOR
    RETURN sinh_approx
ENDFUNCTION
```

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

```python
def factorial_fcn(value):
    res = 1
    for i in range(value):
        res *= (i+1)
    return res

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

Exercise 4: Viết các function để ước lượng các hàm số

❖ cosh(x)

num

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

denom

- Sử dụng loop
- Viết 1 hàm tính giai thừa
- Viết 1 hàm thực thiện tính toán và gọi hàm tính giai thừa

```
FUNCTION approx_cosh(x, n)
    cosh_approx = 0
    FOR i FROM 0 TO n
        coef = 1
        num = x**(2*i)
        denom = factorial_fcn(2*i)
        cosh_approx += coef * (num / denom)
    ENDFOR
    RETURN cosh_approx
ENDFUNCTION
```

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

```python
def factorial_fcn(value):
    res = 1
    for i in range(value):
        res *= (i+1)
    return res

def approx_cosh(x, n):
    cosh_approx = 0
    for i in range(n+1):
        coef = 1
        num = x**(2*i)
        denom = factorial_fcn(2*i)
        cosh_approx += coef * (num/denom)
    return cosh_approx
```

Exercise 5: Viết function thực hiện hàm MD_nRE

Exercise 5: Viết function thực hiện hàm MD_nRE

From YOLO loss function

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

$$\text{MD\_nRE} = \frac{1}{m} \sum_{i=1}^{m} \left(\sqrt[n]{y_i} - \sqrt[n]{\hat{y}_i} \right)^p$$

"Sum-squared error also equally weights errors in large boxes and small boxes. Our error metric should reflect that small deviations in large boxes matter less than in small boxes. To partially address this we predict the square root of the bounding box width and height instead of the width and height directly." - **You Only Look Once: Unified, Real-Time Object Detection (YOLOv1)**

Exercise 5: Viết function thực hiện hàm MD_nRE

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

$$\text{MD\_nRE} = \frac{1}{m}\sum_{i=1}^{m}\left(\sqrt[n]{y_i} - \sqrt[n]{\hat{y}_i}\right)^p$$

y	$\hat{y}$	MAE	MD_nRE $(n = 2, p = 1)$
100	99.5	0.5	0.025
50	49.5	0.5	0.035
20	19.5	0.5	0.056
5.5	5.0	0.5	0.110
1.0	0.5	0.5	0.293
0.6	0.1	0.5	0.458

Exercise 5: Viết function thực hiện hàm MD_nRE

$$\text{MD\_nRE} = \frac{1}{m}\sum_{i=1}^{m}\left(\sqrt[n]{y_i} - \sqrt[n]{\hat{y}_i}\right)^p$$

$$\downarrow$$

$$\left(\sqrt[n]{y} - \sqrt[n]{\hat{y}}\right)^p$$

```
md_nre_single_sample(y=100, y_hat=99.5, n=2, p=1)
>> 0.025031328369998107

md_nre_single_sample(y=50, y_hat=49.5, n=2, p=1)
>> 0.035444173213033135

md_nre_single_sample(y=20, y_hat=19.5, n=2, p=1))
>> 0.056255521835656574

md_nre_single_sample(y=0.6, y_hat=0.1, n=2, p=1)
>> 0.458368903224645 46
```

FUNCTION md_nre_single_sample(y, y_hat, n, p)
    y_root = y ** (1/n)
    y_hat_root = y_hat ** (1/n)
    difference = y_root - y_hat_root
    loss = difference ** p
    RETURN loss
ENDFUNCTION

Summary

- ✔ Overview of the print() function in Python
- ✔ Using for loop in Python
- ✔ Basic programming steps in Python
- ✔ Using of if, else, and string in Python
- ✔ Development of functions for evaluating classification and regression tasks
- ✔ Creation of functions for approximating trigonometric functions