

# AI VIET NAM – COURSE 2022

## Python Basic – Exercise

Ngày 3 tháng 5 năm 2022

Note: Các bạn có thể dùng hàm print để in các variable như sau

```
1 # Examples
2 var1 = 'str_var1'
3 var2 = 3
4 var3 = 15.5
5 print('Print single variable (var1): ', var1)
6 print(f'Print more than one variables: var1={var1}, var2={var2}, var3={var3}')
7
8 >> Print single variable (var1): str_var1
9 Print more than one variables: var1=str_var1, var2=3, var3=15.5
```

### 1. Viết 4 functions thực hiện tính thể tích của các khối sau:

- Hình lập phương  $V = s^3$  (s là độ dài của cạnh)
  - input function: s
  - output function: V
- Hình hộp chữ nhật  $V = l * w * h$  (l là chiều dài, w là chiều rộng, h: là chiều cao)
  - input function: l, w, h
  - output function: V
- Hình trụ tròn:  $V = \pi * r^2 * h$  (r là bán kính đáy, h là chiều cao)
  - input function: r, h
  - output function: V
- Hình cầu:  $V = \frac{4}{3} * \pi * r^3$  (r bán kính hình cầu)
  - input function: r
  - output function: V

NOTE: Tùy thuộc vào giá trị  $\pi$  các bạn định nghĩa mà kết quả có thể sai lệch. Ví dụ tự định nghĩa  $\pi = 3.14$  hoặc dùng math.pi  $\pi = 3.141592653589793$

```
1 # Examples
2 calc_cube_volume(s=3)
3 >> 27
4
5 calc_rectangular_prism_volume(l=1, w=2, h=3)
6 >> 6
7
8 calc_cylinder_volume(r=1.2, h=4)
9 >> 18.0864
```

```

10
11 calc_shpere_volume(r=1.3)
12 >> 9.198106666666666

```

Code Listing 1: Đây là các ví dụ các bạn không cần thiết đặt tên giống ví dụ

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

- Precision =  $\frac{TP}{TP + FP}$
- Recall =  $\frac{TP}{TP + FN}$
- F1-score =  $2 * \frac{Precision * Recall}{Precision + Recall}$
- 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

```

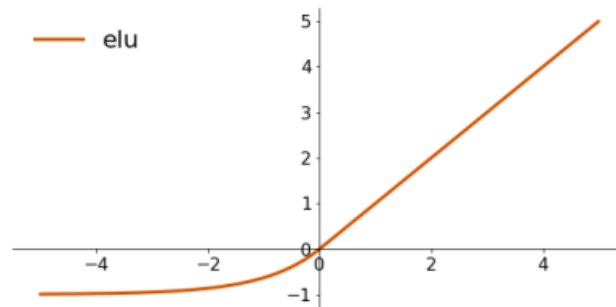
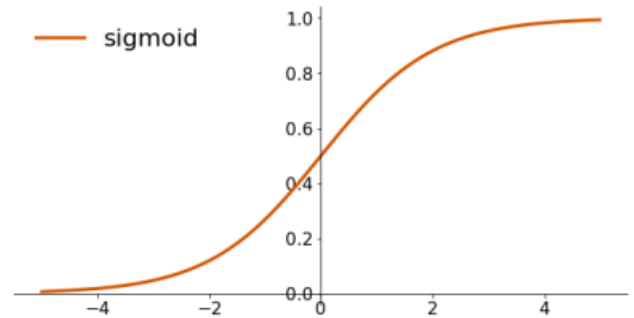
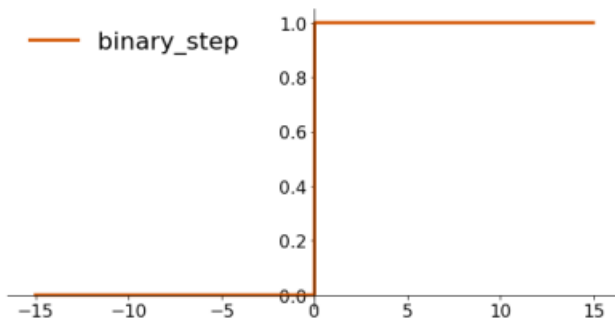
1 # Examples
2 calc_f1_score(tp=2, fp=3, fn=4)
3 >> precision is 0.4
4 recall is 0.3333333333333333
5 f1-score is 0.3636363636363636
6
7 calc_f1_score(tp='a', fp=3, fn=4)
8 >> tp must be int
9
10
11 calc_f1_score(tp=2, fp='a', fn=4)
12 >> fp must be int
13
14
15 calc_f1_score(tp=2, fp=3, fn='a')
16 >> tp must be int
17
18 calc_f1_score(tp=2, fp=3, fn=0)
19 >> tp and fp and fn must be greater than zero
20
21 calc_f1_score(tp=2.1, fp=3, fn=0)
22 >> tp must be int

```

Code Listing 2: Đây là các ví dụ các bạn không cần thiết đặt tên giống ví dụ

### 3. Viết function mô phỏng theo 3 activation function.:

- Binary Step Function  $f(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$
- Sigmoid Function  $f(x) = \frac{1}{1 + e^{-x}}$
- Elu Function  $f(x) = \begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$



- Input:
  - Người dùng nhập giá trị **x**
  - Người dùng nhập tên **activation function** chỉ có 3 loại (**binary, sigmoid, 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 = 'binary'**. Output: **print 'binary: f(3)=1'**

**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 (**binary, sigmoid, elu**). Nếu không print **'ten\_function\_user is not supported'** (vd người dùng nhập 'relu' thì print 'relu 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
- $\alpha = 0.01$

```

1 # Given
2 def is_number(n):
3     try:
4         float(n)      # Type-casting the string to 'float'.
5                        # If string is not a valid 'float',
6                        # it'll raise 'ValueError' exception
7     except ValueError:
8         return False
9     return True

```

Code Listing 3: Cho trước hàm `is_number`

```

1 exercise3()
2 >> Input x = 1.5
3 Input activation Function (binary|sigmoid|elu): sigmoid
4 sigmoid: f(1.5) = 0.8175744761936437
5
6 exercise3()
7 >> Input x = abc
8 x must be a number
9
10 exercise3()
11 >> Input x = 1.5
12 Input activation Function (binary|sigmoid|elu): relu
13 relu is not supported

```

Code Listing 4: Đây là các ví dụ các bạn không cần thiết đặt tên giống ví dụ

#### 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  $\hat{y}_i$  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.

**Note:** Các bạn lưu ý

- 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). Nếu **không hợp lệ print 'number of samples must be an integer number'** và dùng 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 giản 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

```

1 exercise4()
2 >> Input number of samples (integer number) which are generated: 5
3 Input loss name: RMSE
4 loss name: RMSE, sample: 0, pred: 6.659262156575629, target: 4.5905830130732355,
   loss: 4.279433398761796
5 loss name: RMSE, sample: 1, pred: 4.592264312227207, target: 8.447168720237958,
   loss: 14.860287994900718
6 loss name: RMSE, sample: 2, pred: 8.701801828625959, target: 9.280646891626386,
   loss: 0.3350616069599687
7 loss name: RMSE, sample: 3, pred: 4.799972972282257, target: 9.877147335937869,
   loss: 25.777699518961764
8 loss name: RMSE, sample: 4, pred: 0.20159822778697878, target: 5.540221923628147,
   loss: 28.50090296579681
9 final RMSE: 3.8406610234536727
10
11
12 exercise4()
13 >> Input number of samples (integer number) which are generated: aa
14 number of samples must be an integer number

```

Code Listing 5: Đây là các ví dụ các bạn không cần thiết đặt tên giống ví dụ

## 5. (OPTIONAL) Viết các function thực hiện tam giác Pascal và dãy số Fibonacci.

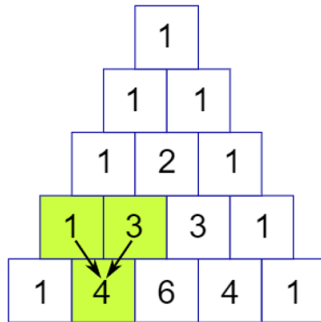
### Pascal's Triangle

- Input: level là số lượng hàng
- Output: Print ra kết quả từng hàng theo Pascal's Triangle

### Fibonacci Sequence

- Input: length là độ dài của chuỗi Fibonacci
- Output: Print ra kết quả các elemet trong chuỗi Fibonacci
- Các bạn nên tìm hiểu các công thức có thể thực hiện được Pascal's Triangle và Fibonacci Sequence

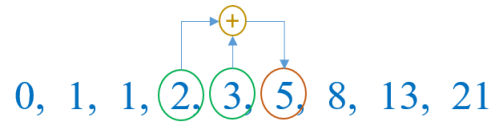
## Pascal's Triangle



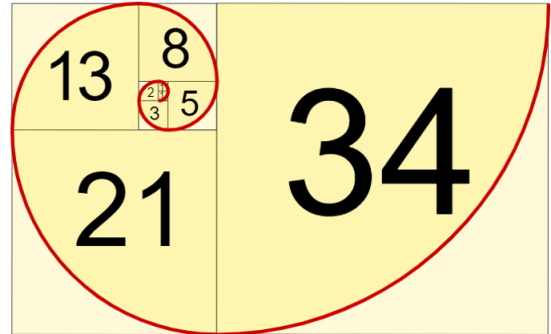
Level = 5

<https://www.mathsisfun.com>

## Fibonacci Sequence



Length = 9



- Nếu các bạn có kiến thức về list hoặc các kiến thức khác chưa có trong bài học thì các bạn vẫn sử dụng được

**NOTE:** Các bạn có thể dùng function in như sau để in kết quả trên cùng 1 hàng

```
1 value1 = 10
2 value2 = 20
3 value3 = 30
4 print(value1, end=' ')
5 print(value2, end=' ')
6 print(value3, end=' ')
7
8 >> 10 20 30
```

Code Listing 6: Print value trên cùng 1 hàng

```
1 calc_pas(level=5)
2 >> 1
3 1 1
4 1 2 1
5 1 3 3 1
6 1 4 6 4 1
7
8 calc_fib(length=9)
9 >> Fibonacci sequence:
10 0 1 1 2 3 5 8 13 21
```

Code Listing 7: Đây là các ví dụ các bạn không cần thiết đặt tên giống ví dụ

### 6. (OPTIONAL) Viết 4 functions để ước lượng các hàm số sau.

- Input: x (số muốn tính toán) và n (số lượng bậc muốn xấp xỉ)
- Output: Kết quả ước lượng hàm tương ứng với x. Ví dụ hàm  $\cos(x=0)$  thì output = 1

**NOTE:** Các bạn chú ý các điều kiện sau

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

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

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

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

- x là radian
- n là số nguyên dương  $\geq 0$
- các bạn nên viết một hàm tính giai thừa riêng

```

11 approx_sin(x=3.14, n=10)
12 >> 0.0015926529267151343
13
14 approx_cos(x=3.14, n=10)
15 >> -0.9999987316527259
16
17 approx_sinh(x=3.14, n=10)
18 >> 11.53029203039954
19
20 approx_cosh(x=3.14, n=10)
21 >> 11.573574828234543

```

Code Listing 8: Đây là các ví dụ các bạn không cần thiết đặt tên giống ví dụ

7. (OPTIONAL) Cho một số nguyên dương n, viết phương trình đảo ngược thứ tự các vị trí trong số n. Chỉ dùng while (hay for) và những phép toán cơ bản như +, -, \*, /, %, // ...

- Input: n là một dãy số nguyên dương.
- Output: Đảo ngược vị trí các số trong n. Ví dụ input: 12345678910, output: 1987654321

**NOTE:** Các bạn chú ý các điều kiện sau

- Không được ép kiểu sang string
- Chỉ sử dụng while hoặc for loop

```

22 reverse_number(n=12345678910)
23 >> 1987654321
24
25 reverse_number(n=123456789)
26 >> 987654321

```

Code Listing 9: Đây là các ví dụ các bạn không cần thiết đặt tên giống ví dụ