

Report - week 3

Le Xuan Duan

Ngày 28 tháng 12 năm 2024

1 Warm up - Exercise week 3

Câu hỏi 9: Function sau nhận vào 1 list các số và trả về giá trị nhỏ nhất trong các số đó. Bạn hãy hoàn thành function và chọn phương án đúng.

```
1 def get_min(data):
2     min_value = data[0]
3
4     ##### Your code here
5
6     #####
7
8     return min_value
9
10 data1 = [1, 5, 2, 91, -10]
11 assert get_min(data1) == -10
12
13 data2 = [2, 5, -5, 10, 11]
14 print(get_min(data2))
```

- a) 3
- b) -5
- c) 2
- d) 0

```
[1] def get_min(data):
    min_value = data[0]
    for i in range(len(data)):
        if data[i] < min_value:
            min_value = data[i]

    return min_value

data1 = [1, 5, 2, 91, -10]
assert get_min(data1) == -10

data2 = [2, 5, -5, 10, 11]
print(get_min(data2))
```

 -5

Câu hỏi 10: Function sau nhận vào 1 list các số và trả về giá trị lớn nhất trong các số đó. Bạn hãy hoàn thành function và chọn phương án đúng.

```
1 def get_max(data):
2     max_value = data[0]
3
4     ##### Your code here
5
6     #####
7
8     return max_value
9
10 data1 = [1, 5, 2, 91, -10]
11 assert get_max(data1) == 91
12
13 data2 = [2, 5, -5, 10, 11]
14 print(get_max(data2))
```

- a) 11
- b) 5
- c) 3
- d) 10

```
[2] def get_max(data):
    max_value = data[0]

    for i in range(len(data)):
        if data[i] > max_value:
            max_value = data[i]

    return max_value

data1 = [1, 5, 2, 91, -10]
assert get_max(data1) == 91

data2 = [2, 5, -5, 10, 11]
print(get_max(data2))
```

11

Câu hỏi 11: Function sau nhận vào 1 list các số và 1 số cho trước, function trả về số lần xuất hiện của số đó trong list. Bạn hãy hoàn thành function và chọn phương án đúng.

```
1 def count(data, value):
2     count = 0
3
4     ##### Your code here
5
6     #####
7
8     return count
9
10 data1 = [1, 5, 2, -10, 5, 10, 5]
11 assert count(data1, 5) == 3
12
13 data2 = [5, 4, 2, 10, 4, 8, 1, 3]
14 print(count(data2, 4))
```

- a) 1
- b) 0
- c) 3
- d) 2

```
[4] def count(data, value):
    count = 0

    for i in range(len(data)):
        if data[i] == value:
            count = count + 1

    return count

    data1 = [1, 5, 2, -10, 5, 10, 5]
    assert count(data1, 5) == 3

    data2 = [5, 4, 2, 10, 4, 8, 1, 3]
    print(count(data2, 4))
```

2

Câu hỏi 12: Function sau nhận vào 1 list các số và trả về tổng của các số đó. Bạn hãy hoàn thành function và chọn phương án đúng.

```
1 def get_sum(data):
2     sum_value = 0
3
4     ##### Your code here
5
6     #####
7
8     return sum_value
9
10 data1 = [1, 5, 2, 8, -1]
11 assert get_sum(data1) == 15
12
13 data2 = [5, 4, 2, 10, 4]
14 print(get_sum(data2))
```

- a) 13
- b) 25
- c) 11
- d) 22

```
def get_sum(data):
    sum_value = 0

    for i in data:
        sum_value = sum_value + i

    return sum_value

data1 = [1, 5, 2, 8, -1]
assert get_sum(data1) == 15

data2 = [5, 4, 2, 10, 4]
print(get_sum(data2))
```

25

```
[11] def get_sum(data):
      return sum(data)

data1 = [1, 5, 2, 8, -1]
assert get_sum(data1) == 15

data2 = [5, 4, 2, 10, 4]
print(get_sum(data2))
```

25

Câu hỏi 13: Function sau nhận vào 1 list các số và trả về tổng của các số chẵn. Bạn hãy hoàn thành function và chọn phương án đúng.

```
1 def get_sum_even(data):
2     sum_value = 0
3
4     ##### Your code here
5
6     #####
7
8     return sum_value
9
10 data1 = [1, 5, 2, 4, -10, 5]
11 assert get_sum_even(data1) == -4
12
13 data2 = [5, 4, 2, 10, 3, 8]
14 print(get_sum_even(data2))
```

- a) 22
- b) 18
- c) 24
- d) 26

```
def get_sum_even(data):
    sum_value = 0

    for i in range(len(data)):
        if data[i] % 2 == 0:
            sum_value = sum_value + data[i]

    return sum_value

data1 = [1, 5, 2, 4, -10, 5]
assert get_sum_even(data1) == -4

data2 = [5, 4, 2, 10, 3, 8]
print(get_sum_even(data2))
```

24

```
[9] def get_sum_even(data):
    sum_value = 0

    for value in data:
        if value % 2 == 0:
            sum_value += value

    return sum_value

data1 = [1, 5, 2, 4, -10, 5]
assert get_sum_even(data1) == -4

data2 = [5, 4, 2, 10, 3, 8]
print(get_sum_even(data2))
```

24

Câu hỏi 14: Function sau nhận vào 1 list các số và trả về giá trị trung bình của các số đó. Sử dụng hàm `round()` để làm tròn kết quả đến chữ số thập phân thứ 2. Bạn hãy hoàn thành function và chọn phương án đúng.

```
1 def get_mean(data):
2     sum_value = 0
3
4     ##### Your code here
5
6     #####
7
8     return mean
9
10 data1 = [1, 5, 2, 7, -3]
11 assert get_mean(data1) == 2.4
12
13 data2 = [2, 5, -5, 10, 11]
14 print(get_mean(data2))
```

- a) 4.6
- b) 3.2
- c) 4.4
- d) 3.8

```
[10] def get_mean(data):
      sum_value = 0

      for value in data:
          sum_value += value
          mean = round(sum_value / len(data), 2)
      return mean

      data1 = [1, 5, 2, 7, -3]
      assert get_mean(data1) == 2.4

      data2 = [2, 5, -5, 10, 11]
      print(get_mean(data2))
```

 4.6

```
[12] def get_mean(data):
      return round(sum(data)/len(data),2)

      data1 = [1, 5, 2, 7, -3]
      assert get_mean(data1) == 2.4

      data2 = [2, 5, -5, 10, 11]
      print(get_mean(data2))
```

 4.6

Câu hỏi 15: Function sau nhận vào 1 list các số và 1 số cho trước, function kiểm tra số đó có xuất hiện trong list hay không và trả về giá trị True hoặc False. Bạn hãy hoàn thành function và chọn phương án đúng.

```
1 def contain(data, value):
2     is_contain = False
3
4     ##### Your code here
5
6     #####
7
8     return is_contain
9
10 data1 = [1, 5, 2, 91, -10]
11 assert contain(data1, 5) == True
12
13 data2 = [2, 5, -5, 10, 11]
14 print(contain(data2, 3))
```

- a) True
- b) False


```
[15] def contain(data, value):
      is_contain = False


      for i in data:
          if i == value:
              is_contain = True
              break

      return is_contain

      data1 = [1, 5, 2, 91, -10]
      assert contain(data1, 5) == True


      data2 = [2, 5, -5, 10, 11]
      print(contain(data2, 3))
```

 False

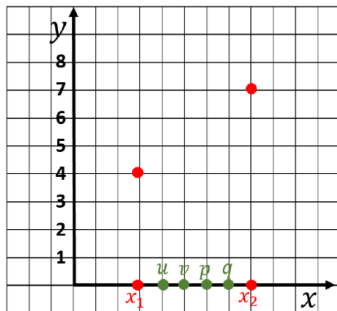
```
 def contain(data, value):
      return value in data

      data1 = [1, 5, 2, 91, -10]
      assert contain(data1, 5) == True

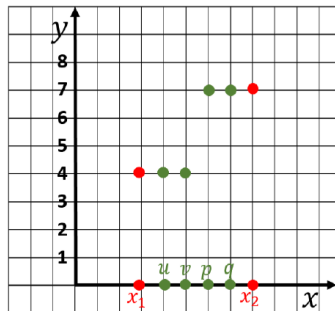
      data2 = [2, 5, -5, 10, 11]
      print(contain(data2, 3))
```

 False

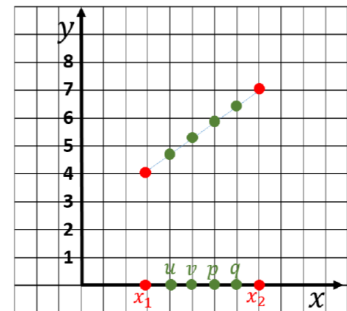
2 Interpolation in image resizer



Tim giá trị cho các vị trí u, v, p và q



Nearest neighbor: Tính khoảng cách đến x_1 và x_2 , và lấy giá trị của x gần hơn



Nội suy theo hàm tuyến tính

2.1 Nearest neighbor

Khi bạn muốn nội suy giá trị tại một vị trí chưa được biết, NN tìm điểm dữ liệu có khoảng cách gần nhất đến vị trí đó và sử dụng giá trị của điểm gần nhất này.

NN thích hợp để scale hình ảnh hoặc dữ liệu khi độ chính xác không phải là ưu tiên cao. Thường xử lý dữ liệu không liên tục hoặc có tính chất phân đoạn vì ưu điểm nhanh và dễ thực hiện

Bài toán đặt ra là từ một ma trận 3x3 chuyển thành một ma trận 6x6

```
source_h = 3
source_w = 3
target_h = 6
target_w = 6

source = [[1,3,7],
          [2,5,6],
          [9,4,1]]

#step 1
# tạo ma trận mới, duyệt qua 3 hàng (source_h)
# trong mỗi 3 hàng, duyệt 6 cột mới và gán giá trị = 0
# như vậy ta có ma trận mới với 3 hàng và 6 cột
new_data_1 = [[0]*target_w for _ in range(source_h)]
for i in range(source_h):
    for j in range(target_w):
        new_data_1[i][j] = source[i][j//2] # giá trị mới hàng i, index j = giá trị cũ hàng i index j//2

#step 2
# tạo một ma trận mới 6x6 từ ma trận cũ 3x6
new_data_2 = [[0]*target_w for _ in range(target_h)]
for j in range(target_w):
    for i in range(target_h):
        new_data_2[i][j] = new_data_1[i//2][j] # giá trị mới hàng i, index j = giá trị cũ hàng i index j//2

[ ] for row in new_data_2:
    print(row)
```

```
[1, 1, 3, 3, 7, 7]
[1, 1, 3, 3, 7, 7]
[2, 2, 5, 5, 6, 6]
[2, 2, 5, 5, 6, 6]
[9, 9, 4, 4, 1, 1]
[9, 9, 4, 4, 1, 1]
```


2.2 Linear interpolation

Linear interpolation (nội suy tuyến tính) sử dụng phương trình đường thẳng để ước tính giá trị giữa hai điểm dữ liệu đã biết.

Với 2 điểm x_1, y_1 và x_2, y_2 cho trước, giá trị y tại một điểm x bất kỳ giữa x_1 và x_2 được tính theo công thức:

$$y = y_1 + \frac{(x-x_1)}{(x_2-x_1)} \cdot (y_2 - y_1)$$

Bài toán đặt ra: trong một list, tìm cách vị trí None và sử dụng linear interpolation để điền vào các giá trị

```
def hoanthien(data):
    # Tìm vị trí của None và hoàn thiện dần dần
    while None in data:
        # Tìm vị trí của None
        index = -1
        for count, value in enumerate(data):
            if value is None:
                index = count
                break

        # Tìm vị trí của begin
        begin = index - 1
        while begin >= 0 and data[begin] is None:
            begin -= 1

        # Tìm vị trí của end
        end = index + 1
        while end < len(data) and data[end] is None:
            end += 1

        # Xử lý trường hợp đặc biệt nếu begin hoặc end ngoài phạm vi hợp lệ
        if begin < 0:
            p1 = (0, data[end])
            p2 = (end, data[end])
        elif end >= len(data):
            p1 = (begin, data[begin])
            p2 = (begin + 1, data[begin])
        else:
            p1 = (begin, data[begin])
            p2 = (end, data[end])

        # Tính giá trị nội suy
        y = linear(p1, p2, index)
        data[index] = y

    return data

data = [None, 0, 2, None, 3, 4, 5, None]
print(hoanthien(data))
```

⇒ [0.0, 0, 2, 2.5, 3, 4, 5, 5.0]