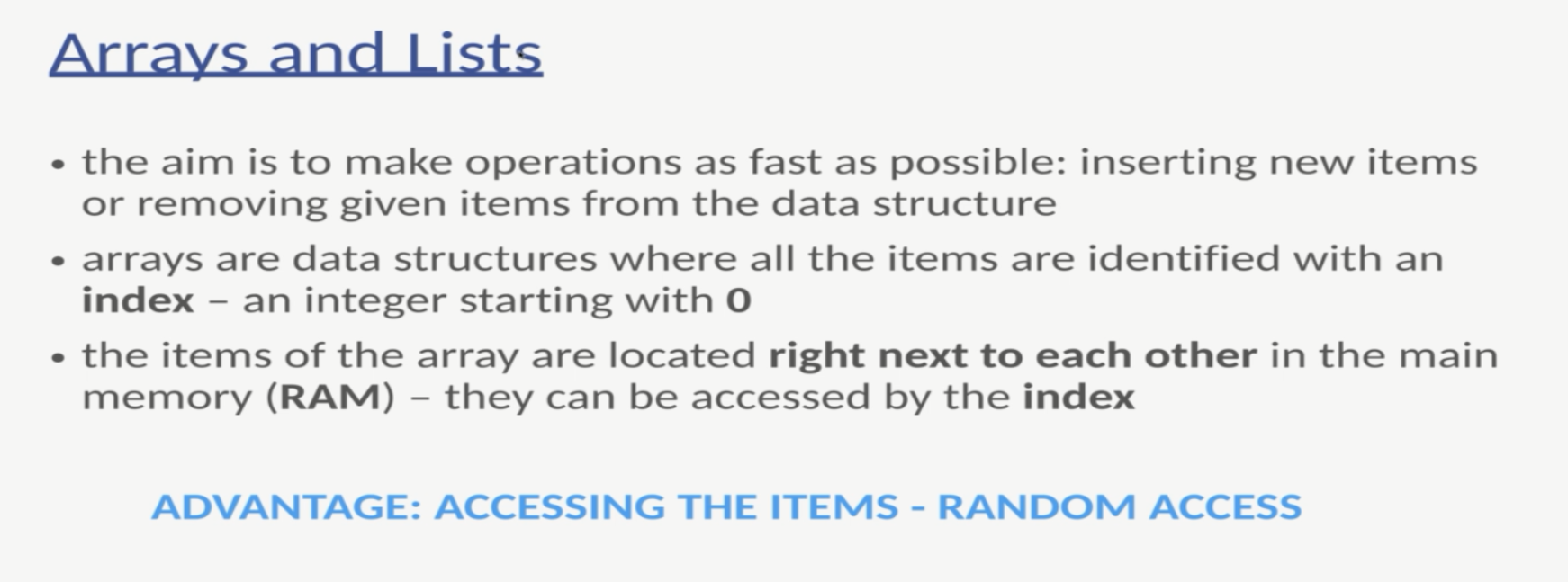
Class: N21CQDK-01N

Name: Nguyễn Đức Hào

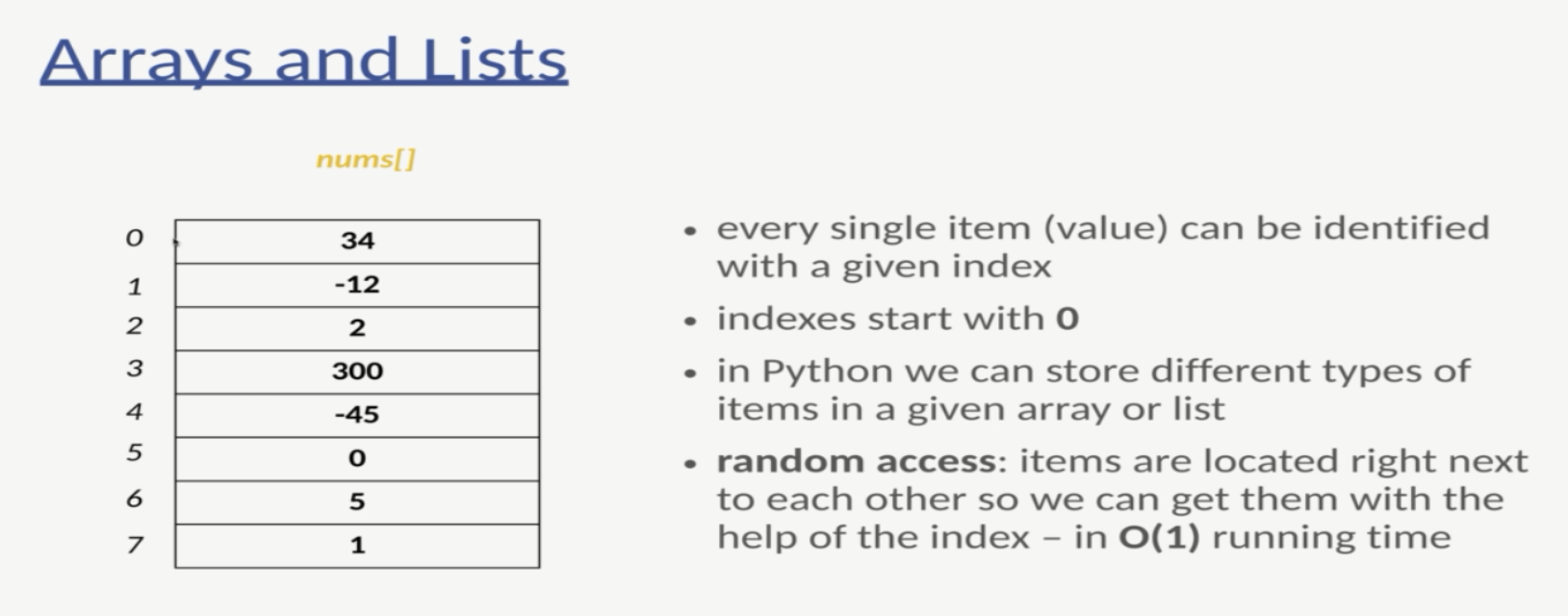
HOMEWORK



1. Introduction



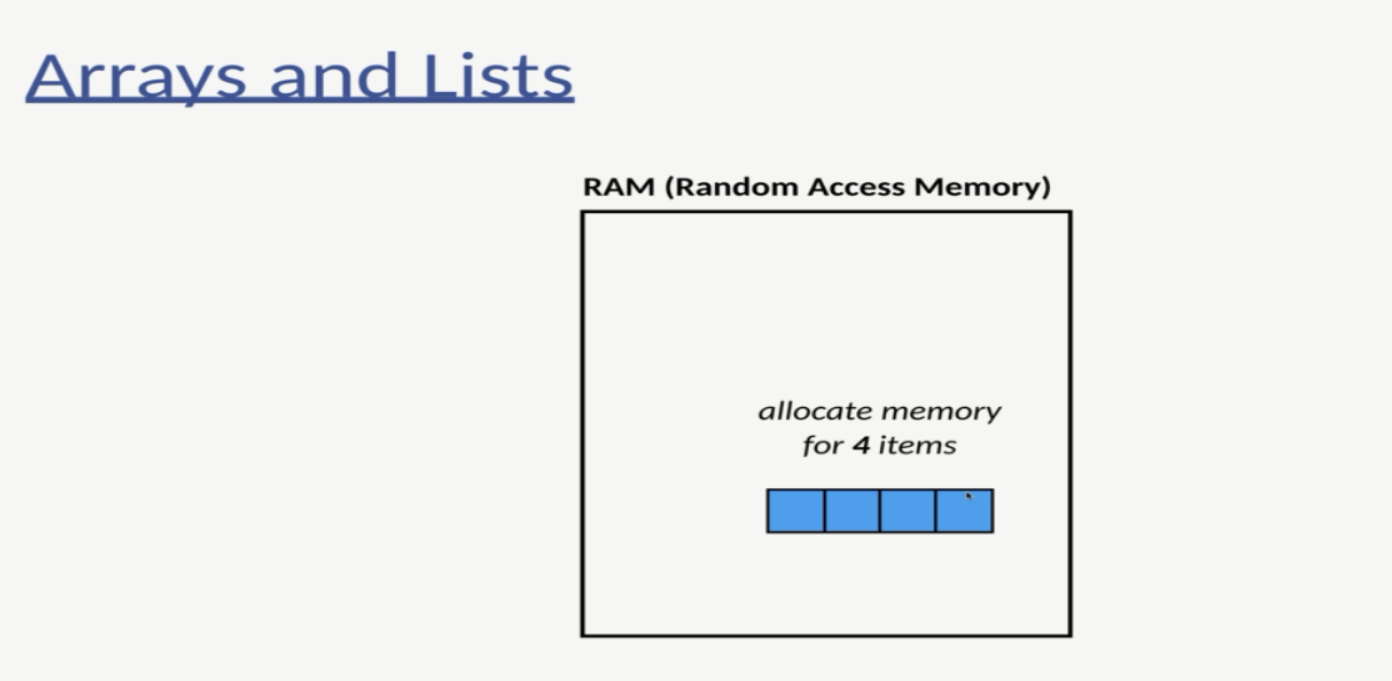
* Mục đích là thực hiện các thao tác nhanh nhất, có thể chèn hoặc xóa các mục đã cho khỏi cấu trúc dữ liệu
* Arrays là cấu trúc dữ liệu trong đó được xác định bằng 1 số nguyên bắt đầu bằng 0
* Các mục (items) sắp xếp nằm cạch nhau trong bộ nhớ chính (RAM) chúng có thể được truy cập bằng chỉ mục (index)



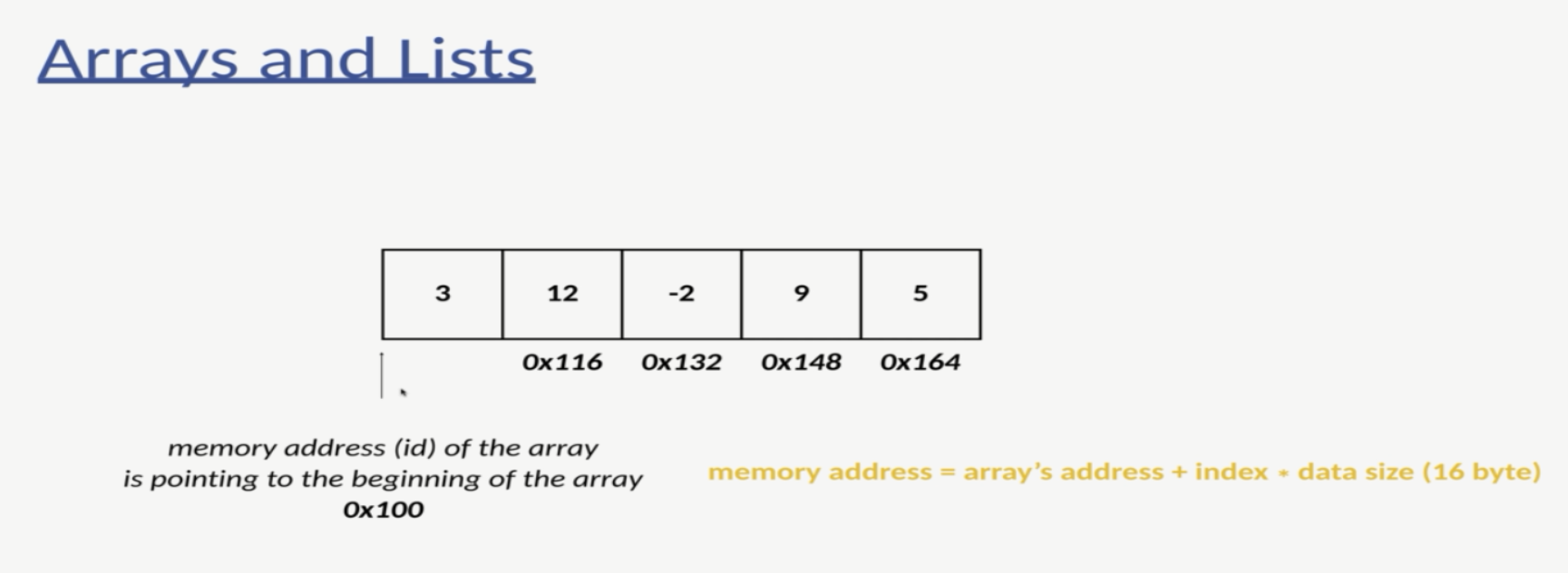
* Mỗi item được xác định bằng các chỉ mục xác định

For example: nums[0] = 34; nums[4] = -45 etc

* Trong một mảng(arrays) phần tử truy cập đầu tiên là 0
* Với độ truy cập ngẫu nhiên và các Items được sắp xếp cạnh nhau với độ phức tạp là 0(1)
* Trong python có thể lưu nhiều kiểu(types) trong một array hoặc list nhất định



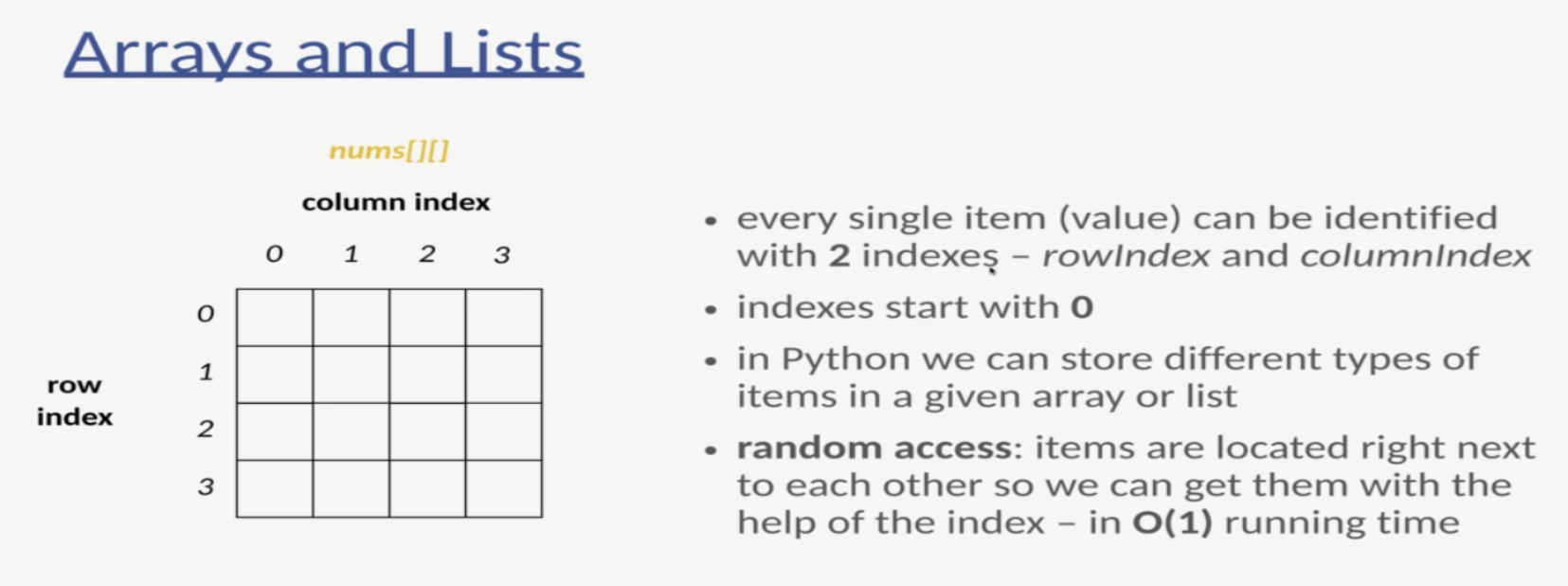
* Nó sẽ truy cập vào bộ nhớ chính (RAM) và các items được sắp xếp nằm liền kề nhau.



Cách tính địa chỉ của 1 bit như sau: địa chỉ 1 bit cần tính = bằng địa chỉ đầu tiên trong một arrays + (chỉ số index x data size)

For example: như hình ở trên muốn biết địa chỉ của bit (hexa) của phần tử thứ 3 và 4 như sau

* 0x100 + (3 x 16) = 0x148
* 0x100 + (4 X 16) = 0x164

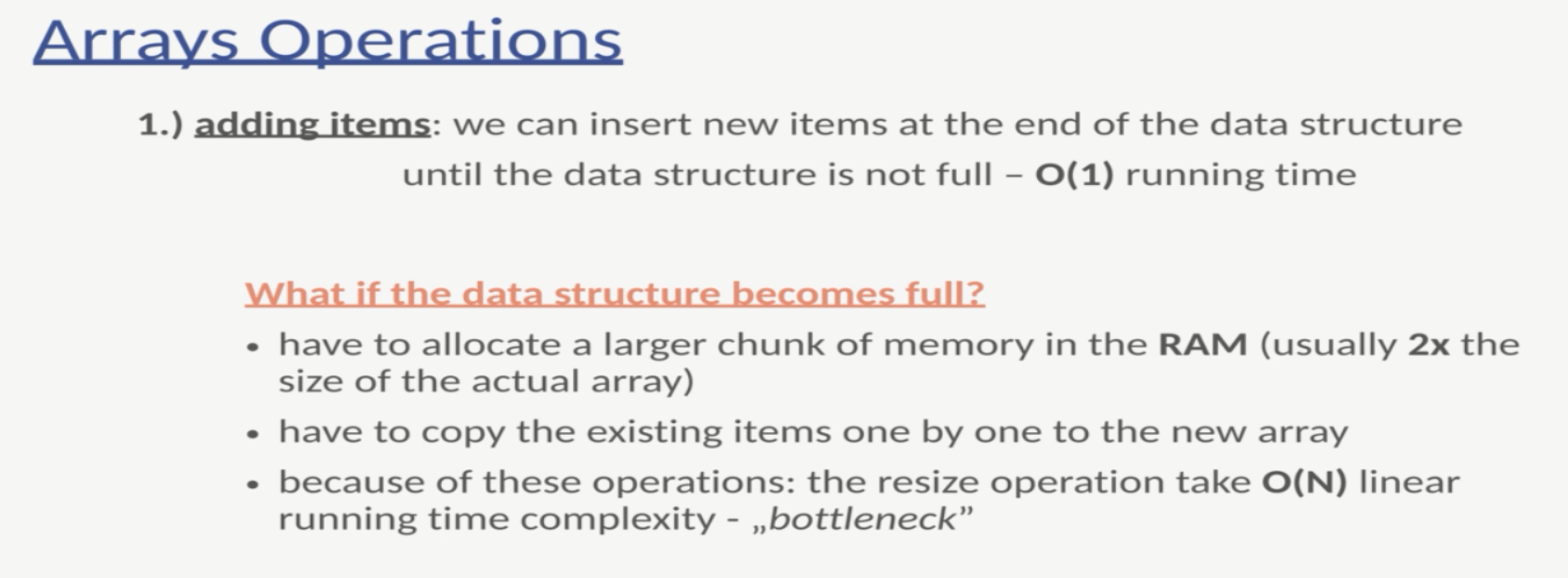


Đây là mảng hai chiều gồm hai ma trận hàng và cột với độ phức tap là 0(1), chỉ địa chỉ của phần tử như nhau noums[hàng][cột]

Ví dụ như noums[3][4] thì nhìn vào ma trận minh họa ở trên thì nó ở vị trí hàng 3 cột 4



Mảng thông thường đều là tỉnh (static array) nếu muốn trở thành động (dynamic array) thì ta phải cấp phát bộ nhớ động cho nó thường cách này không khuyên dùng vì sẽ gây ra lỗi tràn bộ nhớ và độ phứ tạp là 0(n) sẽ mất thời gian hơn.





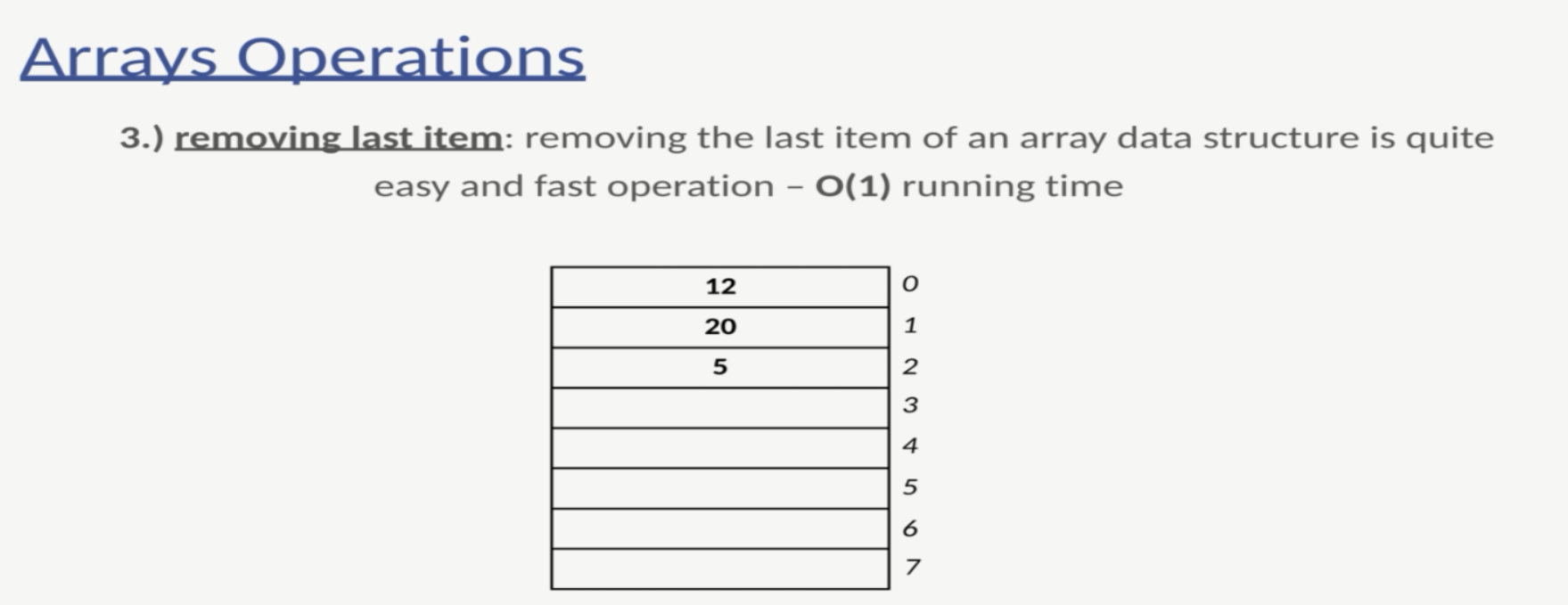
Hành động thêm items với độ phức tạp -O(1) ví dụ a[3] = 7

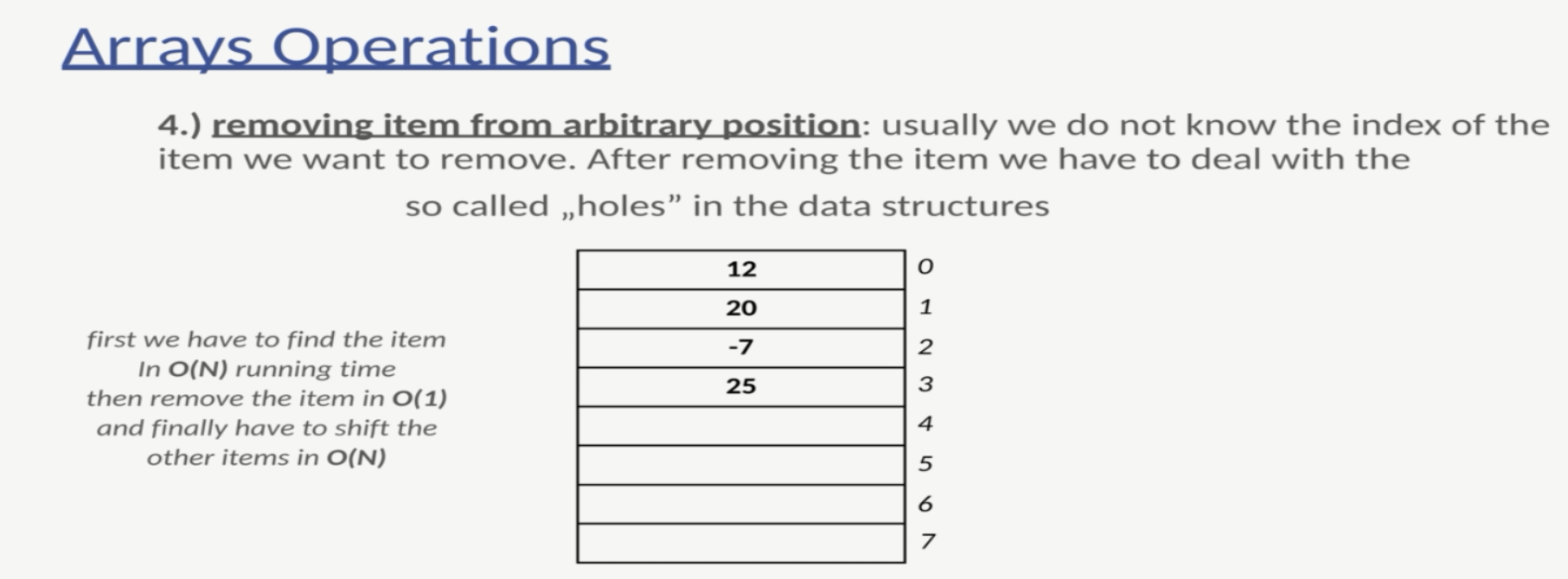
Sẽ ra sao nếu lưu các phần tử quá kích thước của mảng trong Ram (thông thường nó sẽ x2 giá trị ban đầu) và độ phức tạp là O(n)).

Array chỉ thêm được các giá trị có cùng khiểu

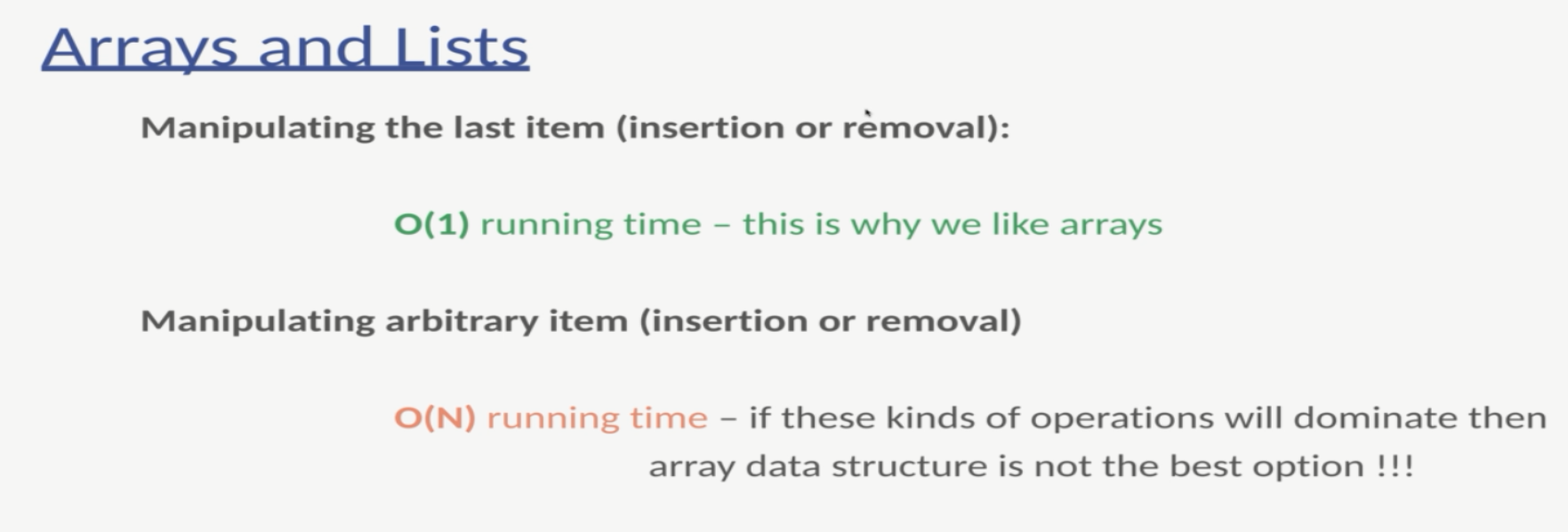


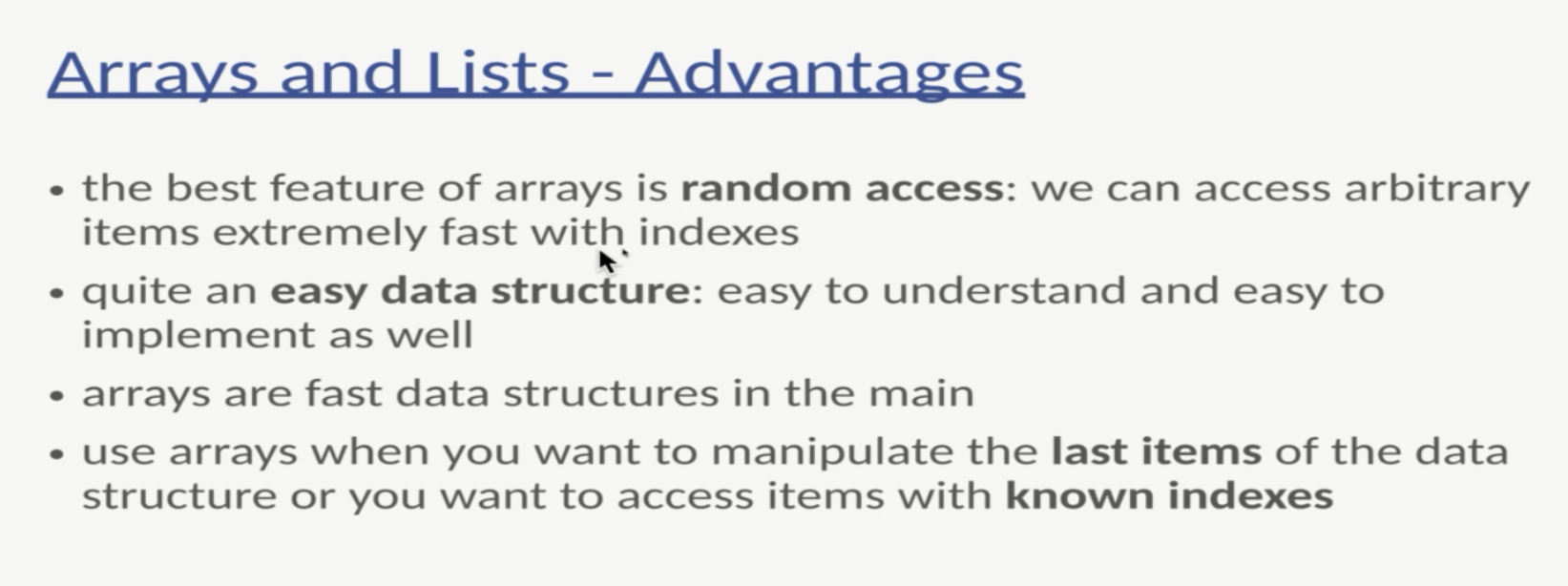
* Với việc thêm phần tử ở vị trí ngẫu nhiên thì độ phức tạp sẽ là O(n) vì cần di chuyển tất cả các phần tử sau vị trí đó nên độ phức tạp là O(n)
* Lưu ý nếu ta thêm phần từ nằm ở cuối cùng ở mảng thì sẽ không có phần tử để dịch nên độ phức tạp sẽ là O(1).





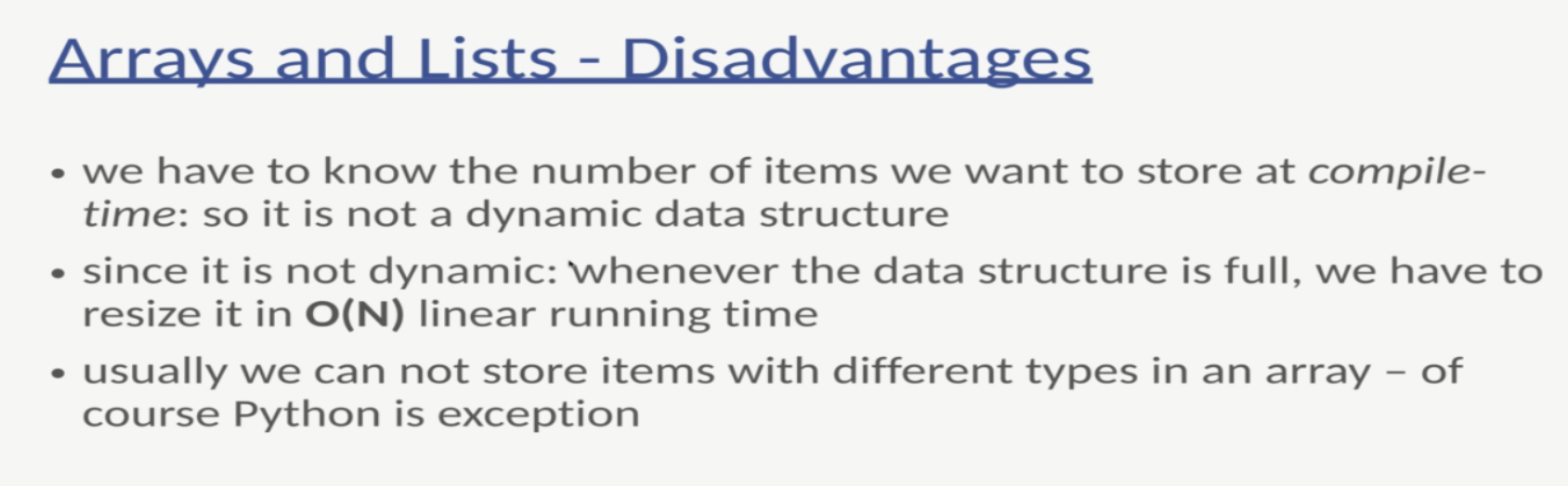
* Ngyên tắt của xóa đi 1 phần tử cũng giống như cách ta thêm vào nếu thêm xóa ở vị trí bất kì là 0(n) vì phải dịch các phần tử sau nó lại và O(1) nếu xóa vị trí cuối và không ảnh đến các phần trước nó.



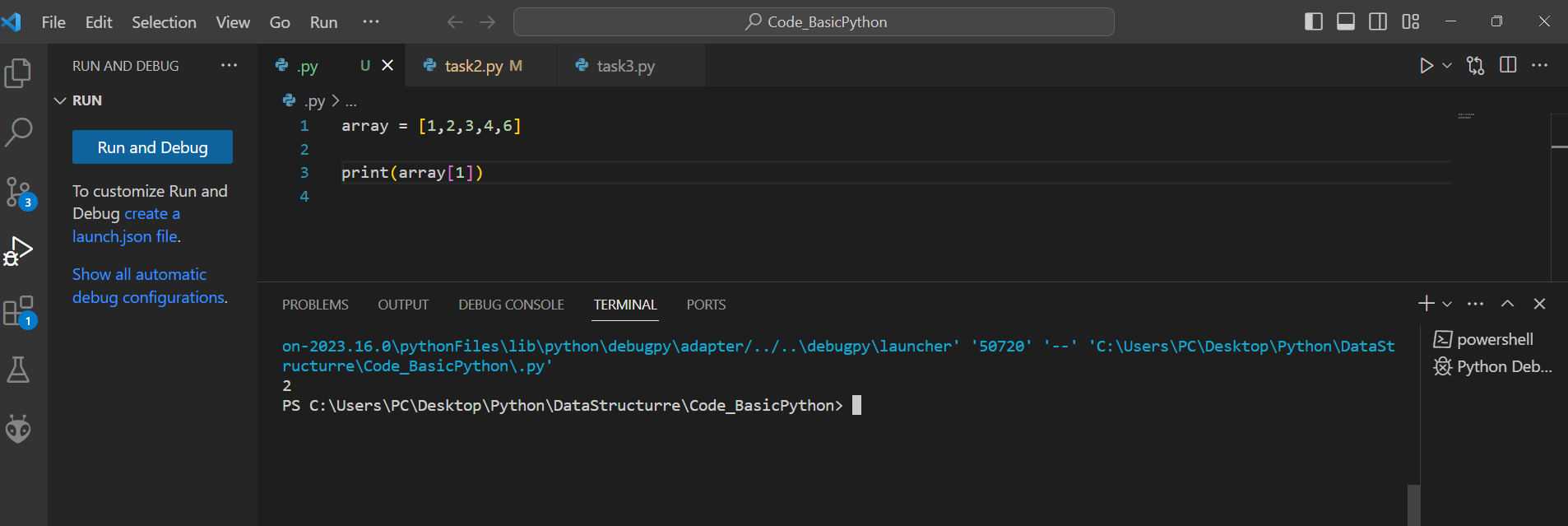


Tóm tắt lại thao tác với mục cuối (items) cho việc chèn hoặc xóa độ phức tặp sẽ là O(1) vì có tính đơn giản nên thơi gian thực thi nhanh và linh hoạt.

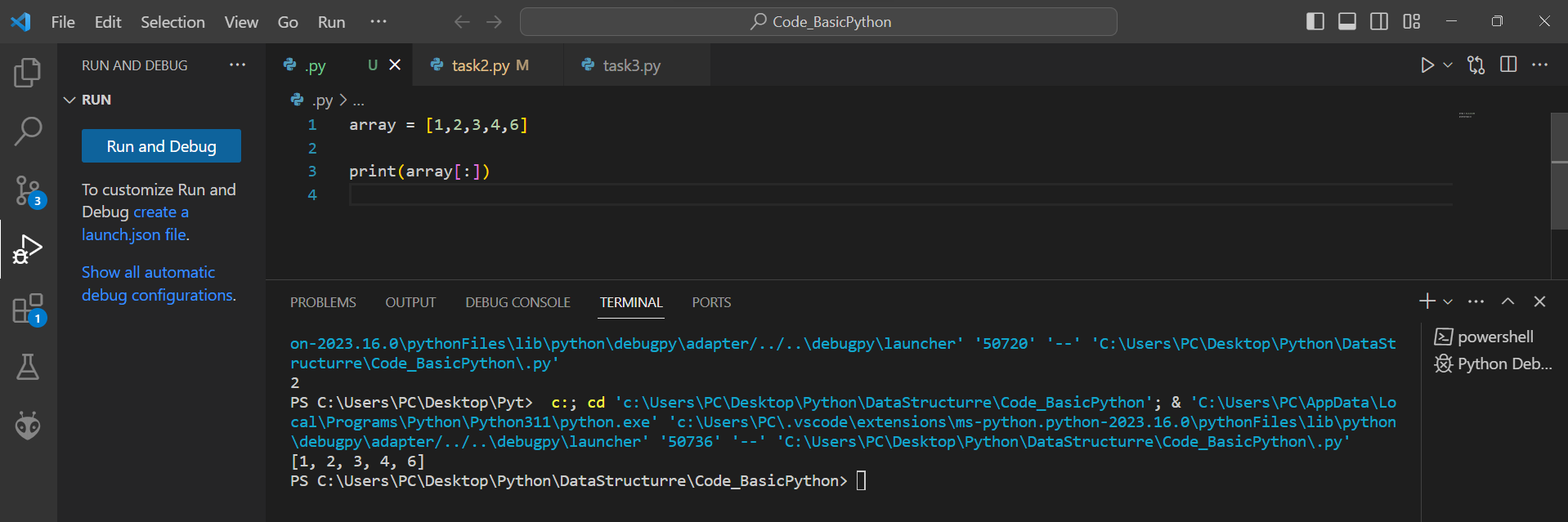
Còn đối với việc thao tác với các mục tùy ý độ phức tap sẽ là O(N) nên dẫn tới nhiều hạn chế không chiếm ưu thế nhiều thời gian thực thi lâu và phức tạp



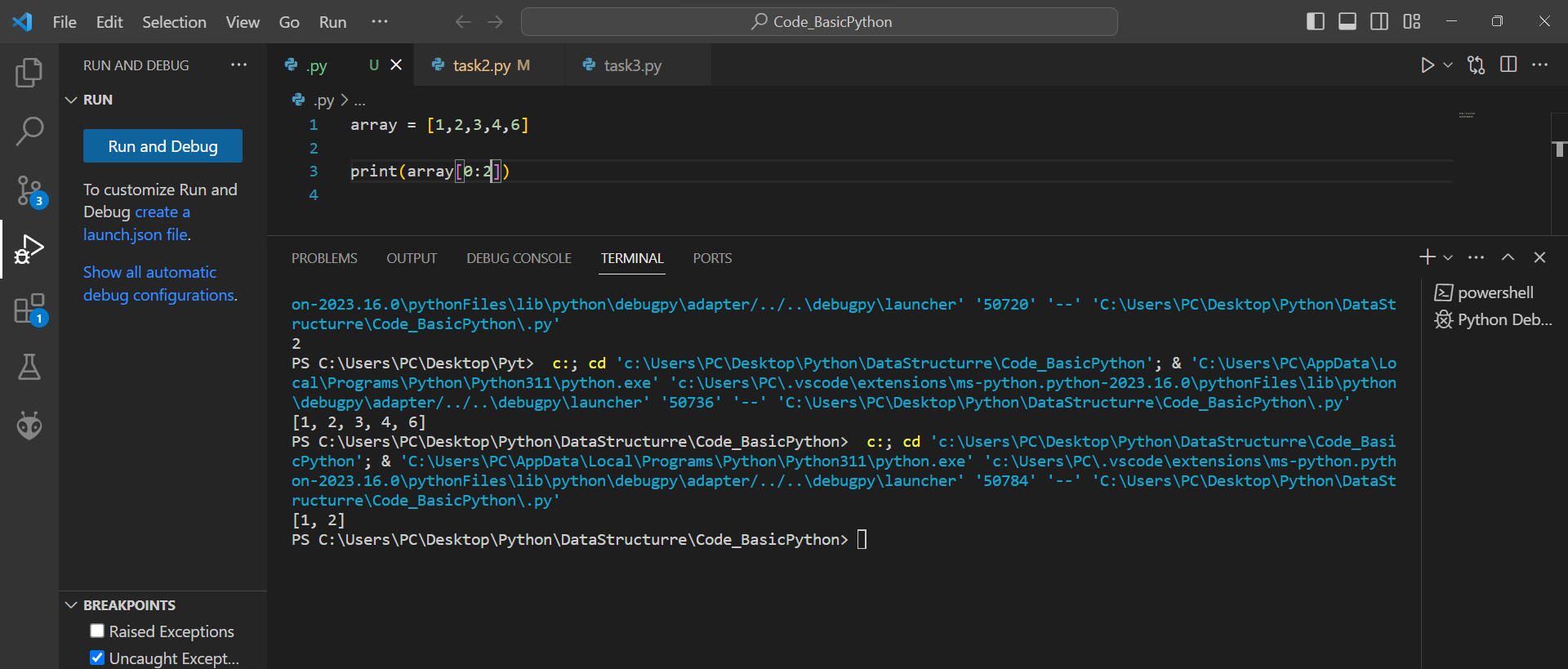
* Phải biết số lượng phần tử cần lưu ngay từ đầu, không linh hoạt.
* Khi đầy phải mở rộng kích thước trong O(N), tốn kém.
* Thường không thể lưu các kiểu dữ liệu khác nhau trong cùng một mảng.

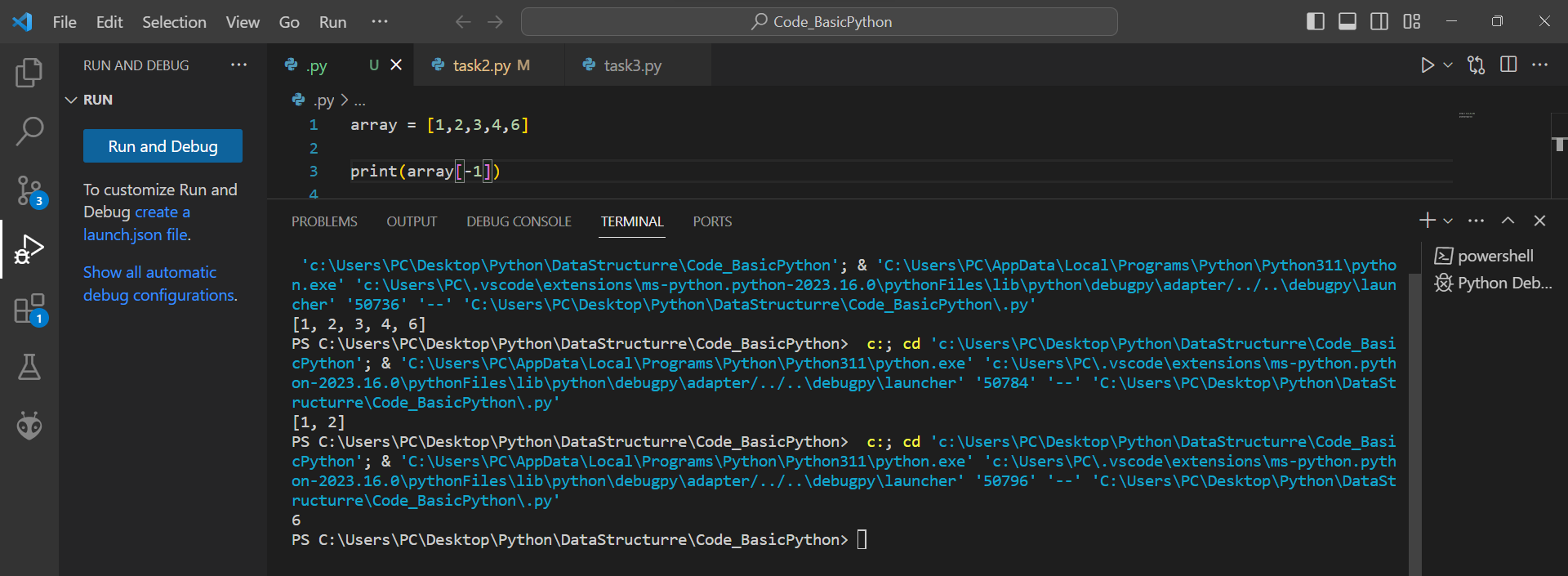


Thứ tự tương ứng trong hàm khai báo array sẽ được có 5 phần tử và bắt đầu từ 0 –> 4. Với câu lệnh in ra màn hình print(array[1]) kết quả in ra là 2.

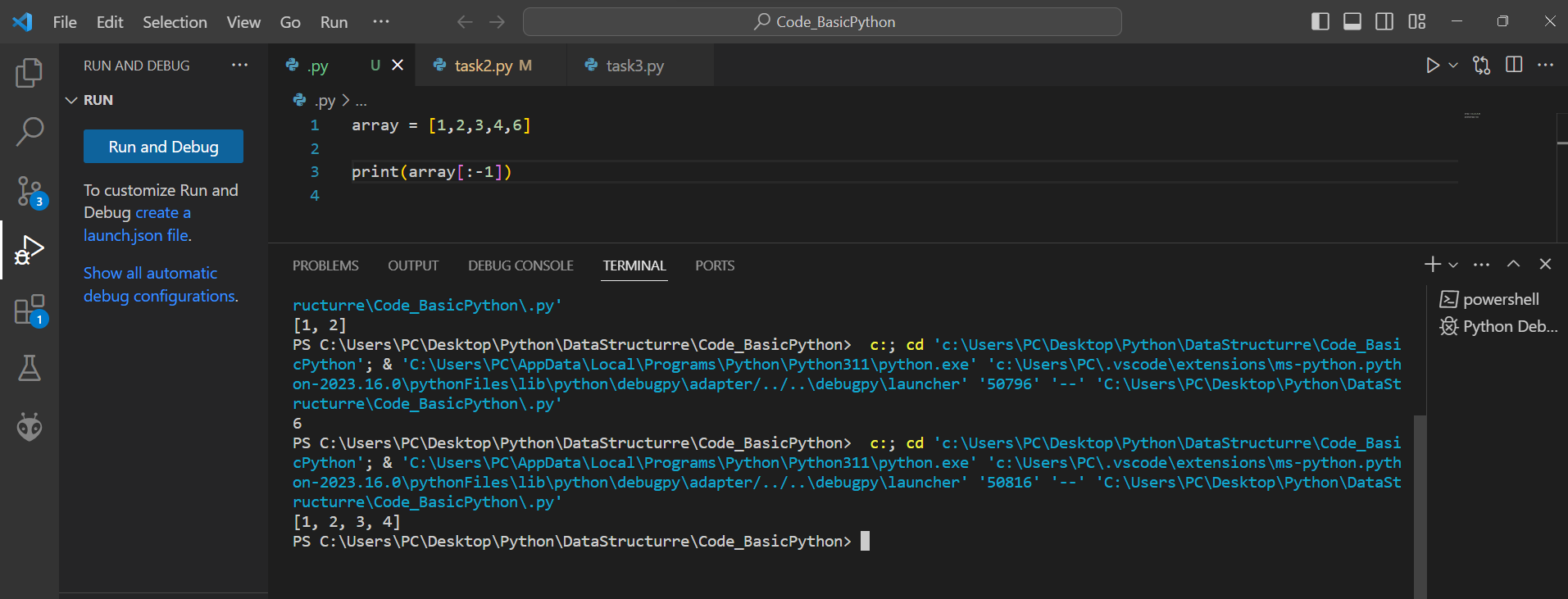


Và khi thực hiện câu lệnh print(array[:]) là duyệt tất cả các phần tử trong mảng và in toàn bộ ra màn hình.





Với lệnh print(array[-1]) câu lệnh sẽ thực hiện in ra màn hình theo thứ tự ngược lại là 6.

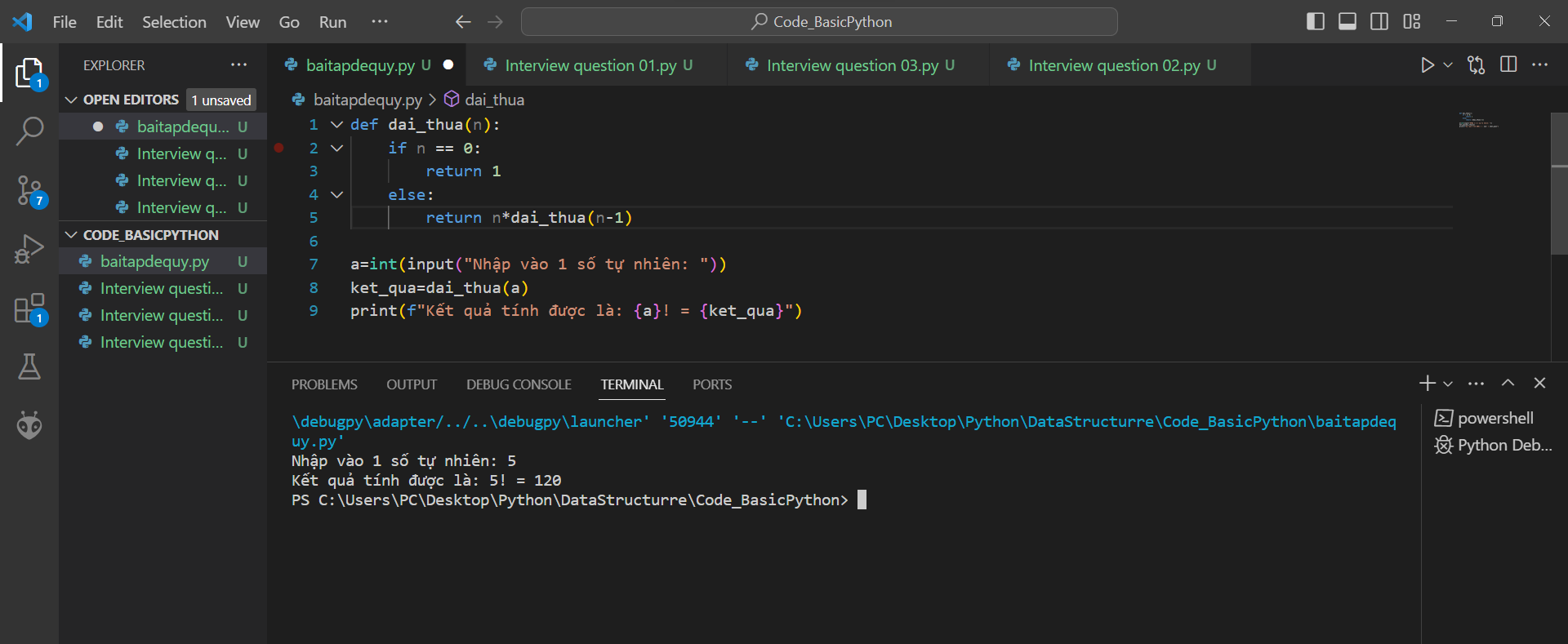


Với lệnh print(array[:-1]) sẽ được thực hiện in ra màn hình trừ phần tử cuối. Vậy tổng quát nếu là [:-n] sẽ lấy hết trừ lấy n phần tử cuối.

* Kiểu list phần tử khác kiểu dữ liệu
* Array phải giống kiểu dữ liệu
* Bài tập:

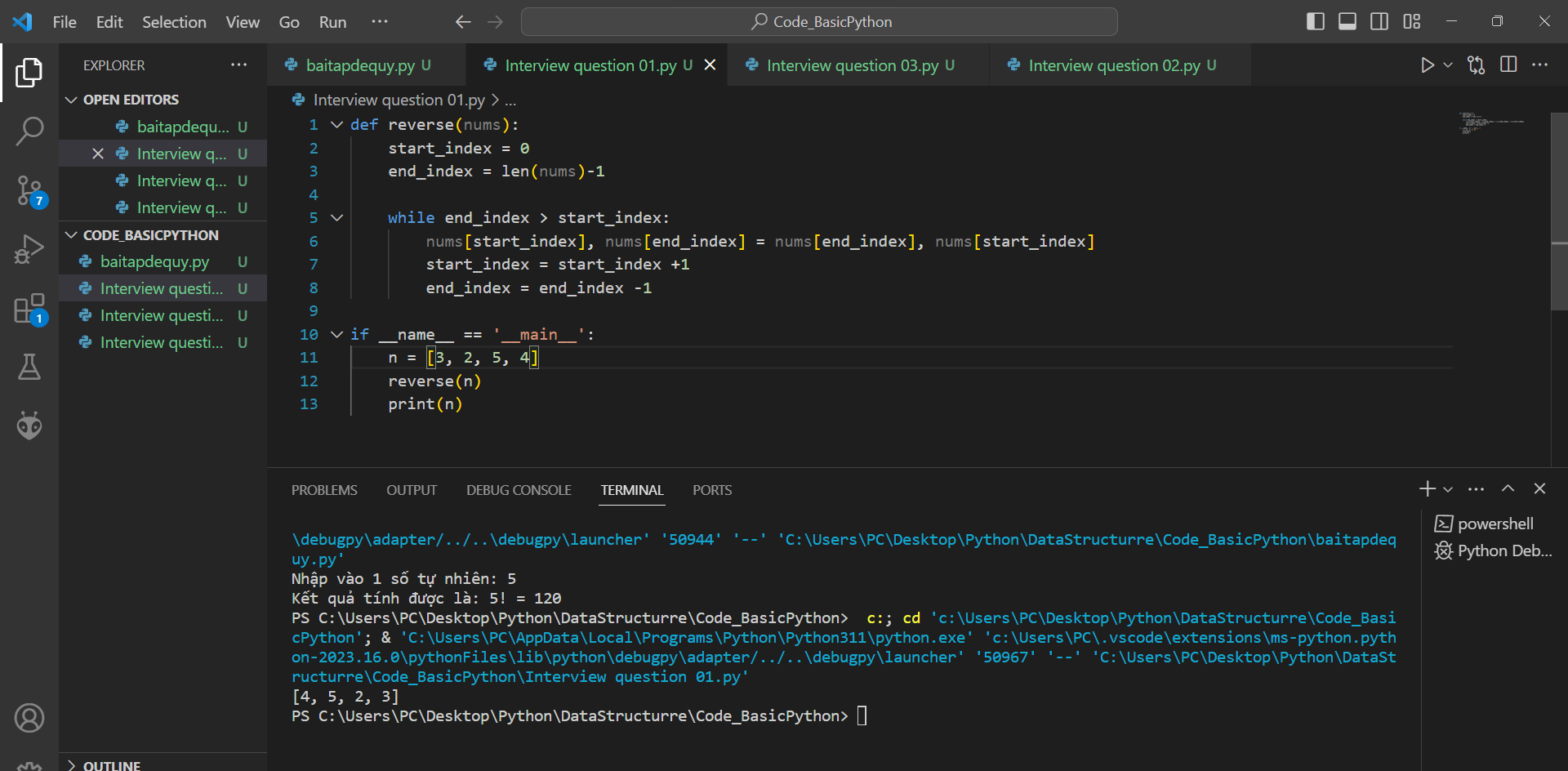
Bài tập viết đệ quy: TÍnh giai thừa

https://github.com/Haoduc/HomeWork/blob/0b77c6fd6600e10055f9dd21ef73eab49f97650c/baitapdequy.py



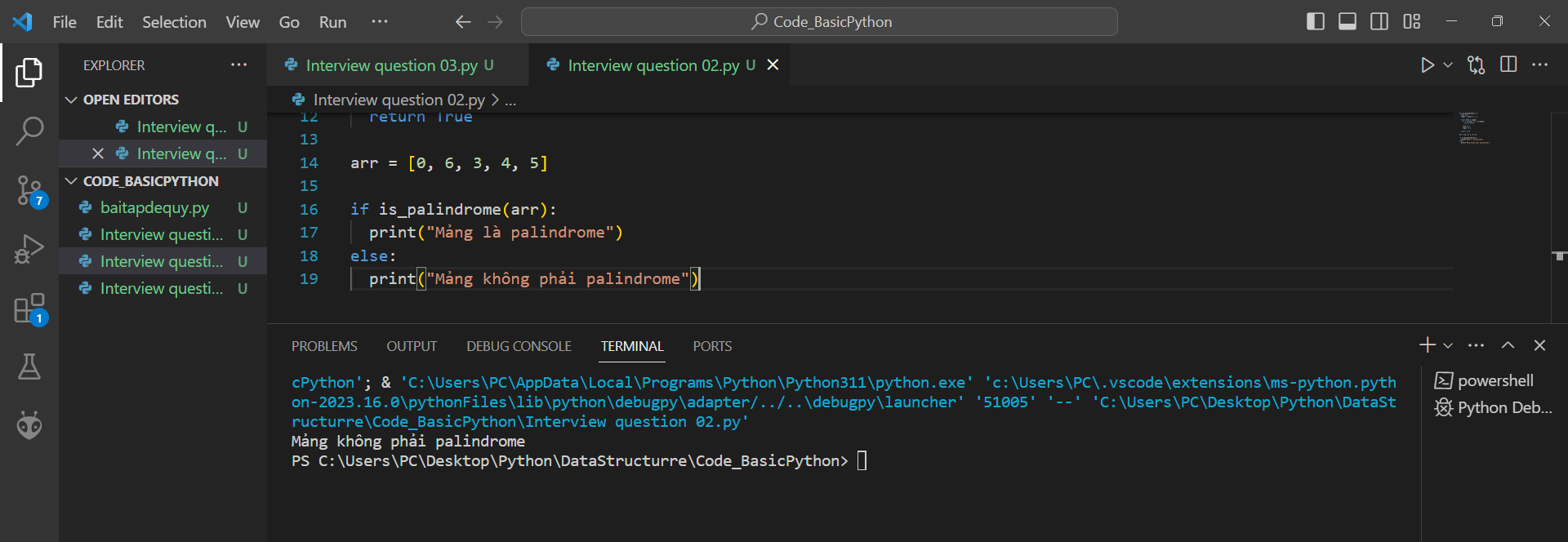
Interview question 01

https://github.com/Haoduc/HomeWork/blob/0b77c6fd6600e10055f9dd21ef73eab49f97650c/Interview%20question%2001.py



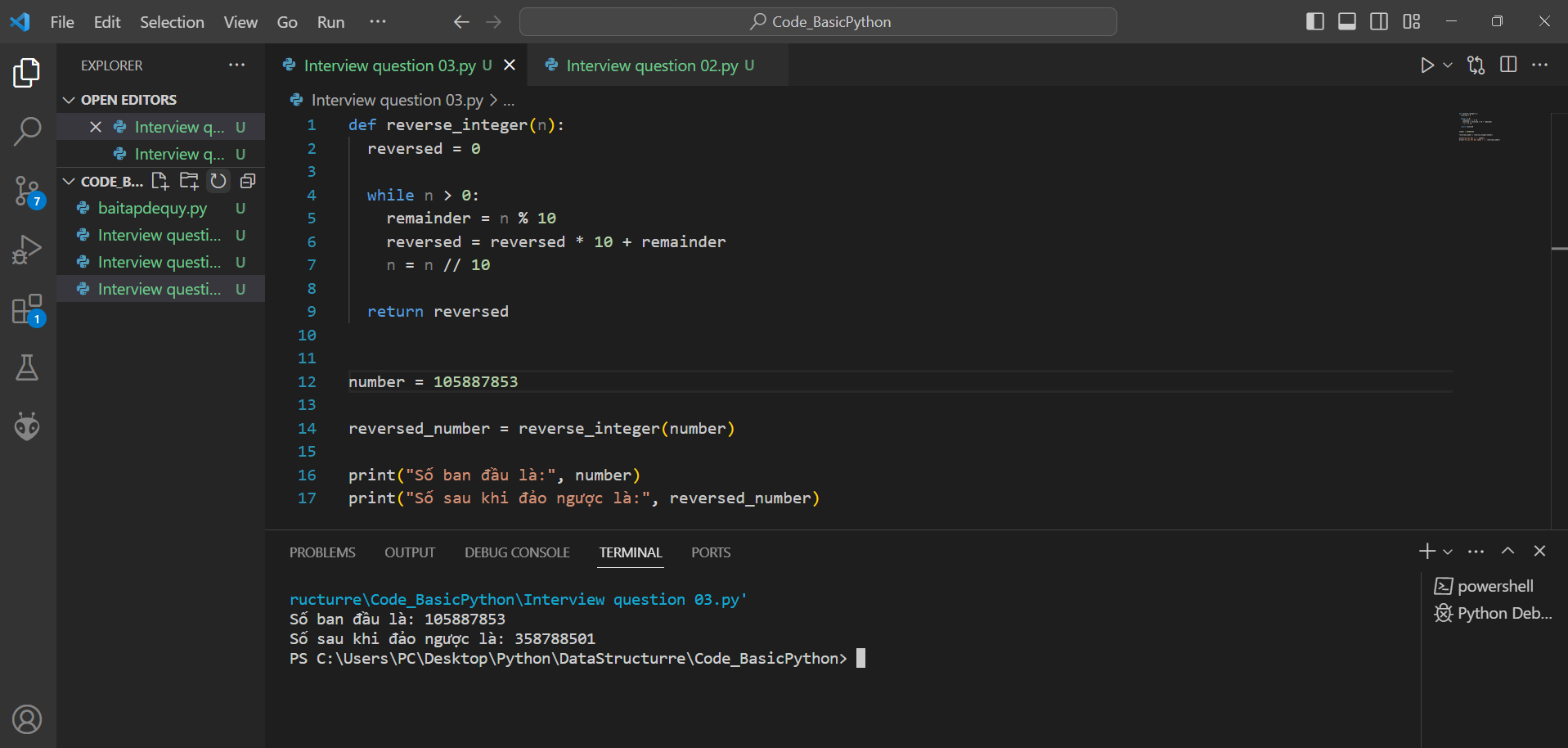
Interview question 02

https://github.com/Haoduc/HomeWork/blob/0b77c6fd6600e10055f9dd21ef73eab49f97650c/Interview%20question%2002.py



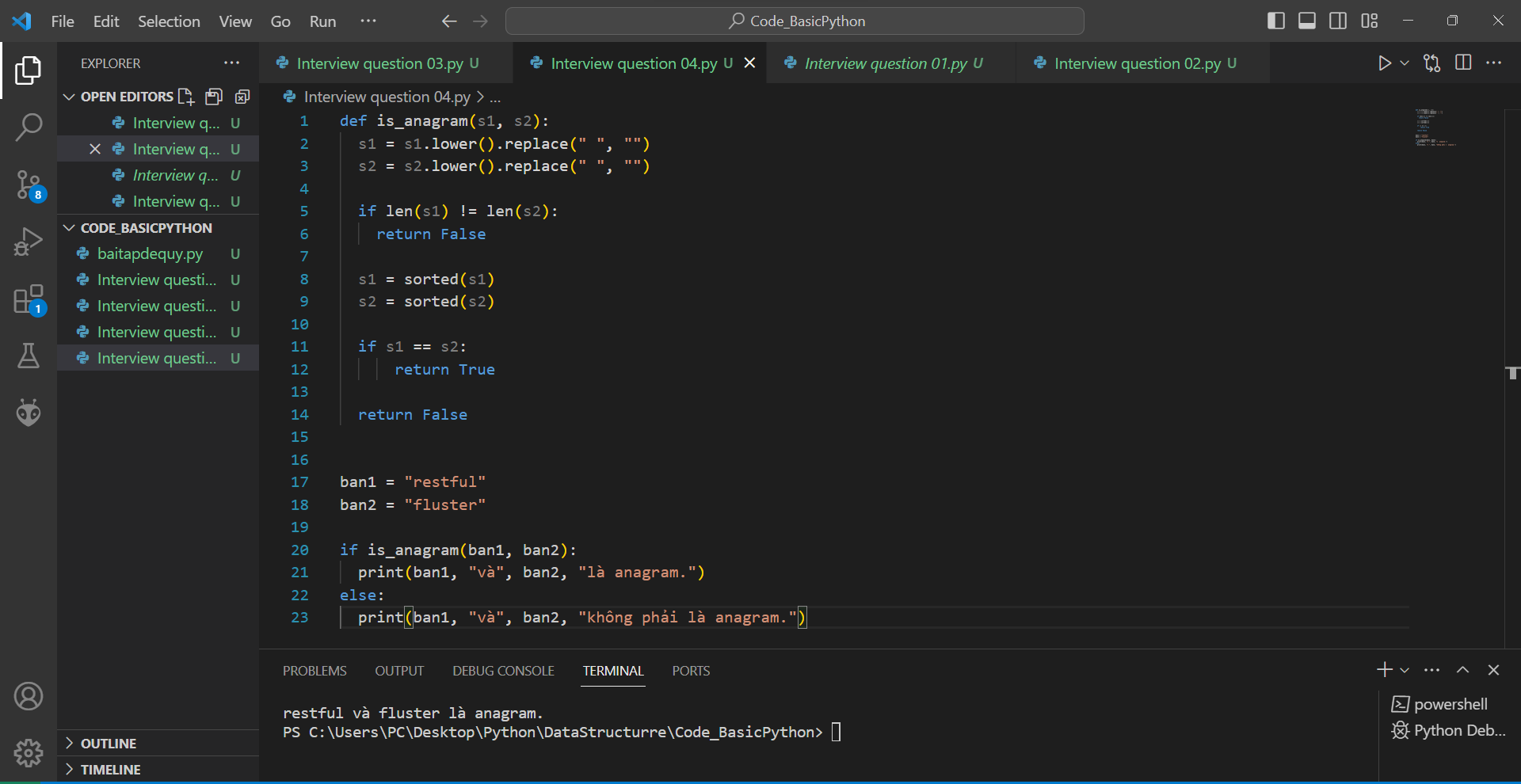
Interview question 03

https://github.com/Haoduc/HomeWork/blob/0b77c6fd6600e10055f9dd21ef73eab49f97650c/Interview%20question%2003.py



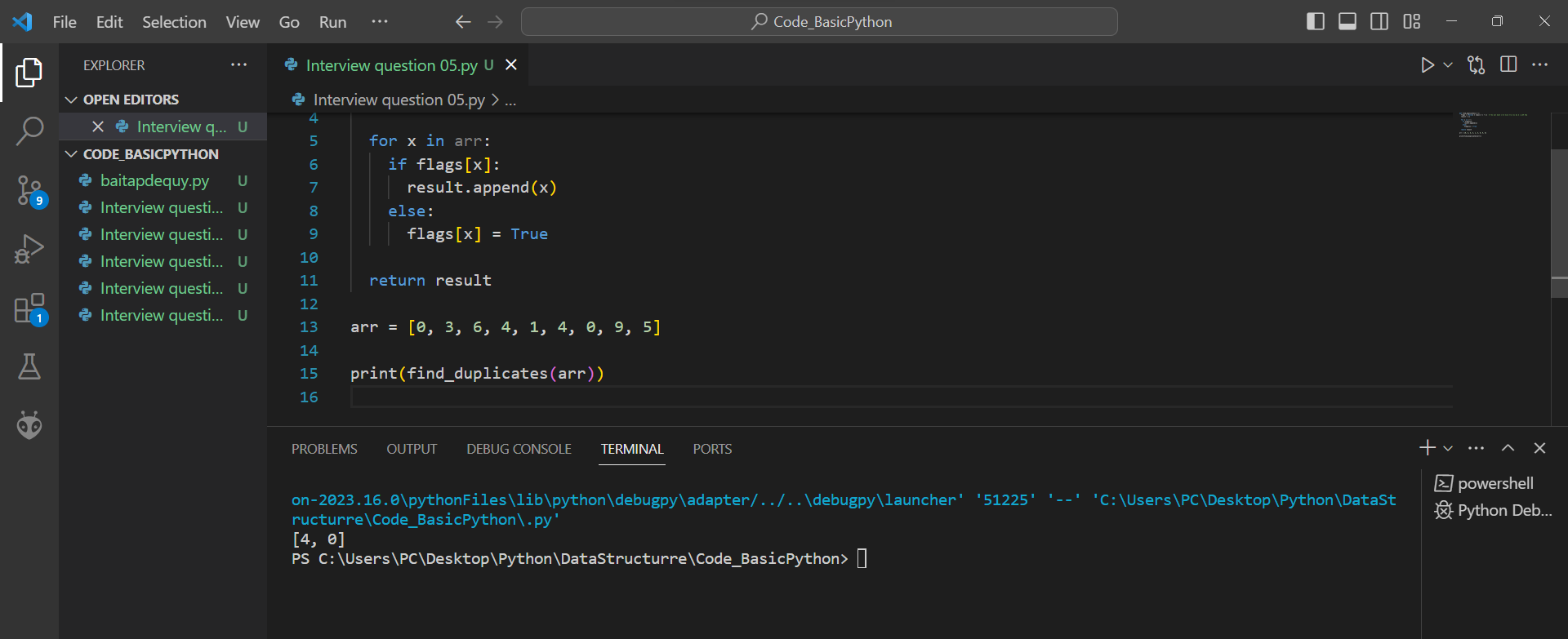
Interview question 04

https://github.com/Haoduc/HomeWork/blob/0b77c6fd6600e10055f9dd21ef73eab49f97650c/Interview%20question%2004.py



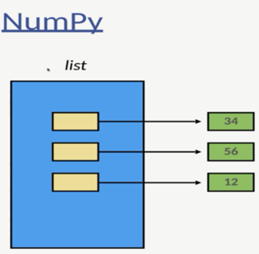
Interview question 05

https://github.com/Haoduc/HomeWork/blob/0b77c6fd6600e10055f9dd21ef73eab49f97650c/Interview%20question%2005.py



What are Lists in Python? (Python Programming)

NumPython:  
Trong Python không có kiểu dữ liệu mảng thực sự vì mọi thứ đều là đối tượng. NumPy là một thư viện được dùng để xử lý mảng. Và chúng có thể đạt được tốc độ xử lý nhanh hơn 50 lần.



Trong Python mọi thứ đều là đối tượng, danh sách lưu tham chiếu tới các đối tượng. Mỗi tham chiếu có kích thước 8 bytes nên lưu nhiều phần tử trong list tốn nhiều bộ nhớ. Mảng NumPy lưu trữ liên tục trong bộ nhớ, các phần tử nằm sát nhau.

Linked List Data Structure (Algorithms and Data Structures)  
  
Linked List:  
  
Linked list là cấu trúc dữ liệu cho phép lưu trữ các phần tử một cách hiệu quả (các thao tác chèn, xóa).

Mảng có nhược điểm là có thể có "lỗ trống" và phải dịch chuyển nhiều phần tử.

Vấn đề này có thể được khắc phục bằng linked list.

Mỗi node trong linked list lưu trữ dữ liệu và liên kết tới node kế tiếp.

Do đó linked list cần nhiều bộ nhớ hơn mảng.

Nhưng linked list không bị "lỗ trống" nên không cần dịch chuyển phần tử.

* Ưu điểm lớn: chèn phần tử vào đầu danh sách rất nhanh O(1).
* Nhược điểm: chèn vào cuối danh sách chậm O(N).

Linked Lists Advantages:  
  
Là cấu trúc dữ liệu động, có thể cấp phát bộ nhớ thông qua các chèn các node mới.

Không cần phải resize như mảng.

Có thể mở rộng dần dần, không vấn đề nếu không biết kích thước từ đầu.

Thao tác đầu danh sách nhanh O(1).

Có thể lưu các phần tử có kích thước khác nhau.  
  
Linked Lists Disadvantages:  
  
Cần nhiều bộ nhớ hơn do các con trỏ.

Không truy xuất ngẫu nhiên, chỉ truy xuất được node đầu tiên.