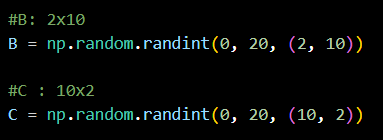


Đoạn code này dùng hàm np.random.randint() có trong thư viện numpy để tạo một ma trận 10x10, với các phần tử trong ma trận được lấy ngẫu nhiên từ 1 đến 100 và gán nó vào biến A.

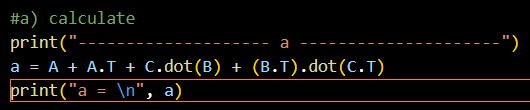
Các tham số trong hàm np.random.randint() lần lượt là:

* **0**: giới hạn dưới của các phần tử số nguyên ngẫu nhiên được tạo ra.
* **101**: giới hạn trên (ngoại trừ 101) của các phần tử số nguyên ngẫu nhiên được tạo ra.
* **(10, 10)**: kích thước của ma trận 10x10 được tạo ra.



Cách tạo ma trận B và C cũng giống như ma trận A ở phía trên.

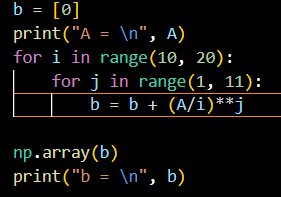
Câu a:



Đoạn code này dùng để tính toán các phép toán cộng và nhân ma trận. Cụ thể, ma trận A cộng với ma trận chuyển vị của A, được biểu thị bởi A.T ( .T là một phương thức để biểu thị ma trận chuyển vị trong thư viện Numpy ). Sau đó cộng tổng của tích ma trận C và ma trận B, được biểu thị bởi C.dot(B) ( numpy.dot() là phương thức được dùng để nhân hai ma trận với nhau trong thư viện numpy ), với tích của ma trận chuyển vị của B với ma trận chuyển vị của C, được biểu thị bởi (B.T).dot(C.T). Sau cùng, kết quả sẽ được lưu vào biến a.

Đoạn code thứ hai dùng để in ra màn hình giá trị của biến a.

Câu b:



Đầu tiên khai báo một list rỗng b

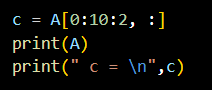
Sau đó sử dụng 2 vòng lặp để tính toán biểu thức mà đề bài cho theo công thức: . Vòng lặp đầu tiên lặp qua các giá trị từ 10 đến 19, và vòng lặp thứ hai lặp qua các giá trị từ 1 đến 10.

Sau mỗi lần lặp, chương trình sẽ tính toán giá trị của biểu thức (A/i)\*\*j (với I và j là các biến lặp). Sau đó, kết quả của biểu thức sẽ được cộng vào list b.

Sau khi hoàn thành vòng lặp, list b sẽ chứa giá trị đã tính toán. Sau đó chương trình sử dụng phương thức np.array() trong thư viện để chuyển đổi list b thành mảng numpy

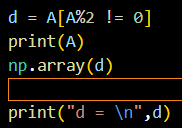
Sau cùng là in kết quả của của b lên màn hình.

Câu c:



* Đoạn code này dùng để lưu những dòng lẻ cảu ma trận A vào một ma trận mới.
* Đoạn code “**c = A[0:10:2, :]”** dùng để lấy các hàng lẻ của ma trận A ra và gán vào ma trận c.
* Cú pháp lát cắt **“0:10:2”** có nghĩa là "bắt đầu từ vị trí thứ 0, kết thúc tại vị trí thứ 9 và có bước nhảy là 2 trong đoạn [0, 9], Ký hiệu **“ : ”** sau dấu phẩy có nghĩa là "bao gồm tất cả các cột".
* Sau cùng là in giá trị của c ra màn hình

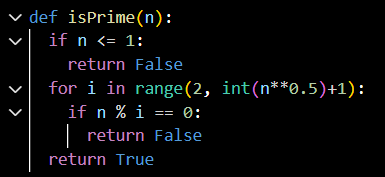
Câu d:



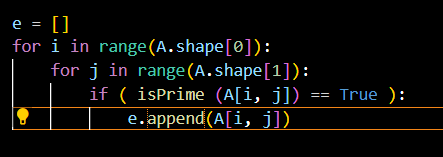
Đoạn code này dùng để lấy tất cả các phần tử trong mảng “A” có giá trị là số lẻ và lưu trữ chúng trong mảng “d”. Sau đó chuyển mảng d thành vector bằng phương thức có trong thư viện numpy “np.array()”. Cuối cùng là in vector d ra màn hình.

Câu e: lưu các số nguyên tố trong ma trận A vào một vector mới:

* Định nghĩa một hàm kiểm tra số nguyên tố isPrime() có tham số là một số nguyên:



* Số nguyên tố là số chỉ có hai ước là 1 và chính nó. Cho nên, đoạn code này hoạt động bằng cách:
* Đầu tiên, so sánh số nguyên cần được kiểm tra với 1, nếu số đó nhỏ hơn số 1 thì ta trả về “False”.
* Sau đó, ta chạy vòng lặp for với biến i nằm trong khoảng từ 2 đến làm tròn chữ số nguyên của . Nếu như số cần kiểm tra (n) chia hết cho i, thì ta trả về “False”.
* Khi chạy xong vòng lặp, cuối cùng thì ta trả về “True”.



* khai báo list rỗng e.
* Sau đó, Sử dụng vòng lặp để duyệt qua từng phần tử của ma trận A. Cụ thể, với mỗi giá trị của i từ 0 đến A.shape[0] (số dòng của ma trận), và với mỗi giá trị của j từ 0 đến A.shape[1] (số cột của ma trận), điều kiện lọc “isPrime (A[i, j]) == True” kiểm tra xem giá trị phần tử đó có là số nguyên tố không. Nếu có, giá trị đó được thêm vào list e bằng phương thức “append()”.
* Kết quả cuối cùng của đoạn code này là danh sách e chứa tất cả các giá trị là số nguyên tố trong ma trận A.
* Sau cùng, dùng phương thức “np.array()” để chuyển list e thành mảng numpy và in ra màn hình bằng câu lệnh “print()”.

Câu f:

* Để có được ma trận D, nhân hai ma trận C với B với nhau thông qua phương thức ( C.dot(B) – đây là hàm nhân ma trận có trong thư viện numpy )
* Tiếp đến, khai báo một list rỗng “D\_rev”
* Sau đó, dùng vòng lặp để duyệt qua từng hàng của D. Cụ thể, với mỗi giá trị của i từ 0 đến D.shape[0] (số dòng của ma trận). Nếu “i == 0” hoặc “i % 2 == 0” (với i là số chẳn) thì đảo ngược các phần tử của dòng đó bằng câu lệnh “np.flip()”, sau đó dùng câu lệnh “append” để cho dòng đã được đảo ngược vào trong list rỗng “D\_rev”. Ngược lại, nếu i không phải là số chẳn thì ta dùng câu lệnh “append” để cho dòng không đảo ngược vào list rỗng “D\_rev”.
* Tiếp đến, sử dụng phương thức “np.array()” để biến list “D\_rev” trở thành mảng numpy.
* Cuối cùng, in giá trị của mảng numpy “D\_rev” ra màn hình.

Câu g:

* Khai báo biến “maxCount” để giữ giá trị số lượng số nguyên tố lớn nhất của mảng
* Khai báo một list rỗng “resultant” để lưu các dòng có nhiều số nguyên tố nhất
* Tiếp đến, duyệt qua từng hàng của ma trận A.
* Khởi tạo giá trị cho biến count bằng 0
* Duyệt qua từng phần tử trong hàng đó
* Kiểm tra xem phần tử đó có phải là số nguyên tố không bằng lời gọi hàm “isPrime(num)” đã được định nghĩa ở câu 1e, nếu phần tử đó là số nguyên tố thì tăng giá trị của “count” lên 1.
* Sau khi duyệt qua các phần tử của dòng đó, biến count sẽ lưu giữ giá trị số lượng số nguyên tố trong dòng đó.
* Nếu “count” lớn hơn giá trị của “maxCount”, thì gán giá trị của “maxCount” bằng “count” và gán giá trị của “resultant” bằng mảng chứa “row”.
* Nếu “count” bằng giá trị của “maxCount”, thì thêm “row” vào mảng “resultant” bằng phương thức “append”.
* Sau cùng, in giá trị của Resultant ra màn hình.
* Tiếp đến, sử dụng phương thức “np.array()” để biến list “resultant” trở thành mảng numpy
* Cuối cùng, sử dụng vòng lặp để in từng dòng của mảng numpy “resultant” ra màn hình

Câu h:

* Khai báo biến “maxCount” để lưu giữ giá trị số lượng các số lẻ nguyên liên tiếp lớn nhất.
* Khai báo một list rỗng “results” để lưu giữ các dòng các số lẻ nguyên liên tiếp nhiều nhất.
* Tiếp đến, sử dụng vòng lặp duyệt qua từng hàng của ma trận A.
* Khởi tạo giá trị cho biến count bằng 0
* Tiếp đến, sử dụng vòng lặp duyệt qua từng cột của hàng hiện tại của ma trận A.
* Nếu duyệt đến phần tử cuối cùng của dòng thì bỏ qua
* Nếu hai phần tử liền kề đều là số lẻ thì tăng giá trị của “count” lên 1.
* Nếu “count” lớn hơn giá trị của “maxCount”, thì gán giá trị của “maxCount” bằng “count” và gán giá trị của “resultant” bằng mảng chứa “A[i]”.
* Nếu “count” bằng giá trị của “maxCount”, thì thêm “A[i]” vào mảng “results” bằng phương thức “append”.
* Sau cùng, in giá trị của Results ra màn hình.
* Tiếp đến, sử dụng phương thức “np.array()” để biến list “results” trở thành mảng numpy
* Cuối cùng, sử dụng vòng lặp để in từng dòng của mảng numpy “results” ra màn hình