TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

A red and blue logo

Description automatically generated

**BÀI TẬP ĐÁNH GIÁ QUÁ TRÌNH 2 MÔN XÁC SUẤT VÀ THỐNG KÊ ỨNG DỤNG CHO CÔNG NGHỆ THÔNG TIN**

*Người hướng dẫn*: GV NGUYỄN LÂM

*Người thực hiện*: **NGUYỄN ĐÌNH VIỆT HOÀNG – 522H0120**

Lớp **: 22H50302**

Khoá  **: 26**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

A red and blue logo

Description automatically generated

**BÀI TẬP ĐÁNH GIÁ QUÁ TRÌNH 2 MÔN XÁC SUẤT VÀ THỐNG KÊ ỨNG DỤNG CHO CÔNG NGHỆ THÔNG TIN**

*Người hướng dẫn*: GV NGUYỄN LÂM

*Người thực hiện*: **NGUYỄN ĐÌNH VIỆT HOÀNG – 522H0120**

Lớp **: 22H50302**

Khoá  **: 26**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

LỜI CẢM ƠN

Kính gửi thầy Nguyễn Lâm cùng toàn thể quý thầy cô nhà trường, trong không gian tri thức và sự hướng dẫn của thầy và nhà trường, tôi đã có cơ hội tham gia vào một chặng đường đầy ý nghĩa và phát triển - quá trình đánh giá quá trình 2 của tôi. Nhân dịp quá trình này đi đến bước kết thúc, tôi xin gửi lời cảm ơn sâu sắc nhất đến thầy và toàn thể quý thầy cô trong nhà trường. Trước tiên, tôi muốn bày tỏ lòng biết ơn vô hạn đến thầy Nguyễn Lâm. Sự tận tâm, kiên nhẫn và sự truyền cảm hứng mà thầy đã dành cho tôi không thể đong đếm được. Thầy không chỉ là người truyền đạt kiến thức mà còn là người hướng dẫn và đồng hành trên con đường phát triển cá nhân của tôi. Thầy đã dành nhiều thời gian và công sức để giúp tôi hiểu rõ hơn về quá trình 2 và đồng thời hỗ trợ tôi trong việc nắm vững kiến thức và kỹ năng cần thiết để có thể hoàn thành bài tập đánh giá quá trình 2 một cách tốt nhất. Tôi cũng muốn gửi lời cảm ơn đến gia đình và những người thân yêu của mình. Sự hỗ trợ, khích lệ và động viên không ngừng từ phía họ đã là nguồn lực quan trọng giúp tôi vượt qua những khó khăn trong quá trình nghiên cứu và thực hiện quá trình 2. Đối với những người thân yêu của tôi, quá trình này không chỉ là của riêng tôi mà còn là của cả gia đình, và tôi rất biết ơn sự ủng hộ và tin tưởng của họ.

**BÀI TẬP ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm bài tập của riêng tôi và được sự hướng dẫn của GV Nguyễn Lâm. Các nội dung nghiên cứu, kết quả trong bài tập này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung bài tập của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 1 tháng 8 năm 2023*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Hoàng*

*Nguyễn Đình Việt Hoàng*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

* Đầu tiên tôi sẽ xem tổng thể yêu cầu đề bài và đặt mục tiêu cho từng chương để hoàn thành xuất sắc chương đấy. Với chương 1,2 và 3, tôi sẽ tra google về từng hàm của mỗi chương có chức năng gì, định nghĩa hàm là gì và xem đoạn mã ví dụ trên các trang web có sẵn để từ đó có thể tự viết cho mình một đoạn code của riêng mình và chạy thử xem có lỗi gì hay không để có thể debug.
* Trong thống kê, việc tính toán các giá trị trung bình và đo lường vị trí trung tâm của tập dữ liệu là rất quan trọng để hiểu và phân tích dữ liệu. Thư viện ‘**statistics**’ trong Python cung cấp nhiều hàm hữu ích để giúp chúng ta thực hiện các phép tính này một cách dễ dàng và chính xác.
* Thư viện ‘**statistics**’ chứa nhiều hàm để tính toán các giá trị thống kê như trung bình cộng, trung vị, trung bình hình học, độ lệch chuẩn và các hàm đo lường mối quan hệ giữa hai tập dữ liệu.

MỤC LỤC

LỜI CẢM ƠN…………………………………………………………………………..3

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN…………………………...5

TÓM TẮT………………………………………………………………………………6

MỤC LỤC……………………………………………………………………………....7

DANH MỤC CÁC HÌNH ẢNH………………………………………………………………..8

CHƯƠNG 1 - CÁC GIÁ TRỊ TRUNG BÌNH VÀ ĐO LƯỜNG VỊ TRÍ TRUNG TÂM……10

1.1 Tổng quan về các hàm…………………………………………………………….10

1.2 Mỗi hàm cần được giải thích về cách sử dụng, tham số đầu vào, ý nghĩa của kết quả trả về và mã ví dụ……………………………………………………...12

1.3 Đưa ra mã ví dụ của riêng tôi. Chụp ảnh màn hình cho hoạt động của chúng…...26

CHƯƠNG 2 - ĐO LƯỜNG SỰ LAN TRUYỀN (SỰ PHÂN TÁN)…………………36

2.1 Tổng quan về các hàm……………………………………………………...36

2.2 Mỗi hàm cần được giải thích về cách sử dụng, tham số đầu vào, ý nghĩa của kết quả trả về và mã ví dụ……………………………………………………....................37

2.3 Đưa ra mã ví dụ của riêng bạn. Chụp ảnh màn hình cho hoạt động của chúng…...43

CHƯƠNG 3 - THỐNG KÊ VỀ MỐI QUAN HỆ GIỮA HAI ĐẦU VÀO…………………...48

3.1 Tổng quan về các hàm……………………………………………………...48

3.2 Mỗi hàm cần được giải thích về cách sử dụng, tham số đầu vào, ý nghĩa của kết quả trả về và mã ví dụ……………………………………………………....................48

3.3 Đưa ra mã ví dụ của riêng bạn. Chụp ảnh màn hình cho hoạt động của chúng…...53

DANH MỤC CÁC HÌNH ẢNH

**DANH MỤC HÌNH**

Hình 1.3.1.a. Ảnh chụp một đoạn code ví dụ của tôi…………………………...25

Hình 1.3.1.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.1.a……………..25

Hình 1.3.2.a. Ảnh chụp một đoạn code ví dụ của tôi…………………………...26

Hình 1.3.2.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.2.a……………..26

Hình 1.3.3.a. Ảnh chụp một đoạn code ví dụ của tôi…………………………...27

Hình 1.3.3.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.3.a……………..27

Hình 1.3.4.a. Ảnh chụp một đoạn code ví dụ của tôi…………………………...28

Hình 1.3.4.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.4.a……………..28

Hình 1.3.5.a. Ảnh chụp một đoạn code ví dụ của tôi…………………………...29

Hình 1.3.5.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.5.a……………..29

Hình 1.3.6.a. Ảnh chụp một đoạn code ví dụ của tôi…………………………...29

Hình 1.3.6.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.6.a……………..30

Hình 1.3.7.a. Ảnh chụp một đoạn code ví dụ của tôi…………………………...30

Hình 1.3.7.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.7.a……………..30

Hình 1.3.8.a. Ảnh chụp một đoạn code ví dụ của tôi…………………………...31

Hình 1.3.8.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.8.a……………..31

Hình 1.3.9.a. Ảnh chụp một đoạn code ví dụ của tôi…………………………...32

Hình 1.3.9.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.9.a……………..32

Hình 1.3.10.a. Ảnh chụp một đoạn code ví dụ của tôi………………………….32

Hình 1.3.10.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.10.a…………..32

Hình 1.3.11.a. Ảnh chụp một đoạn code ví dụ của tôi………………………….33

Hình 1.3.11.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.11.a…………..33

Hình 2.3.1.a. Ảnh chụp một đoạn code ví dụ của tôi…………………………...44

Hình 2.3.1.b. Ảnh chụp kết quả của đoạn code trong hình 2.3.1.a……………..44

Hình 2.3.2.a. Ảnh chụp một đoạn code ví dụ của tôi…………………………...45

Hình 2.3.2.b. Ảnh chụp kết quả của đoạn code trong hình 2.3.2.a……………..45

Hình 2.3.3.a. Ảnh chụp một đoạn code ví dụ của tôi…………………………...46

Hình 2.3.3.b. Ảnh chụp kết quả của đoạn code trong hình 2.3.3.a……………..46

Hình 2.3.4.a. Ảnh chụp một đoạn code ví dụ của tôi…………………………...47

Hình 2.3.4.b. Ảnh chụp kết quả của đoạn code trong hình 2.3.4.a……………..47

Hình 3.3.1.a. Ảnh chụp một đoạn code ví dụ của tôi…………………………...54

Hình 3.3.1.b. Ảnh chụp kết quả của đoạn code trong hình 3.3.1.a……………..54

Hình 3.3.2.a. Ảnh chụp một đoạn code ví dụ của tôi…………………………...55

Hình 3.3.2.b. Ảnh chụp kết quả của đoạn code trong hình 3.3.2.a……………..55

Hình 3.3.3.a. Ảnh chụp một đoạn code ví dụ của tôi…………………………...56

Hình 3.3.3.b. Ảnh chụp kết quả của đoạn code trong hình 3.3.3.a……………..56

CHƯƠNG 1 – CÁC GIÁ TRỊ TRUNG BÌNH VÀ ĐO LƯỜNG VỊ TRÍ TRUNG TÂM

* 1. Tổng quan về các hàm:

1. *statistics.mean(data):* Hàm statistics.mean(data) là một hàm trong thư viện statistics của Python, được sử dụng để tính giá trị trung bình (mean) của dữ liệu. Giá trị trung bình là tổng của tất cả các giá trị trong dữ liệu chia cho số lượng các giá trị trong dữ liệu.
2. *statistics.fmean(data, weights=None):* Hàm statistics.fmean(data, weights=None) là một hàm tính toán thống kê trong Python, được sử dụng để tính giá trị trung bình có trọng số của một tập dữ liệu số. Ngoài ra, nó còn có thể chuyển đổi dữ liệu thành số thực (float) và tính toán trung bình số học (the algorithm mean). Điều này chạy nhanh hơn hàm ‘mean()’ của thư viện statistics và nó luôn trả về kiểu float.
3. *statistics.geometric\_mean(data):* Hàm statistics.geometric\_mean(data) là một hàm tính toán thống kê trong Python, được sử dụng để tính trung bình hình học của một tập dữ liệu số.
4. *statistics.harmonic\_mean(data, weights=None):* Hàm ‘statistics.harmonic\_mean()’ trong thư viện ‘statistics’ của Python được sử dụng để tính giá trị trung bình nghịch đảo của một tập dữ liệu. Giá trị trung bình nghịch đảo là nghịch đảo của trung bình các nghịch đảo của từng giá trị trong tập dữ liệu.
5. *statistics.median(data):* Hàm statistics.median() trong thư viện statistics của Python được sử dụng để tính giá trị trung vị của một tập dữ liệu. Trung vị là giá trị ở vị trí giữa khi các giá trị trong tập dữ liệu được sắp xếp theo thứ tự tăng dần hoặc giảm dần.
6. *statistics.median\_low(data):* Hàm ‘statistics.median\_low()’ trong thư viện ‘statistics’ của Python được sử dụng để tính giá trị trung vị của một tập dữ liệu. Trung vị là giá trị ở vị trí giữa khi các giá trị trong tập dữ liệu được sắp xếp theo thứ tự tăng dần hoặc giảm dần. Tuy nhiên, hàm này luôn chọn giá trị ở vị trí thấp hơn trong trường hợp số lượng phần tử là số lẻ.
7. *statistics.median\_high(data):* Hàm ‘statistics.median\_high()’ trong thư viện statistics của Python được sử dụng để tính giá trị trung vị của một tập dữ liệu. Trung vị là giá trị ở vị trí giữa khi các giá trị trong tập dữ liệu được sắp xếp theo thứ tự tăng dần hoặc giảm dần. Hàm này luôn chọn giá trị ở vị trí cao hơn trong trường hợp số lượng phần tử là số lẻ.
8. *statistics.median\_grouped(data, interval=1):* Hàm statistics.median\_grouped() trong thư viện statistics của Python được sử dụng để tính giá trị trung vị của một tập dữ liệu được nhóm lại vào các khoảng (interval).
9. *statistics.mode(data):* Hàm ‘statistics.mode()’ trong thư viện ‘statistics’ của Python được sử dụng để tính giá trị mode của một tập dữ liệu. Mode là giá trị xuất hiện nhiều nhất trong tập dữ liệu, tức là giá trị có tần suất xuất hiện cao nhất.
10. *statistics.multimode(data):* Hàm ‘statistics.multimode()’ trong thư viện ‘statistics’ của Python được sử dụng để tìm các giá trị mode của một tập dữ liệu. Mode là giá trị xuất hiện nhiều nhất trong tập dữ liệu, tức là giá trị có tần suất xuất hiện cao nhất. Trong trường hợp có nhiều giá trị có cùng tần suất xuất hiện cao nhất, hàm này sẽ trả về tất cả các giá trị mode đó.
11. *statistics.quantiles(data, \*, n=4, method='exclusive'):* Hàm ‘statistics.quantiles()’ trong thư viện ‘statistics’ của Python được sử dụng để tính các phân vị của một tập dữ liệu. Phân vị là các giá trị chia tập dữ liệu thành các phần bằng nhau (hoặc gần như bằng nhau). Có nhiều cách tính phân vị và ‘statistics.quantiles()’ cung cấp hai phương pháp tính phân vị, bao gồm phân vị "exclusive" (mặc định) và phân vị "inclusive".
    1. Mỗi hàm cần được giải thích về cách sử dụng, tham số đầu vào, ý nghĩa của kết quả trả về và mã ví dụ:
12. *statistics.mean(data):*

* Cách sử dụng:

+ Chúng ta cần import thư viện ‘statistics’ trước khi sử dụng hàm. Sau đó, gọi hàm ‘statistics.mean(data)’ để tính giá trị trung bình của ‘data’.

* Tham số đầu vào:

+ Tham số đầu vào của hàm là ‘data’ – là một list hoặc sequence chứa các giá trị số học.

* Ý nghĩa của kết quả trả về:

+ Trả về giá trị trung bình số học mẫu của dữ liệu có thể là một chuỗi hoặc có thể lặp lại.

+ Trung bình số học là tổng dữ liệu chia cho số điểm dữ liệu. Nó thường được gọi là "trung bình", mặc dù nó chỉ là một trong nhiều trung bình toán học khác nhau. Nó là thước đo vị trí trung tâm của dữ liệu.

+ Nếu dữ liệu trống, số lỗi thống kê (StatisticsError) sẽ được tăng lên.

* Mã ví dụ:

import statistics

data = [10, 20, 30, 40, 50]

mean\_value = statistics.mean(data)

print(mean\_value)

+ Trong ví dụ trên, dữ liệu ‘data’ có các giá trị là [10, 20, 30, 40, 50]. Hàm ‘statistics.mean(data)’ tính giá trị trung bình của dữ liệu này và trả về kết quả là 30.

* Lưu ý: Nếu dữ liệu là một list trống hoặc không chứa bất kỳ giá trị nào, hàm ‘mean()’ sẽ ném ra một lỗi ‘statistics.StatisticsError’. Do đó, trước khi tính giá trị trung bình, chúng ta nên kiểm tra xem dữ liệu có tồn tại và không rỗng hay không.

1. *statistics.fmean(data, weights=None):*

* Cách sử dụng:

+ Đầu tiên, chúng ta cần nhập module statistics bằng cách sử dụng lệnh ‘import statistics’.

+ Sau đó, chúng ta có thể gọi hàm ‘statistics.fmean(data, weights=None)’ để tính giá trị trung bình có trọng số của tập dữ liệu ‘data’ với trọng số tương ứng trong ‘weights’.

* Tham số đầu vào:

+ ‘data’: Là một tập hợp các giá trị số mà bạn muốn tính giá trị trung bình có trọng số. Đây có thể là một danh sách, tuple hoặc tập hợp các số.

+ ‘weights’ (tùy chọn): Là một tập hợp các số có cùng độ dài với data, đại diện cho trọng số tương ứng của các giá trị trong data. Mặc định là None, trong trường hợp không có trọng số nào được cung cấp và tất cả các giá trị đều có trọng số bằng nhau.

* Ý nghĩa của kết quả trả về:

+ Hàm ‘statistics.fmean(data, weights=None)’ giúp chuyển đổi thành số thực (float) và trả về giá trị trung bình có trọng số của tập dữ liệu ‘data’.

+ Dữ liệu có thể là một chuỗi hoặc có thể lặp lại. Nếu tập dữ liệu đầu vào trống, sẽ tăng số lỗi thống kê (StatisticsError). Trọng số tùy chọn được hỗ trợ. Nếu trọng số được cung cấp, nó phải có cùng độ dài với dữ liệu hoặc số giá trị bị lỗi (ValueError) sẽ được tăng lên - mới có trong Python 3.8.

+ Thay đổi trong Python 3.11: Đã thêm hỗ trợ cho ‘weights’.

* Mã ví dụ:

import statistics

# Danh sách các số cần tính giá trị trung bình có trọng số

data\_list = [10, 20, 30, 40, 50]

# Trọng số tương ứng với các giá trị trong data\_list

weights\_list = [1, 2, 3, 4, 5]

# Tính giá trị trung bình có trọng số của tập dữ liệu

weighted\_mean = statistics.fmean(data\_list, weights=weights\_list)

# In kết quả

print("Dữ liệu:", data\_list)

print("Trọng số:", weights\_list)

print("Giá trị trung bình có trọng số:", weighted\_mean)

* Lưu ý: Khi tính toán giá trị trung bình có trọng số, cần đảm bảo tổng các trọng số là khác 0, nếu không sẽ xảy ra lỗi ‘ZeroDivisionError’.

1. *statistics.geometric\_mean(data):*

* Cách sử dụng:

+ Đầu tiên, chúng ta cần nhập module statistics bằng cách sử dụng lệnh ‘import statistics’. Sau đó, chúng ta có thể gọi hàm ‘statistics.geometric\_mean(data)’ để tính trung bình hình học của tập dữ liệu ‘data’.

* Tham số đầu vào:

+ ‘data’: Là một tập hợp các giá trị số mà bạn muốn tính trung bình hình học. Đây có thể là một danh sách, tuple hoặc tập hợp các số.

* Ý nghĩa của kết quả trả về:

+ Chuyển đổi dữ liệu thành số thực (float).

+ Hàm ‘statistics.geometric\_mean(data)’ trả về giá trị trung bình hình học của tập dữ liệu ‘data’.

+ Giá trị trung bình hình học chỉ ra xu hướng trung tâm hoặc giá trị điển hình của dữ liệu bằng cách sử dụng tích của các giá trị (trái ngược với trung bình số học sử dụng tổng của chúng).

+ Tăng số lỗi thống kê (StatisticsError) nếu tập dữ liệu đầu vào trống, nếu nó chứa số không hoặc nếu nó chứa giá trị âm. Dữ liệu có thể là một chuỗi hoặc có thể lặp lại.

+ Không có cố gắng đặc biệt nào được thực hiện để đạt được kết quả chính xác. (Tuy nhiên, điều này có thể thay đổi trong tương lai.)

+ Thay đổi trong Python 3.10: Đã thêm hỗ trợ cho ‘weights’.

* Mã ví dụ:

import statistics

# Danh sách các số cần tính trung bình hình học

data\_list = [1, 2, 3, 4, 5]

# Tính trung bình hình học của tập dữ liệu

geometric\_mean\_value = statistics.geometric\_mean(data\_list)

# In kết quả

print("Dữ liệu:", data\_list)

print("Trung bình hình học:", geometric\_mean\_value)

1. *statistics.harmonic\_mean(data, weights=None):*

* Cách sử dụng:

+ Đầu tiên, chúng ta cần nhập module statistics bằng cách sử dụng lệnh ‘import statistics’. Sau đó, chúng ta có thể gọi hàm ‘statistics.harmonic\_mean’ (data) để tính trung bình nghịch đảo của tập dữ liệu ‘data’.

* Tham số đầu vào:

+ Hàm statistics.harmonic\_mean(data, weights=None) có hai tham số:

* ‘data’: Đây là một tập hợp các giá trị số, đại diện cho dữ liệu bạn muốn tính giá trị trung bình nghịch đảo.
* ‘weights’ (tùy chọn): Đây là một dãy các trọng số tương ứng với từng giá trị trong ‘data’. Nếu không cung cấp tham số này, tất cả các giá trị trong ‘data’ sẽ được coi như có trọng số bằng nhau.
* Ý nghĩa của kết quả trả về:

+ Kết quả trả về là giá trị trung bình nghịch đảo của tập dữ liệu ‘data’. Nó được tính bằng cách lấy tổng nghịch đảo của các giá trị trong ‘data’, sau đó chia cho số lượng giá trị trong ‘data’.

+ Số lỗi thống kê (StatisticsError) được tăng lên nếu ‘data’ trống, bất kỳ phần tử nào nhỏ hơn 0 hoặc nếu tổng ‘weights’ không dương.

+ Thuật toán hiện tại có một sớm ra khi nó gặp một số 0 trong đầu vào. Điều này có nghĩa là các đầu vào tiếp theo không được kiểm tra tính hợp lệ. (Hành vi này có thể thay đổi trong tương lai.)

+ Thay đổi trong phiên bản 3.10: Đã thêm hỗ trợ cho ‘weights’.

* Mã ví dụ:

import statistics

data\_list = [2, 4, 6, 8, 10]

# Tính giá trị trung bình nghịch đảo

harmonic\_mean\_value = statistics.harmonic\_mean(data\_list)

print("Dữ liệu:", data\_list)

print("Giá trị trung bình nghịch đảo:", harmonic\_mean\_value)

1. *statistics.median(data):*

* Cách sử dụng:

+ Đầu tiên, chúng ta cần nhập module statistics bằng cách sử dụng lệnh ‘import statistics’. Sau đó, chúng ta có thể gọi hàm ‘statistics.median(data)’ để tính giá trị trung vị của tập dữ liệu ‘data’.

* Tham số đầu vào:

+ Hàm ‘statistics.median(data)’ có một tham số:

* ‘data’: Đây là một tập hợp các giá trị số, đại diện cho dữ liệu bạn muốn tính giá trị trung vị.
* Ý nghĩa của kết quả trả về:

+ Kết quả trả về là giá trị trung vị của tập dữ liệu ‘data’. Nếu số lượng phần tử trong ‘data’ là lẻ, giá trị trung vị sẽ là giá trị ở vị trí giữa của dãy số sau khi sắp xếp. Nếu số lượng phần tử là chẵn, giá trị trung vị sẽ là trung bình cộng của hai giá trị ở vị trí giữa.

+ Nếu ‘data’ trống, số lỗi thống kê (StatisticsError) sẽ được tăng lên. ‘data’ có thể là một chuỗi hoặc có thể lặp lại. Trung vị là một thước đo mạnh mẽ của vị trí trung tâm và ít bị ảnh hưởng bởi sự hiện diện của các ngoại lệ.

* Mã ví dụ:

import statistics

data\_list = [15, 23, 12, 45, 10, 32]

# Tính giá trị trung vị

median\_value = statistics.median(data\_list)

# In kết quả

print("Dữ liệu:", data\_list)

print("Giá trị trung vị:", median\_value)

1. *statistics.median\_low(data):*

* Cách sử dụng:

+ Đầu tiên, chúng ta cần nhập module statistics bằng cách sử dụng lệnh ‘import statistics’. Sau đó, chúng ta có thể gọi hàm ‘statistics.median\_low(data)’ để tính giá trị trung vị ở vị trí thấp hơn trong trường hợp số lượng phần tử là số lẻ của tập dữ liệu ‘data’.

* Tham số đầu vào:

+ Hàm statistics.median\_low(data) có một tham số:

* ‘data’: Đây là một tập hợp các giá trị số, đại diện cho dữ liệu bạn muốn tính giá trị trung vị.
* Ý nghĩa của kết quả trả về:

+ Kết quả trả về là giá trị trung vị của tập dữ liệu ‘data’. Nếu số lượng phần tử trong ‘data’ là lẻ, hàm sẽ luôn chọn giá trị ở vị trí thấp hơn (nằm bên trái) trong dãy số sau khi sắp xếp. Nếu số lượng phần tử là chẵn, giá trị trung vị sẽ là trung bình cộng của hai giá trị ở vị trí giữa.

+ Nếu dữ liệu trống, số lỗi thống kê (StatisticsError) sẽ được tăng lên. ‘data’ có thể là một chuỗi hoặc có thể lặp lại.

+ Sử dụng trung vị thấp khi ‘data’ của bạn rời rạc và bạn thích trung vị là một điểm dữ liệu thực tế hơn là nội suy.

* Mã ví dụ:

import statistics

data\_list = [10, 15, 20, 25, 30]

# Tính giá trị trung vị (lấy giá trị thấp hơn nếu số lượng phần tử là lẻ)

median\_low\_value = statistics.median\_low(data\_list)

# In kết quả

print("Dữ liệu:", data\_list)

print("Giá trị trung vị (lấy giá trị thấp hơn):", median\_low\_value)

1. *statistics.median\_high(data):*

* Cách sử dụng:

+ Chúng ta cần import thư viện ‘statistics’ trước khi sử dụng hàm. Sau đó, gọi hàm ‘statistics.median\_high(data)’ để tính giá trị trung vị ở vị trí cao hơn trong trường hợp số lượng phần tử là số lẻ của tập dữ liệu ‘data’.

* Tham số đầu vào:

+ Hàm statistics.median\_high(data) có một tham số:

* ‘data’: Đây là một tập hợp các giá trị số, đại diện cho dữ liệu bạn muốn tính giá trị trung vị.
* Ý nghĩa của kết quả trả về:

+ Kết quả trả về là giá trị trung vị của tập dữ liệu data. Nếu số lượng phần tử trong data là lẻ, hàm sẽ luôn chọn giá trị ở vị trí cao hơn (nằm bên phải) trong dãy số sau khi sắp xếp. Nếu số lượng phần tử là chẵn, giá trị trung vị sẽ là trung bình cộng của hai giá trị ở vị trí giữa.

+ Nếu dữ liệu trống, số lỗi thống kê (StatisticsError) sẽ được tăng lên. ‘data’ có thể là một chuỗi hoặc có thể lặp lại.

+ Sử dụng trung vị cao khi dữ liệu của bạn rời rạc và bạn thích trung vị là một điểm dữ liệu thực tế hơn là nội suy.

* Mã ví dụ:

import statistics

data\_list = [5, 8, 12, 15, 20, 25]

# Tính giá trị trung vị (lấy giá trị cao hơn nếu số lượng phần tử là lẻ)

median\_high\_value = statistics.median\_high(data\_list)

# In kết quả

print("Dữ liệu:", data\_list)

print("Giá trị trung vị (lấy giá trị cao hơn):", median\_high\_value)

1. *statistics.median\_grouped(data, interval=1):*

* Cách sử dụng:

+ Chúng ta cần import thư viện ‘statistics’ trước khi sử dụng hàm. Sau đó, gọi hàm ‘statistics.median\_grouped(data, interval=1)’ để tính giá trị trung vị của một tập ‘data’ được nhóm lại vào các khoảng (interval).

* Tham số đầu vào:

+ Hàm statistics.median\_grouped(data, interval=1) có hai tham số:

* ‘data’: Đây là một tập hợp các giá trị số, đại diện cho dữ liệu bạn muốn tính giá trị trung vị.
* ‘interval’ (tùy chọn): Đây là giá trị số nguyên dương đại diện cho kích thước của từng khoảng. Mặc định là 1, tức là các giá trị được nhóm lại không thay đổi.
* Ý nghĩa của kết quả trả về:

+ Kết quả trả về là giá trị trung vị của tập dữ liệu data sau khi được nhóm lại vào các khoảng. Để tính giá trị trung vị này, hàm sử dụng công thức ‘median = L + (n/2 - F) \* h’, trong đó:

* ‘L’: Là giá trị trung vị của khoảng mà giá trị trung vị nằm trong.
* ‘n’: Là tổng số phần tử trong tập dữ liệu.
* ‘F’: Là số phần tử nằm trong các khoảng nhỏ hơn khoảng chứa giá trị trung vị.
* ‘h’: Là kích thước của từng khoảng (interval).

+ Nếu dữ liệu trống, số lỗi thống kê (StatisticsError) sẽ được tăng lên. Dữ liệu có thể là một chuỗi hoặc có thể lặp lại.

+ Trong một số trường hợp, ‘median\_grouped()’ có thể ép buộc các điểm dữ liệu thành phao. Hành vi này có thể sẽ thay đổi trong tương lai.

* Mã ví dụ:

import statistics

data\_list = [15, 20, 25, 30, 35, 40]

interval\_size = 5

# Tính giá trị trung vị của tập dữ liệu sau khi được nhóm lại vào các khoảng

median\_grouped\_value = statistics.median\_grouped(data\_list, interval=interval\_size)

# In kết quả

print("Dữ liệu:", data\_list)

print("Giá trị trung vị sau khi nhóm lại vào các khoảng:", median\_grouped\_value)

1. *statistics.mode(data):*

* Cách sử dụng:

+ Chúng ta cần import thư viện ‘statistics’ trước khi sử dụng hàm. Sau đó, gọi hàm ‘statistics.mode(data)’ để tính giá trị ‘mode’ của một tập dữ liệu ‘data’ – ‘mode’ là giá trị xuất hiện nhiều nhất trong tập dữ liệu, tức là giá trị có tần suất xuất hiện cao nhất.

* Tham số đầu vào:

+ Hàm statistics.mode(data) có một tham số:

* ‘data’: Đây là một tập hợp các giá trị số, đại diện cho dữ liệu bạn muốn tính giá trị ‘mode’.
* Ý nghĩa của kết quả trả về:

+ Kết quả trả về là giá trị ‘mode’ của tập dữ liệu ‘data’. Nếu có nhiều giá trị có cùng tần suất xuất hiện cao nhất, hàm sẽ trả về giá trị ‘mode’ đầu tiên mà nó gặp.

+ ‘mode’ là giá trị điển hình nhất và đóng vai trò là thước đo vị trí trung tâm.

+ Nếu muốn lấy ‘mode’ lớn nhất hoặc nhỏ nhất trong số các giá trị có cùng tần suất xuất hiện thì sử dụng ‘max(multimode(data))’ hoặc ‘min(multimode(data))’.

+ Nếu dữ liệu đầu vào trống, số lỗi thống kê (StatisticsError) sẽ được tăng lên.

+ Thay đổi trong phiên bản Python 3.8: Bây giờ xử lý các bộ dữ liệu đa phương thức bằng cách trả về chế độ đầu tiên gặp phải. Trước đây, nó đã tăng số lỗi thống kê (StatisticsError) khi tìm thấy nhiều hơn một chế độ.

* Mã ví dụ:

import statistics

data\_list = [2, 4, 6, 4, 8, 10, 4]

# Tính giá trị mode của tập dữ liệu

mode\_value = statistics.mode(data\_list)

# In kết quả

print("Dữ liệu:", data\_list)

print("Giá trị mode:", mode\_value)

1. *statistics.multimode(data):*

* Cách sử dụng:

+ Chúng ta cần import thư viện ‘statistics’ trước khi sử dụng hàm. Sau đó, gọi hàm ‘statistics.multimode(data)’ để tìm các giá trị mode của một tập dữ liệu ‘data’.

* Tham số đầu vào:

+ Hàm statistics.multimode(data) có một tham số:

* ‘data’: Đây là một tập hợp các giá trị số, đại diện cho dữ liệu bạn muốn tìm các giá trị mode.
* Ý nghĩa của kết quả trả về:

+ Kết quả trả về là một danh sách chứa tất cả các giá trị mode của tập dữ liệu ‘data’. Nếu chỉ có một giá trị mode duy nhất, kết quả sẽ là một danh sách chứa một phần tử. Nếu có nhiều giá trị có cùng tần suất xuất hiện cao nhất, kết quả sẽ chứa tất cả các giá trị mode đó.

* Mã ví dụ:

import statistics

data\_list = [2, 4, 6, 4, 8, 10, 4]

# Tìm các giá trị mode của tập dữ liệu

mode\_values = statistics.multimode(data\_list)

# In kết quả

print("Dữ liệu:", data\_list)

print("Các giá trị mode:", mode\_values)

1. *statistics.quantiles(**data, \*, n=4, method='exclusive'):*

* Cách sử dụng:

+ Chúng ta cần import thư viện ‘statistics’ trước khi sử dụng hàm. Sau đó, gọi hàm ‘statistics.quantiles(data, \*, n=4, method='exclusive')’ để tính các phân vị của một tập dữ liệu ‘data’. Có nhiều cách tính phân vị và statistics.quantiles() cung cấp hai phương pháp tính phân vị, bao gồm phân vị "exclusive" (mặc định) và phân vị "inclusive". ‘data’: Đây là một tập hợp các giá trị số, đại diện cho dữ liệu bạn muốn tính các phân vị. ‘n’: Đây là số lượng phân vị bạn muốn tính. Mặc định là 4, tức là tính phân vị thứ 1, 2 và 3.

* Tham số đầu vào:

+ Hàm statistics.quantiles(data, \*, n=4, method='exclusive') có ba tham số:

* ‘data’: Đây là một tập hợp các giá trị số, đại diện cho dữ liệu bạn muốn tính các phân vị.
* ‘n’: Đây là số lượng phân vị bạn muốn tính. Mặc định là 4, tức là tính phân vị thứ 1, 2 và 3.
* ‘method’: Đây là phương pháp tính phân vị, có hai lựa chọn là "exclusive" (mặc định) và "inclusive".
* Ý nghĩa của kết quả trả về:

+ Kết quả trả về là một danh sách chứa các giá trị phân vị của tập dữ liệu ‘data’. Số lượng phần tử trong danh sách sẽ bằng ‘n’, là số lượng phân vị bạn đã yêu cầu tính. Nếu ‘n=4’, danh sách sẽ chứa ba phân vị thứ 1, 2 và 3. Nếu ‘n=5’, danh sách sẽ chứa bốn phân vị thứ 1, 2, 3 và 4, và cứ tiếp tục như vậy.

+ Chúng ta có thể đặt n = 4 để tính các phân vị (quartiles - tứ phân vị). Đặt n = 10 để tính các phân vị (deciles - phân vị thập phân). Đặt n = 100 để tính các phân vị (percentiles - phân vị phần trăm) và trả về 99 điểm cắt để chia dữ liệu thành 100 nhóm có cùng kích thước.

+ Nếu n không ít nhất bằng 1, hàm sẽ tạo lỗi thống kê (StatisticsError).

+ Dữ liệu đầu vào có thể là bất kỳ đối tượng có thể lặp lại (iterable) nào chứa dữ liệu mẫu. Để có kết quả có ý nghĩa, số điểm dữ liệu trong dữ liệu (data) nên lớn hơn ‘n’. Nếu có ít nhất hai điểm dữ liệu, hàm sẽ không tạo lỗi thống kê (StatisticsError).

+ Các điểm cắt được xác định bằng cách nội suy tuyến tính từ hai điểm dữ liệu gần nhất.

+ Phương pháp tính phân vị có thể được điều chỉnh tùy thuộc vào việc dữ liệu bao gồm hay loại trừ các giá trị cực đại và cực tiểu của quần thể.  
+ Phương pháp mặc định là "exclusive" và được sử dụng cho dữ liệu lấy mẫu từ một quần thể có thể có các giá trị cực đại và cực tiểu hơn so với các mẫu. Phần trăm của quần thể nằm dưới i phần tử trong số m phần tử dữ liệu được sắp xếp được tính là i / (m + 1). Với chín giá trị mẫu, phương pháp này sẽ sắp xếp chúng và gán các phân vị sau đây: 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%.  
+ Việc thiết lập phương pháp thành "inclusive" được sử dụng để miêu tả dữ liệu quần thể hoặc cho các mẫu đã biết bao gồm các giá trị cực đại và cực tiểu của quần thể. Giá trị tối thiểu trong dữ liệu được xem như phân vị 0 và giá trị tối đa được xem như phân vị 100. Phần trăm của quần thể nằm dưới i phần tử trong số m phần tử dữ liệu được sắp xếp được tính là (i - 1) / (m - 1). Với 11 giá trị mẫu, phương pháp này sẽ sắp xếp chúng và gán các phân vị sau đây: 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100%.

* Mã ví dụ:

import statistics

data\_list = [10, 15, 20, 25, 30, 35, 40, 45]

# Tính các phân vị (25%, 50% và 75%)

quantiles\_values = statistics.quantiles(data\_list, n=4)

# In kết quả

print("Dữ liệu:", data\_list)

print("Các phân vị (25%, 50%, 75%):", quantiles\_values)

* 1. Đưa ra mã ví dụ của riêng tôi. Chụp ảnh màn hình cho hoạt động của chúng:

1. *statistics.mean(data):*

* Mã ví dụ của riêng tôi:

import statistics

# Danh sách các điểm số của các học sinh

diem\_so = [85, 90, 78, 92, 88, 95, 83]

# Tính điểm trung bình của các học sinh

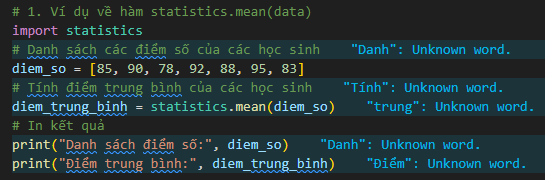
diem\_trung\_binh = statistics.mean(diem\_so)

# In kết quả

print("Danh sách điểm số:", diem\_so)

print("Điểm trung bình:", diem\_trung\_binh)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 1.3.1.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 1.3.1.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.1.a.

1. *tatistics.fmean(data, weights=None):*

* Mã ví dụ của riêng tôi:

import statistics

# Danh sách các số cần tính giá trị trung bình có trọng số

data\_list = [10, 20, 30, 40, 50]

# Trọng số tương ứng với các giá trị trong data\_list

weights\_list = [2, 3, 1, 4, 2]

# Tính giá trị trung bình có trọng số của tập dữ liệu

weighted\_mean = statistics.fmean(data\_list, weights=weights\_list)

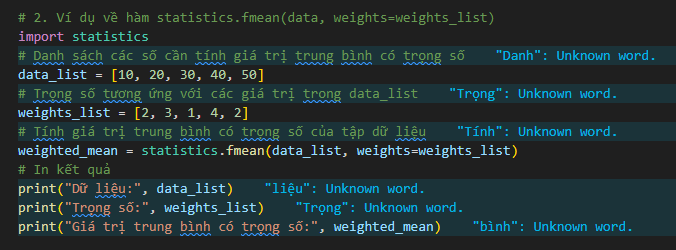
# In kết quả

print("Dữ liệu:", data\_list)

print("Trọng số:", weights\_list)

print("Giá trị trung bình có trọng số:", weighted\_mean)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 1.3.2.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 1.3.2.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.2.a.

1. *statistics.geometric\_mean(data):*

* Mã ví dụ của riêng tôi:

import statistics

# Danh sách các số cần tính trung bình hình học

data\_list = [2, 4, 8, 16, 32]

# Tính trung bình hình học của tập dữ liệu

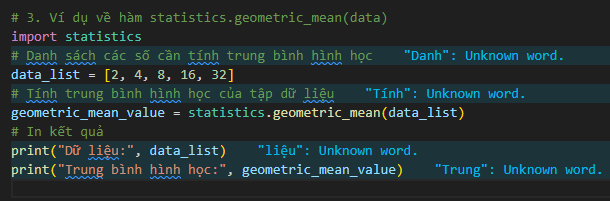
geometric\_mean\_value = statistics.geometric\_mean(data\_list)

# In kết quả

print("Dữ liệu:", data\_list)

print("Trung bình hình học:", geometric\_mean\_value)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 1.3.3.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 1.3.3.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.3.a.

1. *statistics.harmonic\_mean(data, weights=None):*

* Mã ví dụ của riêng tôi:

import statistics

data\_list = [10, 15, 20, 25]

# Tính giá trị trung bình nghịch đảo

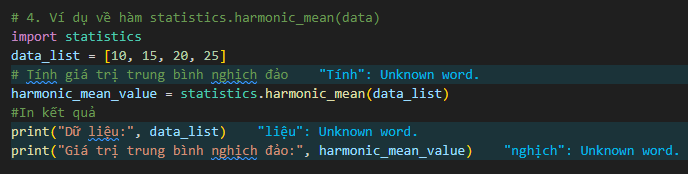
harmonic\_mean\_value = statistics.harmonic\_mean(data\_list)

# In kết quả

print("Dữ liệu:", data\_list)

print("Giá trị trung bình nghịch đảo:", harmonic\_mean\_value)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 1.3.4.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 1.3.4.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.4.a.

1. *statistics.median(data):*

* Mã ví dụ của riêng tôi:

import statistics

data\_list = [10, 15, 20, 25, 30]

# Tính giá trị trung vị

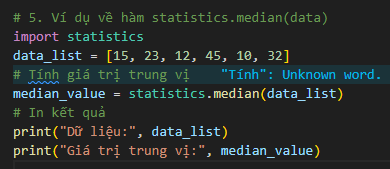
median\_value = statistics.median(data\_list)

# In kết quả

print("Dữ liệu:", data\_list)

print("Giá trị trung vị:", median\_value)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 1.3.5.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 1.3.5.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.5.a.

1. *statistics.median\_low(data):*

* Mã ví dụ của riêng tôi:

import statistics

data\_list = [5, 8, 12, 15, 20, 25]

# Tính giá trị trung vị (lấy giá trị thấp hơn nếu số lượng phần tử là lẻ)

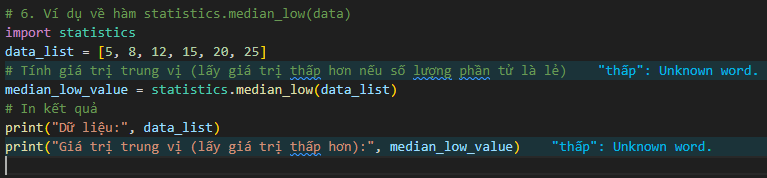
median\_low\_value = statistics.median\_low(data\_list)

# In kết quả

print("Dữ liệu:", data\_list)

print("Giá trị trung vị (lấy giá trị thấp hơn):", median\_low\_value)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 1.3.6.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 1.3.6.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.6.a.

1. *statistics.median\_high(data):*

* Mã ví dụ của riêng tôi:

import statistics

data\_list = [5, 8, 12, 15, 20, 25]

# Tính giá trị trung vị (lấy giá trị cao hơn nếu số lượng phần tử là lẻ)

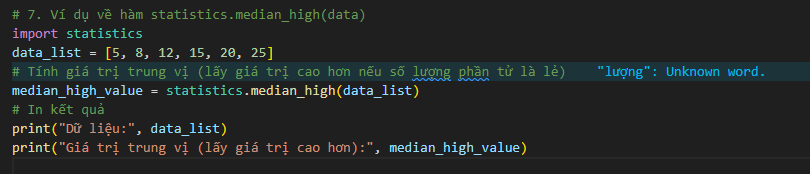
median\_high\_value = statistics.median\_high(data\_list)

# In kết quả

print("Dữ liệu:", data\_list)

print("Giá trị trung vị (lấy giá trị cao hơn):", median\_high\_value)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 1.3.7.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 1.3.7.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.7.a.

1. *statistics.median\_grouped(data, interval=1):*

* Mã ví dụ của riêng tôi:

import statistics

data\_list = [15, 20, 25, 30, 35, 40]

interval\_size = 5

# Tính giá trị trung vị của tập dữ liệu sau khi được nhóm lại vào các khoảng

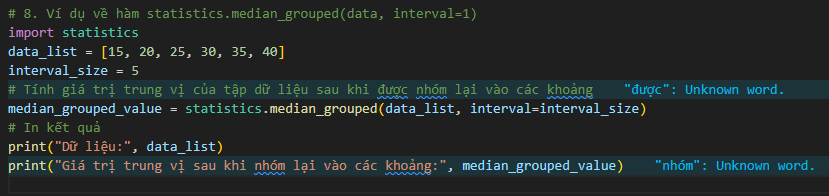
median\_grouped\_value = statistics.median\_grouped(data\_list, interval=interval\_size)

# In kết quả

print("Dữ liệu:", data\_list)

print("Giá trị trung vị sau khi nhóm lại vào các khoảng:", median\_grouped\_value)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 1.3.8.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 1.3.8.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.8.a.

1. *statistics.mode(data):*

* Mã ví dụ của riêng tôi:

#9. Ví dụ về hàm statistics.mode(data)

import statistics

data\_list = [2, 4, 6, 4, 8, 10, 4]

# Tính giá trị mode của tập dữ liệu

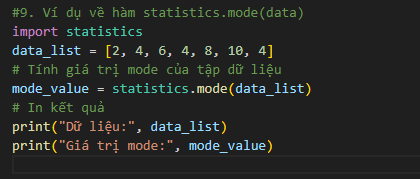
mode\_value = statistics.mode(data\_list)

# In kết quả

print("Dữ liệu:", data\_list)

print("Giá trị mode:", mode\_value)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 1.3.9.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 1.3.9.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.9.a.

1. *statistics.multimode(data):*

* Mã ví dụ của riêng tôi:

# 10. Ví dụ về hàm statistics.multimode(data)

import statistics

data\_list = [2, 4, 6, 4, 8, 10, 4]

# Tìm các giá trị mode của tập dữ liệu

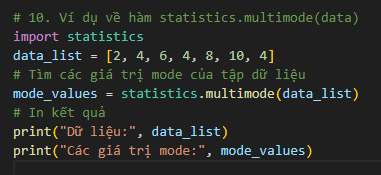
mode\_values = statistics.multimode(data\_list)

# In kết quả

print("Dữ liệu:", data\_list)

print("Các giá trị mode:", mode\_values)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 1.3.10.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 1.3.10.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.10.a.

1. *statistics.quantiles(data, \*, n=4, method='exclusive'):*

* Mã ví dụ của riêng tôi:

# 11. Ví dụ về hàm statistics.multimode(data, n = 4) và method không để gì thì mặc định là exclusive

import statistics

data\_list = [10, 15, 20, 25, 30, 35, 40, 45]

# Tính các phân vị (25%, 50% và 75%)

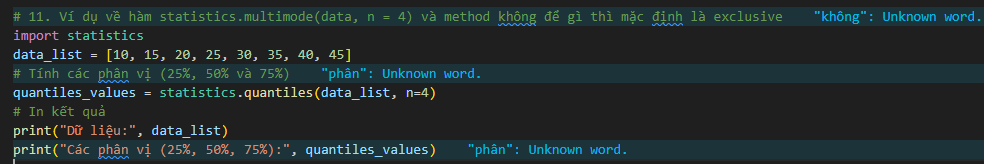
quantiles\_values = statistics.quantiles(data\_list, n=4)

# In kết quả

print("Dữ liệu:", data\_list)

print("Các phân vị (25%, 50%, 75%):", quantiles\_values)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 1.3.11.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 1.3.11.b. Ảnh chụp kết quả của đoạn code trong hình 1.3.11.a.

CHƯƠNG 2 – ĐO LƯỜNG SỰ LAN TRUYỀN (SỰ PHÂN TÁN)

**2.1 Tổng quan về các hàm:**

1. *statistics.pstdev(data, mu=None):* Hàm ‘statistics.pstdev(data, mu=None)’ trong thư viện ‘statistics’ của Python được sử dụng để tính độ lệch chuẩn của quần thể (population standard deviation), đó là căn bậc hai của phương sai của quần thể (population variance). Độ lệch chuẩn là một trong những độ đo quan trọng trong thống kê, dùng để đo lường mức độ phân tán của các giá trị trong quần thể so với giá trị trung bình của quần thể. Hãy xem hàm ‘pvariance()’ để biết thêm thông tin về các đối số và chi tiết khác.
2. *statistics.pvariance(data, mu=None):* Hàm ‘statistics.pvariance(data, mu=None)’ trong thư viện ‘statistics’ của Python được sử dụng để tính phương sai của quần thể (population variance). Phương sai là một trong những độ đo quan trọng trong thống kê, dùng để đo lường sự biến đổi (sự lan truyền hoặc phân tán) của dữ liệu. Phương sai lớn cho thấy dữ liệu lan truyền ra xa; phương sai nhỏ cho thấy dữ liệu tập trung gần trung bình.
3. *statistics.stdev(data, xbar=None):* Hàm ‘statistics.stdev(data, xbar=None)’ trong thư viện ‘statistics’ của Python được sử dụng để tính độ lệch chuẩn của mẫu (sample standard deviation), tức là căn bậc hai của phương sai của mẫu (sample variance). Độ lệch chuẩn là một trong những độ đo quan trọng trong thống kê, dùng để đo lường mức độ phân tán của các giá trị trong mẫu so với giá trị trung bình của mẫu.
4. *statistics.variance(data, xbar=None):* Hàm ‘statistics.variance(data, xbar=None)’ trong thư viện ‘statistics’ của Python được sử dụng để tính phương sai mẫu (sample variance) của một tập hợp dữ liệu (data) chứa ít nhất hai số thực. Phương sai là một đại lượng đo lường mức độ biến thiên (dispersion) của dữ liệu. Phương sai lớn cho thấy dữ liệu được phân tán xa khỏi giá trị trung bình, còn phương sai nhỏ cho thấy dữ liệu được tập trung gần giá trị trung bình.

**2.2 Mỗi hàm cần được giải thích về cách sử dụng, tham số đầu vào, ý nghĩa của kết quả trả về và mã ví dụ:**

1. *statistics.pstdev(data, mu=None):*

* Cách sử dụng:

+ Chúng ta cần import thư viện ‘statistics’ trước khi sử dụng hàm. Sau đó, gọi hàm ‘statistics.pstdev(data, mu=None)’ để tính độ lệch chuẩn của quần thể của một tập dữ liệu ‘data’ với tham số tùy chọn ‘mu’ - giá trị trung bình của quần thể. Nếu chúng ta cung cấp giá trị này, hàm sẽ tính độ lệch chuẩn xung quanh giá trị trung bình này. Nếu không cung cấp (giá trị mặc định là None), hàm sẽ tự động tính giá trị trung bình từ dữ liệu ‘data’.

* Tham số đầu vào:

+ Hàm statistics.pstdev(data, mu=None) có một tham số bắt buộc:

* ‘data’: Đây là tập hợp dữ liệu đại diện cho quần thể (population data) mà chúng ta muốn tính độ lệch chuẩn.

+ Ngoài ra, hàm cũng có một tham số tùy chọn:

* ‘mu’: Đây là giá trị trung bình của quần thể (population mean). Nếu chúng ta cung cấp giá trị này, hàm sẽ tính độ lệch chuẩn xung quanh giá trị trung bình này. Nếu không cung cấp (giá trị mặc định là None), hàm sẽ tự động tính giá trị trung bình từ dữ liệu ‘data’.
* Ý nghĩa của kết quả trả về:

+ Kết quả trả về là giá trị độ lệch chuẩn của quần thể, tức là độ lệch chuẩn của toàn bộ tập dữ liệu. Độ lệch chuẩn đo lường mức độ phân tán của các giá trị trong quần thể so với giá trị trung bình của quần thể. Kết quả trả về là một số thực không âm.

* Mã ví dụ:

import statistics

# Dữ liệu của quần thể (population data)

population\_data = [10, 12, 14, 16, 18, 20]

# Tính độ lệch chuẩn của quần thể

population\_stdev = statistics.pstdev(population\_data)

# In kết quả

print("Dữ liệu của quần thể:", population\_data)

print("Độ lệch chuẩn của quần thể:", population\_stdev)

1. *statistics.pvariance(data, mu=None):*

* Cách sử dụng:

+ Chúng ta cần import thư viện ‘statistics’ trước khi sử dụng hàm. Sau đó, gọi hàm ‘statistics.pvariance(data, mu=None)’ để tính phương sai của quần thể (population variance) của một tập dữ liệu ‘data’ với tham số tùy chọn ‘mu’ - giá trị trung bình của quần thể. Nếu chúng ta cung cấp giá trị này, hàm sẽ tính phương sai xung quanh giá trị trung bình này. Nếu không cung cấp (giá trị mặc định là None), hàm sẽ tự động tính giá trị trung bình từ dữ liệu ‘data’.

* Tham số đầu vào:

+ Hàm ‘statistics.pvariance(data, mu=None)’ có một đối số bắt buộc:

* ‘data’: Đây là một tập hợp dữ liệu không rỗng, chứa các số thực, đại diện cho quần thể mà chúng ta muốn tính phương sai.

+ Ngoài ra, hàm cũng có một đối số tùy chọn:

* ‘mu’: Đây là giá trị trung bình của quần thể (population mean). Nếu chúng ta cung cấp giá trị này, hàm sẽ tính phương sai xung quanh giá trị trung bình này. Nếu không cung cấp (giá trị mặc định là None), hàm sẽ tự động tính giá trị trung bình từ dữ liệu ‘data’.
* Ý nghĩa của kết quả trả về:

+ Kết quả trả về là giá trị phương sai của quần thể, tức là độ biến đổi của toàn bộ tập dữ liệu so với giá trị trung bình của quần thể. Phương sai là một số không âm.

+ Hàm này tính phương sai dựa trên toàn bộ quần thể (population), không phải mẫu dữ liệu (sample). Nếu chúng ta muốn tính phương sai từ mẫu dữ liệu thì nên sử dụng hàm ‘statistics.variance()’.

+ Hàm sẽ gây lỗi thống kê (StatisticsError) nếu dữ liệu ‘data’ là rỗng.

\* Lưu ý:

+ Khi được gọi với toàn bộ quần thể, hàm này tính phương sai của quần thể σ². Khi được gọi với một mẫu thay vì toàn bộ quần thể, hàm này tính phương sai không chính xác của mẫu s², còn được gọi là phương sai với N độ tự do.

+ Nếu chúng ta biết giá trị trung bình của quần thể μ, chúng ta có thể sử dụng hàm này để tính phương sai của một mẫu, bằng cách cung cấp giá trị trung bình của quần thể như đối số thứ hai. Miễn là các điểm dữ liệu là một mẫu ngẫu nhiên từ quần thể, kết quả sẽ là một ước tính không chệch của phương sai của quần thể.

+ Khi hàm được gọi với toàn bộ quần thể, nó trả về phương sai của quần thể σ².

+ Khi hàm được gọi với một mẫu thay vì toàn bộ quần thể, nó tính phương sai không chính xác của mẫu s², còn được gọi là phương sai với N độ tự do.

+ Nếu chúng ta biết giá trị trung bình của quần thể μ, chúng ta có thể sử dụng hàm này để tính phương sai của một mẫu, bằng cách cung cấp giá trị trung bình của quần thể như đối số thứ hai. Kết quả sẽ là một ước tính không chệch của phương sai của quần thể.

* Mã ví dụ:

import statistics

# Dữ liệu của quần thể (population data)

population\_data = [10, 12, 14, 16, 18, 20]

# Tính phương sai của quần thể

population\_variance = statistics.pvariance(population\_data)

# In kết quả

print("Dữ liệu của quần thể:", population\_data)

print("Phương sai của quần thể:", population\_variance)

1. *statistics.stdev(data, xbar=None):*

* Cách sử dụng:

+ Chúng ta cần import thư viện ‘statistics’ trước khi sử dụng hàm. Sau đó, gọi hàm ‘statistics.pvariance(data, mu=None)’ để tính độ lệch chuẩn của mẫu (sample standard deviation) của một tập dữ liệu ‘data’, tức là căn bậc hai của phương sai của mẫu (sample variance) với tham số tùy chọn ‘xbar’ - là giá trị trung bình của mẫu (sample mean). Nếu chúng ta cung cấp giá trị này, hàm sẽ tính độ lệch chuẩn mẫu xung quanh giá trị trung bình này. Nếu không cung cấp (giá trị mặc định là None), hàm sẽ tự động tính giá trị trung bình từ dữ liệu ‘data’.

* Tham số đầu vào:

+ Hàm statistics.stdev(data, xbar=None) có một đối số bắt buộc:

* ‘data’: Đây là một tập hợp dữ liệu không rỗng, chứa các số thực, đại diện cho mẫu mà chúng ta muốn tính độ lệch chuẩn.

+ Ngoài ra, hàm cũng có một đối số tùy chọn:

* ‘xbar’: Đây là giá trị trung bình của dữ liệu (sample mean). Nếu chúng ta cung cấp giá trị này, hàm sẽ tính độ lệch chuẩn mẫu xung quanh giá trị trung bình này. Nếu không cung cấp (giá trị mặc định là None), hàm sẽ tự động tính giá trị trung bình từ dữ liệu data.
* Ý nghĩa của kết quả trả về:

+ Kết quả trả về là giá trị độ lệch chuẩn của mẫu, tức là độ lệch chuẩn của toàn bộ tập dữ liệu mẫu so với giá trị trung bình của mẫu. Độ lệch chuẩn là một số không âm.

\* Lưu ý:

- Hàm này tính độ lệch chuẩn dựa trên mẫu dữ liệu (sample), không phải toàn bộ quần thể (population). Nếu chúng ta muốn tính độ lệch chuẩn từ toàn bộ quần thể, chúng ta nên sử dụng hàm ‘statistics.pstdev()’.

- Hàm sẽ gây lỗi thống kê (StatisticsError) nếu dữ liệu ‘data’ là rỗng hoặc chỉ có một phần tử.

* Mã ví dụ:

import statistics

# Dữ liệu mẫu (sample data)

sample\_data = [10, 12, 14, 16, 18, 20]

# Tính độ lệch chuẩn của mẫu

sample\_stdev = statistics.stdev(sample\_data)

# In kết quả

print("Dữ liệu mẫu:", sample\_data)

print("Độ lệch chuẩn của mẫu:", sample\_stdev)

1. *statistics.variance(data, xbar=None):*

* Cách sử dụng:

+ Chúng ta cần import thư viện ‘statistics’ trước khi sử dụng hàm. Sau đó, gọi hàm ‘statistics.pvariance(data, mu=None)’ để tính phương sai mẫu (sample variance) của một tập hợp dữ liệu (data) chứa ít nhất hai số thực với tham số tùy chọn ‘xbar’ - là giá trị trung bình của mẫu (sample mean). Nếu chúng ta cung cấp giá trị này, hàm sẽ tính phương sai mẫu xung quanh giá trị trung bình này. Nếu không cung cấp (giá trị mặc định là None), hàm sẽ tự động tính giá trị trung bình từ dữ liệu ‘data’.

+ Phương sai lớn cho thấy dữ liệu được phân tán xa khỏi giá trị trung bình, còn phương sai nhỏ cho thấy dữ liệu được tập trung gần giá trị trung bình.

* Tham số đầu vào:

+ Hàm statistics.variance(data, xbar=None) có hai đối số:

* ‘data’ (bắt buộc): Đây là một tập hợp dữ liệu chứa ít nhất hai số thực mà bạn muốn tính phương sai mẫu.
* ‘xbar’ (tùy chọn): Đây là giá trị trung bình của dữ liệu (sample mean). Nếu chúng ta cung cấp giá trị này, hàm sẽ tính phương sai mẫu xung quanh giá trị trung bình này. Nếu không cung cấp (giá trị mặc định là None), hàm sẽ tự động tính giá trị trung bình từ dữ liệu ‘data’.
* Ý nghĩa của kết quả trả về:

+ Kết quả trả về là giá trị phương sai mẫu của dữ liệu, tức là độ lệch chuẩn mẫu của toàn bộ tập dữ liệu so với giá trị trung bình của mẫu. Phương sai mẫu là một số không âm.

\* Lưu ý:

- Hàm này tính phương sai mẫu dựa trên mẫu dữ liệu (sample), không phải toàn bộ quần thể (population). Nếu chúng ta muốn tính phương sai từ toàn bộ quần thể, chúng ta nên sử dụng hàm ‘statistics.pvariance()’.

- Hàm sẽ gây lỗi thống kê (StatisticsError) nếu dữ liệu data có ít hơn hai giá trị.

- Đây là phương sai mẫu s² với điều chỉnh của Bessel, còn được gọi là phương sai với N-1 độ tự do. Miễn là các điểm dữ liệu đại diện (ví dụ: độc lập và phân phối đồng nhất), kết quả sẽ là ước tính không chệch của phương sai của quần thể thực tế.

* Mã ví dụ:

import statistics

# Dữ liệu mẫu (sample data)

sample\_data = [10, 12, 14, 16, 18, 20]

# Tính phương sai mẫu

sample\_variance = statistics.variance(sample\_data)

# In kết quả

print("Dữ liệu mẫu:", sample\_data)

print("Phương sai mẫu:", sample\_variance)

**2.3 Đưa ra mã ví dụ của riêng tôi. Chụp ảnh màn hình cho hoạt động của chúng:**

1. *statistics.pstdev(data, mu=None):*

* Mã ví dụ của riêng tôi:

import statistics

# Dữ liệu của quần thể (population data)

population\_data = [10, 12, 14, 16, 18, 20]

# Tính giá trị trung bình của quần thể (population mean)

population\_mean = statistics.mean(population\_data)

# Tính độ lệch chuẩn của quần thể (population standard deviation)

population\_stdev = statistics.pstdev(population\_data, mu=population\_mean)

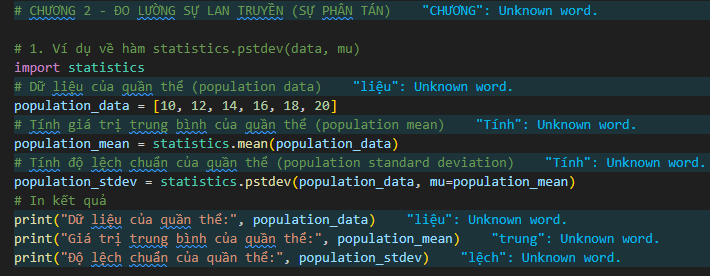
# In kết quả

print("Dữ liệu của quần thể:", population\_data)

print("Giá trị trung bình của quần thể:", population\_mean)

print("Độ lệch chuẩn của quần thể:", population\_stdev)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 2.3.1.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 2.3.1.b. Ảnh chụp kết quả của đoạn code trong hình 2.3.1.a.

1. *statistics.pvariance(data, mu=None):*

* Mã ví dụ của riêng tôi:

# 2. Ví dụ về hàm statistics.pvariance(data, mu=None)

import statistics

# Dữ liệu của quần thể (population data)

population\_data = [10, 12, 14, 16, 18, 20]

# Tính phương sai của quần thể

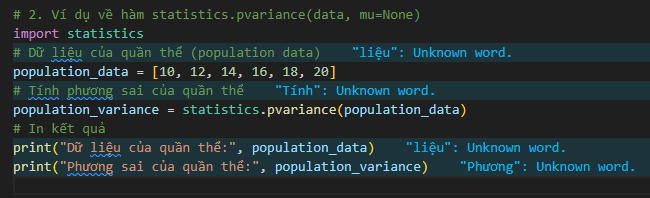
population\_variance = statistics.pvariance(population\_data)

# In kết quả

print("Dữ liệu của quần thể:", population\_data)

print("Phương sai của quần thể:", population\_variance)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 2.3.2.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 2.3.2.b. Ảnh chụp kết quả của đoạn code trong hình 2.3.2.a.

1. *statistics.stdev(data, xbar=None):*

* Mã ví dụ của riêng tôi:

# 3. Ví dụ về hàm statistics.stdev(data, xbar=None)

import statistics

# Dữ liệu mẫu (sample data)

sample\_data = [10, 12, 14, 16, 18, 20]

# Tính độ lệch chuẩn của mẫu

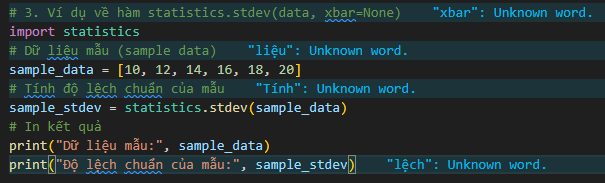
sample\_stdev = statistics.stdev(sample\_data)

# In kết quả

print("Dữ liệu mẫu:", sample\_data)

print("Độ lệch chuẩn của mẫu:", sample\_stdev)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 2.3.3.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 2.3.3.b. Ảnh chụp kết quả của đoạn code trong hình 2.3.3.a.

1. *statistics.variance(data, xbar=None):*

* Mã ví dụ của riêng tôi:

# 4. Ví dụ về hàm statistics.variance(data, xbar=None)

import statistics

# Dữ liệu mẫu (sample data)

sample\_data = [10, 12, 14, 16, 18, 20]

# Tính phương sai mẫu

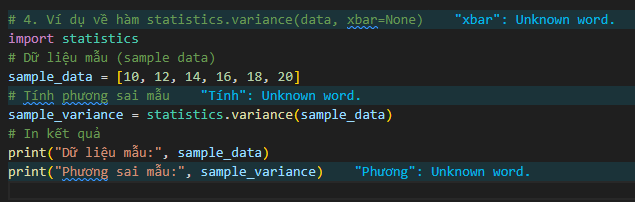
sample\_variance = statistics.variance(sample\_data)

# In kết quả

print("Dữ liệu mẫu:", sample\_data)

print("Phương sai mẫu:", sample\_variance)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 2.3.4.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 2.3.4.b. Ảnh chụp kết quả của đoạn code trong hình 2.3.4.a.

CHƯƠNG 3 – THỐNG KÊ VỀ MỐI QUAN HỆ GIỮA HAI ĐẦU VÀO

**3.1 Tổng quan về các hàm:**

1. *statistics.covariance(x, y, /):* Hàm ‘statistics.covariance(x, y, /)’ trong thư viện ‘statistics’ của Python được sử dụng để tính hiệp phương sai mẫu (sample covariance) giữa hai tập hợp dữ liệu đầu vào x và y. Hiệp phương sai là một đại lượng đo lường sự biến thiên liên hợp của hai tập dữ liệu.
2. *statistics.correlation(x, y, /):* Hàm ‘statistics.correlation(x, y, /)’ trong thư viện ‘statistics’ của Python được sử dụng để tính hệ số tương quan Pearson (Pearson's correlation coefficient) giữa hai tập hợp dữ liệu x và y. Hệ số tương quan Pearson được ký hiệu là r và có giá trị nằm trong khoảng từ -1 đến +1. Hệ số này đo lường mức độ và hướng của mối quan hệ tuyến tính giữa hai tập dữ liệu, trong đó +1 thể hiện mối quan hệ tuyến tính mạnh và tích cực, -1 thể hiện mối quan hệ tuyến tính mạnh và tiêu cực, và 0 thể hiện không có mối quan hệ tuyến tính.
3. *statistics.linear\_regression(x, y, /, \*, proportional=False):* Hàm statistics.linear\_regression(x, y, /, \*, proportional=False) trong thư viện statistics của Python được sử dụng để tính toán các tham số của mô hình hồi quy tuyến tính đơn giản (simple linear regression) bằng phương pháp bình phương tối thiểu (ordinary least squares).

3.2 Mỗi hàm cần được giải thích về cách sử dụng, tham số đầu vào, ý nghĩa của kết quả trả về và mã ví dụ:

1. *statistics.covariance(x, y, /):*

* Cách sử dụng:

+ Chúng ta cần import thư viện ‘statistics’ trước khi sử dụng hàm. Sau đó, gọi hàm ‘statistics.covariance(x, y, /)’ để tính hiệp phương sai mẫu (sample covariance) giữa hai tập hợp dữ liệu đầu vào x và y.

+ Hiệp phương sai là một đại lượng đo lường sự biến thiên liên hợp của hai tập dữ liệu.

* Tham số đầu vào:

+ Hàm statistics.covariance(x, y, /) có hai đối số:

* x (bắt buộc): Đây là một tập hợp dữ liệu chứa ít nhất hai số thực, là tập dữ liệu thứ nhất mà bạn muốn tính hiệp phương sai mẫu.
* y (bắt buộc): Đây cũng là một tập hợp dữ liệu chứa ít nhất hai số thực, là tập dữ liệu thứ hai mà bạn muốn tính hiệp phương sai mẫu.
* Cả hai dữ liệu đầu vào ‘x’ và ‘y’ phải có cùng độ dài (ít nhất hai phần tử), nếu không, hàm sẽ gây ra lỗi thống kê (StatisticsError).
* Ý nghĩa của kết quả trả về:

+ Kết quả trả về là giá trị hiệp phương sai mẫu giữa hai tập dữ liệu x và y. Hiệp phương sai đo lường mức độ tương quan tuyến tính giữa hai tập dữ liệu. Nếu giá trị hiệp phương sai dương, có nghĩa là hai tập dữ liệu đồng biến (cùng tăng hoặc giảm khi nhìn chung). Nếu giá trị hiệp phương sai âm, có nghĩa là hai tập dữ liệu nghịch biến (khi một tăng thì một giảm và ngược lại). Nếu giá trị hiệp phương sai gần bằng 0, có nghĩa là hai tập dữ liệu không có tương quan tuyến tính.

\* Lưu ý: Cả hai tập dữ liệu x và y phải có cùng độ dài (số lượng phần tử) và ít nhất là hai phần tử, nếu không, hàm sẽ gây lỗi thống kê (StatisticsError).

* Mã ví dụ:

import statistics

# Dữ liệu mẫu

x = [1, 2, 3, 4, 5]

y = [5, 4, 3, 2, 1]

# Tính hiệp phương sai mẫu

sample\_covariance = statistics.covariance(x, y)

# In kết quả

print("Dữ liệu x:", x)

print("Dữ liệu y:", y)

print("Hiệp phương sai mẫu:", sample\_covariance)

1. *statistics.correlation(x, y, /):*

* Cách sử dụng:

+ Chúng ta cần import thư viện ‘statistics’ trước khi sử dụng hàm. Sau đó, gọi hàm ‘statistics.correlation(x, y, /)’ để tính hệ số tương quan Pearson (Pearson's correlation coefficient) giữa hai tập hợp dữ liệu x và y.

+ Cả hai tập dữ liệu x và y phải có cùng độ dài (số lượng phần tử) và ít nhất là hai phần tử, nếu không, hàm sẽ gây lỗi thống kê (StatisticsError).

+ Hàm không yêu cầu các tập dữ liệu x và y phải là các tập hợp dữ liệu đồng nhất (constant), tức là giá trị của các phần tử không cần nhất thiết phải như nhau.

* Tham số đầu vào:

+ Hàm statistics.correlation(x, y, /) có hai đối số:

* ‘x’ (bắt buộc): Đây là một tập hợp dữ liệu chứa ít nhất hai số thực, là tập dữ liệu thứ nhất mà bạn muốn tính hệ số tương quan Pearson.
* ‘y’ (bắt buộc): Đây cũng là một tập hợp dữ liệu chứa ít nhất hai số thực, là tập dữ liệu thứ hai mà bạn muốn tính hệ số tương quan Pearson.
* Ý nghĩa của kết quả trả về:

+ Kết quả trả về là giá trị hệ số tương quan Pearson (r) giữa hai tập dữ liệu x và y. Giá trị của hệ số tương quan nằm trong khoảng từ -1 đến +1. Khi hệ số tương quan là +1, nó thể hiện mối quan hệ tuyến tính mạnh và tích cực giữa hai tập dữ liệu, tức là khi một tập dữ liệu tăng thì tập dữ liệu kia cũng tăng theo và ngược lại. Khi hệ số tương quan là -1, nó thể hiện mối quan hệ tuyến tính mạnh và tiêu cực giữa hai tập dữ liệu, tức là khi một tập dữ liệu tăng thì tập dữ liệu kia giảm và ngược lại. Khi hệ số tương quan là 0, nó thể hiện không có mối quan hệ tuyến tính giữa hai tập dữ liệu.

* Mã ví dụ:

import statistics

# Dữ liệu mẫu

x = [1, 2, 3, 4, 5]

y = [5, 4, 3, 2, 1]

# Tính hệ số tương quan Pearson

correlation\_coefficient = statistics.correlation(x, y)

# In kết quả

print("Dữ liệu x:", x)

print("Dữ liệu y:", y)

print("Hệ số tương quan Pearson:", correlation\_coefficient)

1. *statistics.linear\_regression(x, y, /, \*, proportional=False):*

* Cách sử dụng:

+ Chúng ta cần import thư viện ‘statistics’ trước khi sử dụng hàm. Sau đó, gọi hàm ‘statistics.linear\_regression(x, y, /, \*, proportional=False)’ để tính toán các tham số của mô hình hồi quy tuyến tính đơn giản (simple linear regression) bằng phương pháp bình phương tối thiểu (ordinary least squares).

+ Hồi quy tuyến tính đơn giản mô tả mối quan hệ giữa một biến độc lập x và một biến phụ thuộc y dưới dạng một phương trình tuyến tính:

y = slope \* x + intercept + noise

trong đó slope và intercept là các tham số hồi quy được ước lượng và noise đại diện cho sự biến động của dữ liệu không được giải thích bởi mô hình hồi quy tuyến tính (nó bằng hiệu giữa giá trị dự đoán và giá trị thực tế của biến phụ thuộc).

+ Cả hai tập dữ liệu đầu vào x và y phải có cùng độ dài (ít nhất hai phần tử) và biến độc lập x không được là một hằng số; nếu không, hàm sẽ gây lỗi thống kê (StatisticsError).

+ Nếu proportional được đặt thành True, biến độc lập x và biến phụ thuộc y được giả sử tỉ lệ thuận trực tiếp. Dữ liệu sẽ được phù hợp với một đường thẳng đi qua gốc tọa độ, khi đó tham số intercept luôn bằng 0.0 và phương trình tuyến tính đơn giản sẽ đơn giản hơn thành:

y = slope \* x + noise

+ Phương thức này được giới thiệu từ phiên bản Python 3.10 và đã thêm hỗ trợ cho proportional từ phiên bản Python 3.11 trở đi.

* Tham số đầu vào:

+ Hàm statistics.linear\_regression(x, y, /, \*, proportional=False) có các tham số như sau:

* ‘x’ (bắt buộc): Đây là một tập hợp dữ liệu chứa ít nhất hai số thực, đại diện cho biến độc lập ‘x’.
* ‘y’ (bắt buộc): Đây là một tập hợp dữ liệu chứa ít nhất hai số thực, đại diện cho biến phụ thuộc ‘y’.
* ‘proportional’ (không bắt buộc, mặc định là ‘False’): Đây là một giá trị boolean chỉ ra liệu x và y có tỉ lệ thuận trực tiếp hay không. Nếu ‘proportional=True’, phương trình hồi quy sẽ giả sử đi qua gốc tọa độ, ‘intercept’ sẽ luôn bằng 0.0.
* Ý nghĩa của kết quả trả về:

+ Kết quả trả về của hàm là một bộ giá trị (slope, intercept) biểu thị độ dốc và điểm cắt trục y của đường hồi quy tuyến tính.

+ Cả hai tập dữ liệu đầu vào x và y phải có cùng độ dài (ít nhất hai phần tử) và biến độc lập x không được là một hằng số; nếu không, hàm sẽ gây lỗi thống kê (StatisticsError).

* Mã ví dụ:

import statistics

# Dữ liệu mẫu

x = [1, 2, 3, 4, 5]

y = [2, 4, 5, 4, 5]

# Tính toán hồi quy tuyến tính đơn giản

slope, intercept = statistics.linear\_regression(x, y)

# In kết quả

print("Dữ liệu x:", x)

print("Dữ liệu y:", y)

print("Tham số slope (độ dốc):", slope)

print("Tham số intercept (điểm cắt trục y):", intercept)

**3.3 Đưa ra mã ví dụ của riêng tôi. Chụp ảnh màn hình cho hoạt động của chúng:**

1. *statistics.covariance(x, y, /):*

* Mã ví dụ của riêng tôi:

# CHƯƠNG 3 - THỐNG KÊ VỀ MỐI QUAN HỆ GIỮA HAI ĐẦU VÀO

# 1. Ví dụ về hàm statistics.covariance(x, y, /)

import statistics

# Dữ liệu mẫu

x = [1, 2, 3, 4, 5]

y = [5, 4, 3, 2, 1]

# Tính hiệp phương sai mẫu

sample\_covariance = statistics.covariance(x, y)

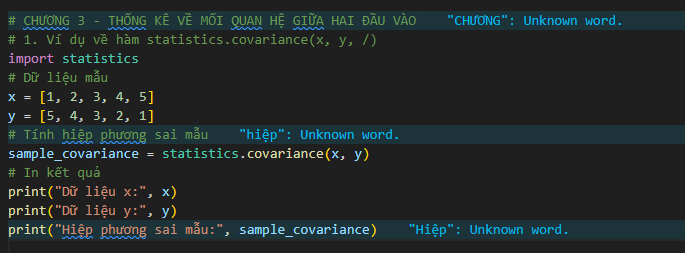
# In kết quả

print("Dữ liệu x:", x)

print("Dữ liệu y:", y)

print("Hiệp phương sai mẫu:", sample\_covariance)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 3.3.1.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 3.3.1.b. Ảnh chụp kết quả của đoạn code trong hình 3.3.1.a.

1. *statistics.correlation(x, y, /):*

* Mã ví dụ của riêng tôi:

# 2. Ví dụ về hàm statistics.correlation(x, y, /)

import statistics

# Dữ liệu mẫu

x = [1, 2, 3, 4, 5]

y = [5, 4, 3, 2, 1]

# Tính hệ số tương quan Pearson

correlation\_coefficient = statistics.correlation(x, y)

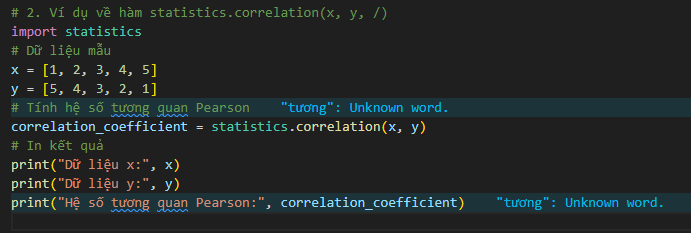
# In kết quả

print("Dữ liệu x:", x)

print("Dữ liệu y:", y)

print("Hệ số tương quan Pearson:", correlation\_coefficient)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 3.3.2.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 3.3.2.b. Ảnh chụp kết quả của đoạn code trong hình 3.3.2.a.

1. *statistics.linear\_regression(x, y, /, \*, proportional=False):*

* Mã ví dụ của riêng tôi:

import statistics

# Dữ liệu mẫu

x = [1, 2, 3, 4, 5]

y = [2, 4, 5, 4, 5]

# Tính toán hồi quy tuyến tính đơn giản

slope, intercept = statistics.linear\_regression(x, y)

# In kết quả

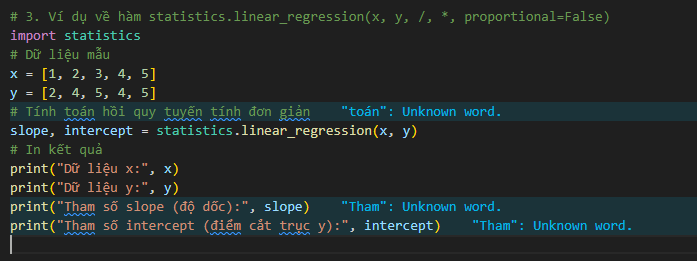
print("Dữ liệu x:", x)

print("Dữ liệu y:", y)

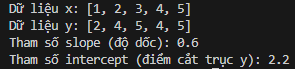
print("Tham số slope (độ dốc):", slope)

print("Tham số intercept (điểm cắt trục y):", intercept)

* Ảnh chụp màn hình cho hoạt động của chúng:



Hình 3.3.3.a. Ảnh chụp một đoạn code ví dụ của tôi.



Hình 3.3.3.b. Ảnh chụp kết quả của đoạn code trong hình 3.3.3.a.

**TÀI LIỆU THAM KHẢO**

* **Việt Nam:**

1. **Python Documentation (Tài liệu chính thức của Python):** Trang web chính thức của Python cung cấp tài liệu về thư viện **‘statistics’** và các hàm có trong đó. Chúng ta có thể tìm hiểu thêm tại: <https://docs.python.org/3/library/statistics.html>

[Page Not Found - GeeksforGeeks](https://www.geeksforgeeks.org/python-statistics/?fbclid=IwAR1nKTEIsTGO78koW3mYWjPWzqI68rhBfMTfS1y5U68oobGZdVNiLyGFBLQ)  
[Python statistics Module (w3schools.com)](https://www.w3schools.com/python/module_statistics.asp)

[Module statistics trong Python - Học lập trình Python - Viettuts](https://viettuts.vn/python/mo-dun-statistics-trong-python)

1. **Các khóa học và tài liệu Python tại các trung tâm đào tạo công nghệ:** Các trung tâm đào tạo công nghệ và lập trình ở Việt Nam thường cung cấp các khóa học và tài liệu học Python, bao gồm cả sử dụng thư viện **‘statistics’**.

* **Nước ngoài:**

1. **Python Documentation (Tài liệu chính thức của Python):** Như đã đề cập ở trên, trang web chính thức của Python cung cấp tài liệu về thư viện **‘statistics’** và các hàm có trong đó. Đây là nguồn tài liệu rất đáng tin cậy.
2. **Stack Overflow:** Diễn đàn lập trình quốc tế Stack Overflow thường có nhiều câu hỏi và bài giải liên quan đến việc sử dụng các thư viện trong Python. Chúng ta có thể tìm kiếm các câu hỏi và bài giải liên quan đến **‘statistics’** trên Stack Overflow.
3. **Python Programming Blogs:** Có nhiều blogs và trang web chia sẻ kiến thức lập trình Python, trong đó có các bài viết về sử dụng thư viện **‘statistics’**. Một số blogs phổ biến như Real Python, Towards Data Science, Analytics Vidhya, etc.
4. **Sách về thống kê và dữ liệu trong Python:** Có nhiều cuốn sách về thống kê và xử lý dữ liệu bằng Python cung cấp hướng dẫn về việc sử dụng thư viện **‘statistics’**. Một số cuốn sách như "Python for Data Analysis" của Wes McKinney và "Statistics for Data Scientists" của Thomas Miller có thể cung cấp thông tin hữu ích.
5. **Các khóa học trực tuyến:** Có nhiều nền tảng học trực tuyến như Coursera, Udemy, edX cung cấp khóa học về thống kê và Python, bao gồm cả sử dụng thư viện **‘statistics’**.

* **Mở rộng:** Để tìm hiểu chi tiết hơn, chúng ta nên tham gia vào cộng đồng lập trình Python để đặt câu hỏi và chia sẻ kinh nghiệm.