TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**TIỂU LUẬN GIỮA KÌ**

**XÁC XUẤT THỐNG KÊ CHO CÔNG NGHỆ THÔNG TIN**

Người thực hiện: **VÕ MINH THUẬN**

Lớp **: 21050201**

Khoá  **: 25**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

LỜI CẢM ƠN

Trước tiên với tình cảm sâu sắc và chân thành nhất, cho phép em được bày tỏ

lòng biết ơn đến tất cả các thầy cô đã tạo điều kiện, hỗ trợ, giúp đỡ chúng em

trong suốt quá trình học tập. Trong suốt thời gian khi bắt đầu học tập tại trường

đến nay, em đã nhận được rất nhiều sự quan tâm và giúp đỡ của thầy cô và bạn

bè.

Với lòng biết ơn sâu sắc nhất, em xin gửi đến quý Thầy Cô ở Khoa Công Nghệ

Thông Tin- Trường Đại học Tôn Đức Thắng đã truyền đạt vốn kiến thức quý

báu cho chúng em trong suốt thời gian học tập tại trường. Nhờ có những lời

hướng dẫn, dạy bảo của các thầy cô nên bài báo cáo của em mới hoàn thiện tốt

đẹp.

**CÔNG TRÌNH ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong luận văn còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung luận văn 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 6 tháng 11 năm 2022*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Võ Minh Thuận*

MỤC LỤC

Contents

[LỜI CẢM ƠN ii](#_Toc118666018)

[MỤC LỤC 1](#_Toc118666019)

[Chapter 1: Statistics library in Python 2](#_Toc118666020)

[Averages and measures of central location 2](#_Toc118666021)

[mean() 2](#_Toc118666022)

[fmean() 4](#_Toc118666023)

[geometric\_mean() 5](#_Toc118666024)

[harmonic \_mean() 6](#_Toc118666025)

[median() 7](#_Toc118666026)

[median\_low() 9](#_Toc118666027)

[median\_high() 10](#_Toc118666028)

[median\_grouped() 11](#_Toc118666029)

[mode () 13](#_Toc118666030)

[multimode () 15](#_Toc118666031)

[quantiles() 16](#_Toc118666032)

[Measures of spread 17](#_Toc118666033)

[pstdev() 17](#_Toc118666034)

[pvariance() 18](#_Toc118666035)

[stdev() 20](#_Toc118666036)

[variance() 21](#_Toc118666037)

[Statistics for relations between two inputs 22](#_Toc118666038)

[covariance() 22](#_Toc118666039)

[correlation() 23](#_Toc118666040)

[linear\_regression () 24](#_Toc118666041)

[Chapter 2: Histogram equalization algorithm 26](#_Toc118666042)

[Định nghĩa: 26](#_Toc118666043)

[Cách thực hiện: 26](#_Toc118666044)

[Công thức: 26](#_Toc118666045)

[Chapter 3 Implementation 27](#_Toc118666046)

[Code: 27](#_Toc118666047)

[Kết Quả Demo: 29](#_Toc118666048)

[Tài liệu tham khảo 30](#_Toc118666049)

# Chapter 1: Statistics library in Python

## Averages and measures of central location

|  |  |
| --- | --- |
|  | mean() |
| **Cách sử dụng** | Hàm mean(data) để tính trung bình của một số dữ liệu nhất định. Nó được tính bằng cách chia tổng của tất cả các data point cho số data point. |
| **Tham số đầu vào** | * List của ( int, float, decimal, fraction) * Nếu dữ liệu rỗng, một StatisticsError sẽ được đưa ra. |
| **Ý nghĩa đầu ra** | Trả về giá trị trung bình. Kiểu dữ liệu trả về giống kiểu dữ liệu của tham số đầu vào( int, float, decimal, fraction …) |

**Code:**

import statistics

from fractions import Fraction as F

from decimal import Decimal as D

x=statistics.mean([11, 2, 13, 14, 44])  #input int

print(x)    #output int

y=statistics.mean([F(8, 10), F(11, 20),

                   F(2, 5), F(28, 5)]) #input Fraction

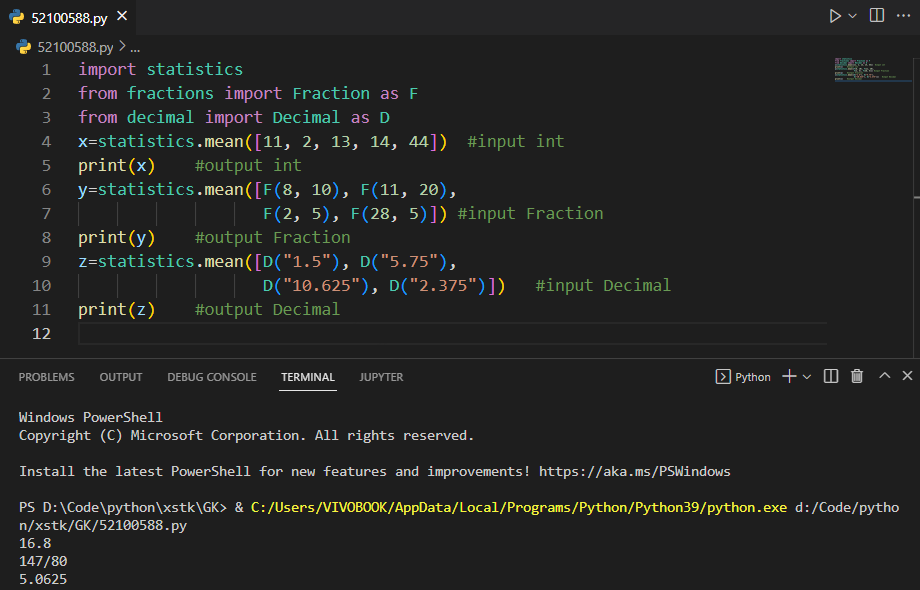
print(y)    #output Fraction

z=statistics.mean([D("1.5"), D("5.75"),

                   D("10.625"), D("2.375")])   #input Decimal

print(z)    #output Decimal

**Kết Quả Demo:**



|  |  |
| --- | --- |
|  | fmean() |
| **Cách sử dụng** | Hàm fmean(data) để tính trung bình của một số dữ liệu nhất định. Hàm fmean() chạy nhanh hơn hàm mean(). |
| **Tham số đầu vào** | * List của (int, float ) * Nếu dữ liệu rỗng, một StatisticsError sẽ được đưa ra. |
| **Ý nghĩa đầu ra** | Trả về giá trị trung bình. Kiểu dữ liệu trả về chuyển thành dạng float |

**Code:**

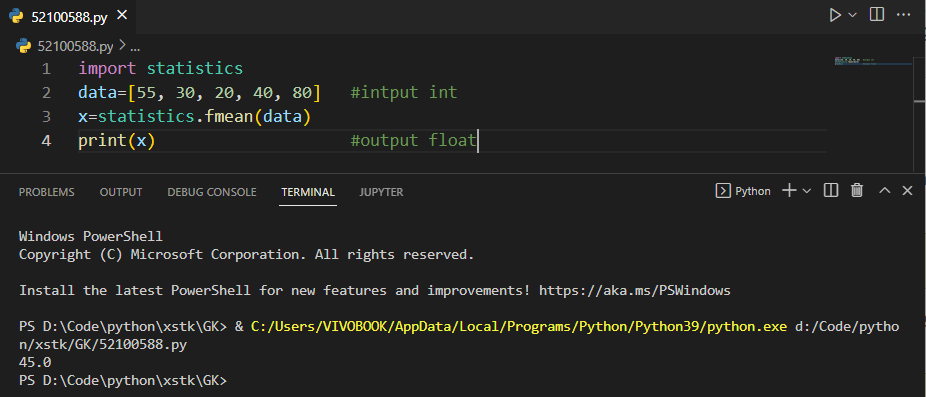
import statistics

data=[55, 30, 20, 40, 80]   #intput list of int

x=statistics.fmean(data)

print(x)                    #output float

**Kết Quả Demo:**



|  |  |
| --- | --- |
|  | geometric\_mean() |
| **Cách sử dụng** | Hàm geometric\_mean(data) dùng để tính trung bình hình học được tính theo công thức:  Geomatric Mean = |
| **Tham số đầu vào** | * List của (int, float ) * Nếu dữ liệu rỗng, một StatisticsError sẽ được đưa ra. |
| **Ý nghĩa đầu ra** | Trả về giá trị trung bình hình học. Kiểu dữ liệu trả về chuyển thành dạng float |

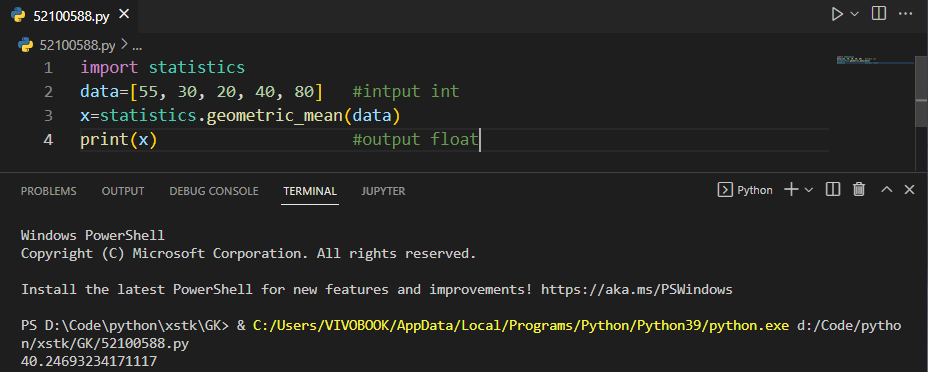
**Code:**

import statistics

data=[55, 30, 20, 40, 80]   #intput list of int

x=statistics.geometric\_mean(data)

print(x)                    #output float

**Kết Quả Demo: **

|  |  |
| --- | --- |
|  | harmonic \_mean() |
| **Cách sử dụng** | Hàm harmonic\_mean(data) dùng để tính trung bình điều hòa theo công thức:  Harmonic Mean = |
| **Tham số đầu vào** | * List của (int, float ) * Nếu dữ liệu rỗng, bất kỳ phần tử nào nhỏ hơn 0 hoặc nếu tổng trọng số không dương thì một StatisticsError sẽ được đưa ra. |
| **Ý nghĩa đầu ra** | Trả về giá trị trung bình điều hòa. Kiểu dữ liệu trả về chuyển thành dạng float |

**Code:**

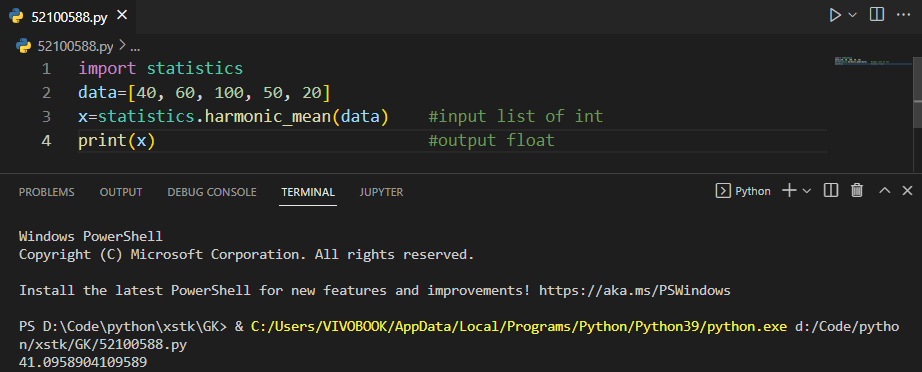
import statistics

data=[40, 60, 100, 50, 20]

x=statistics.harmonic\_mean(data)    #input list of int

print(x)                            #output float

**Kết Quả Demo:**



|  |  |
| --- | --- |
|  | median() |
| **Cách sử dụng** | Hàm median(data) tính trung vị của tập dữ liệu( là số ở giữa của tập dữ liệu) |
| **Tham số đầu vào** | * List của (int, float ) * Nếu dữ liệu rỗng, một StatisticsError sẽ được đưa ra. |
| **Ý nghĩa đầu ra** | * Nếu số lượng phần tử truyền vào chẵn thì trả về trung bình của 2 số ở giữa. Trả về kiểu float * Nếu số lượng phần tử truyền vào lẻ thì trả về phần tử ở giữa. Kiểu dữ liệu trả về là kiểu dữ liệu của phần tử ở giữa |

**Code:**

import statistics

data\_EvenEl=[1, 2, 4, 5]    #so luong phan tu chan

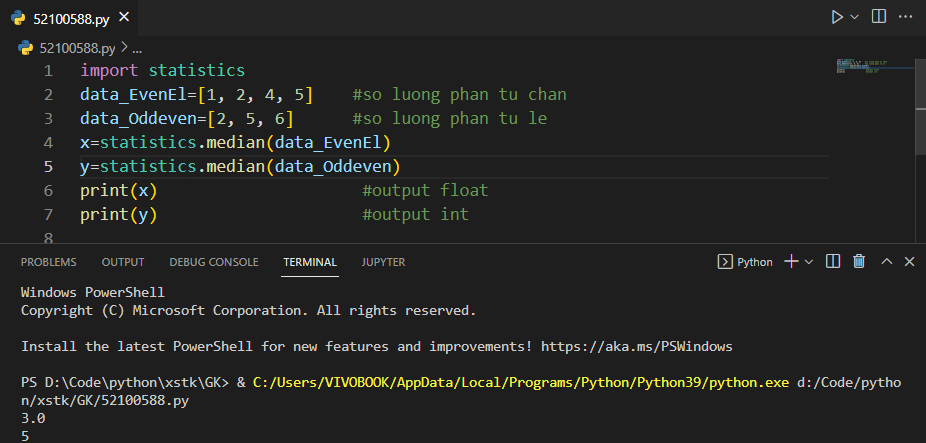
data\_Oddeven=[2, 5, 6]      #so luong phan tu le

x=statistics.median(data\_EvenEl)

y=statistics.median(data\_Oddeven)

print(x)                     #output float

print(y)                     #output int

**Kết Quả Demo: **

|  |  |
| --- | --- |
|  | median\_low() |
| **Cách sử dụng** | Hàm median\_low(data) tính trung vị thấp của tập dữ liệu |
| **Tham số đầu vào** | * List của (int, float) * Nếu dữ liệu rỗng, một StatisticsError sẽ được đưa ra. |
| **Ý nghĩa đầu ra** | * Nếu số lượng phần tử truyền vào chẵn thì trả về phần tử nhỏ hơn trong 2 phần tử ở giữa. Kiểu dữ liệu trả về là kiểu dữ liệu của phần tử nhỏ hơn đó * Nếu số lượng phần tử truyền vào lẻ thì trả về phần tử ở giữa. Kiểu dữ liệu trả về là kiểu dữ liệu của phần tử ở giữa |

**Code**:

import statistics

data\_EvenEl=[1, 2.5, 4, 5]    #so luong phan tu chan

data\_Oddeven=[2, 5, 6]      #so luong phan tu le

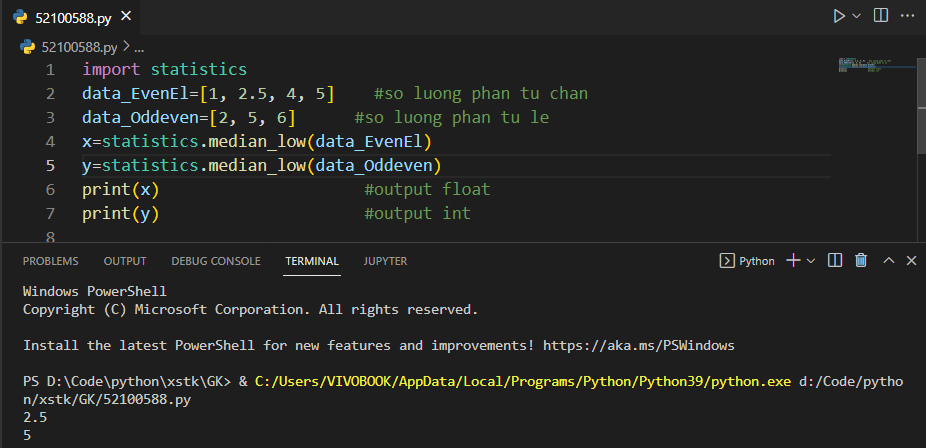
x=statistics.median\_low(data\_EvenEl)

y=statistics.median\_low(data\_Oddeven)

print(x)                     #output float

print(y)                     #output int

**Kết Quả Demo:**



|  |  |
| --- | --- |
|  | median\_high() |
| **Cách sử dụng** | Hàm median\_high(data) tính trung vị cao của tập dữ liệu |
| **Tham số đầu vào** | * List của (int, float ) * Nếu dữ liệu rỗng, một StatisticsError sẽ được đưa ra. |
| **Ý nghĩa đầu ra** | * Nếu số lượng phần tử truyền vào chẵn thì trả về phần tử lớn hơn trong 2 phần tử ở giữa. Kiểu dữ liệu trả về là kiểu dữ liệu của phần tử lớn hơn đó * Nếu số lượng phần tử truyền vào lẻ thì trả về phần tử ở giữa. Kiểu dữ liệu trả về là kiểu dữ liệu của phần tử ở giữa |

**Code:**

import statistics

data\_EvenEl=[1, 2, 4.5, 5]    #so luong phan tu chan

data\_Oddeven=[2, 5, 6]      #so luong phan tu le

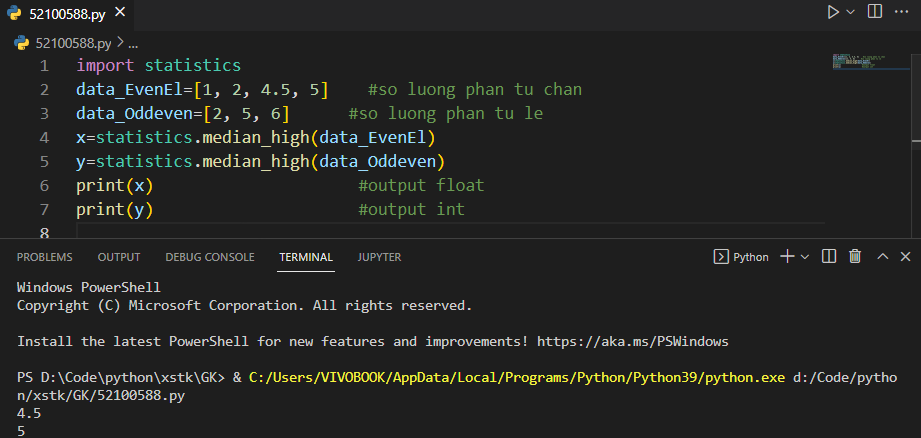
x=statistics.median\_high(data\_EvenEl)

y=statistics.median\_high(data\_Oddeven)

print(x)                     #output float

print(y)                     #output int

**Kết Quả Demo:**



|  |  |
| --- | --- |
|  | median\_grouped() |
| **Cách sử dụng** | Hàm median\_grouped(data, interval) tính trung vị của tập dữ liệu được nhóm( dữ liệu được sắp xếp theo thứ tự tăng dần và được ghi một cách liên tục). |
| **Tham số đầu vào** | List của (int, float ) và interval( xác định độ rộng của dữ liệu được nhóm và thay đổi) được mặc định là 1  Nếu dữ liệu rỗng, một StatisticsError sẽ được đưa ra. |
| **Ý nghĩa đầu ra** | Trả về giá trị trung bình của dữ liệu liên tục được nhóm lại, được tính bằng phân vị thứ 50 |

**Code :**

import statistics

data=[50, 51, 53, 54]

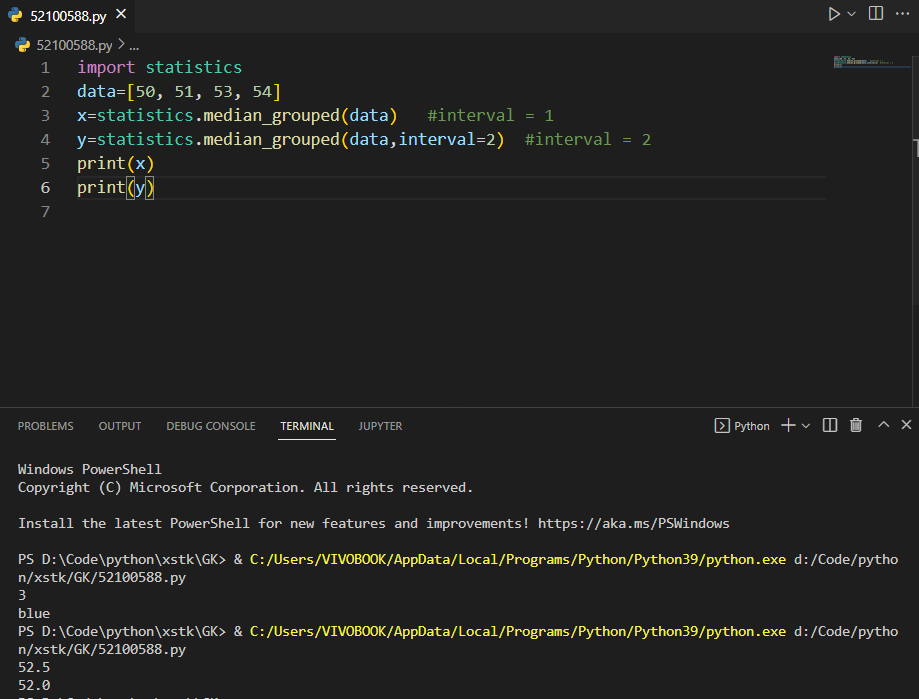
x=statistics.median\_grouped(data)   #interval = 1

y=statistics.median\_grouped(data,interval=2)  #interval = 2

print(x)

print(y)

**Kết Quả Demo:**



|  |  |
| --- | --- |
|  | mode () |
| **Cách sử dụng** | Hàm mode(data) trả về phần tử xuất hiện thường xuyên nhất |
| **Tham số đầu vào** | * List của (int, float hoặc string ) * Nếu dữ liệu rỗng, một StatisticsError sẽ được đưa ra. |
| **Ý nghĩa đầu ra** | Trả về phần tử xuất hiện nhiều nhất |

**Code:**

import statistics

data=[1, 2, 3, 4, 3, 2, 3, 1]

data\_1=["blue", "blue", "blue", "green", "red", "red"]

x=statistics.mode(data)

# 1 xuất hiện 2 lần

# 2 xuất hiện 2 lần

# 3 xuất hiện 3 lần ->output 3

# 4 xuất hiện 1 lần

y=statistics.mode(data\_1)

# blue xuất hiện 3 lần ->output blue

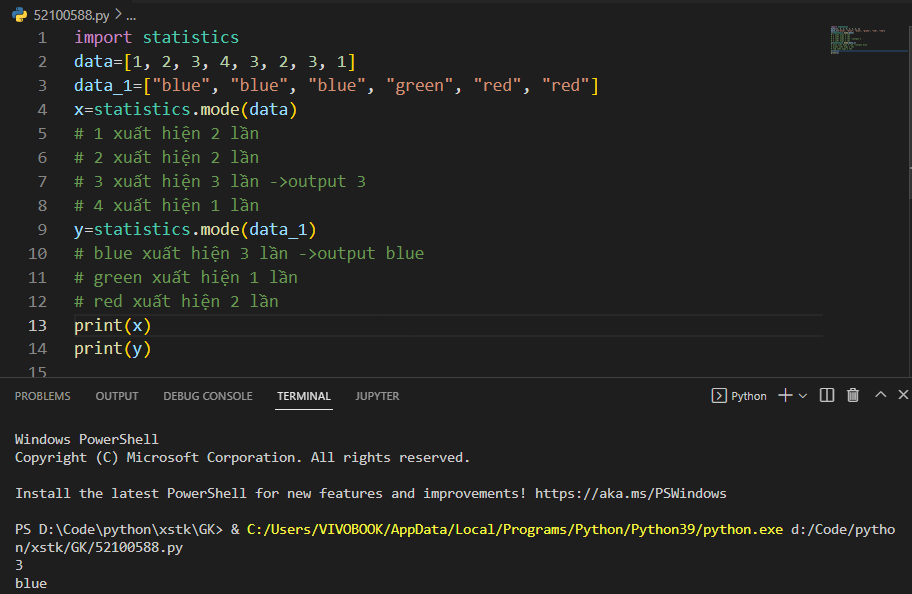
# green xuất hiện 1 lần

# red xuất hiện 2 lần

print(x)

print(y)

**Kết Quả Demo:**

****

|  |  |
| --- | --- |
|  | multimode () |
| **Cách sử dụng** | Hàm multimode(data) trả về phần tử xuất hiện thường xuyên nhất theo thứ tự lần đầu tiên chúng gặp trong dữ liệu |
| **Tham số đầu vào** | List của (int, float hoặc string ) |
| **Ý nghĩa đầu ra** | Trả về phần tử xuất hiện nhiều nhất. Có thể trả về danh sách nhiều phần tử nếu chúng xuất hiện số lần nhiều nhất đều bằng nhau hoặc trả về danh sách trống nếu dữ liệu truyền vào rỗng. |

**Code:**

import statistics

data=[]

data\_1=["blue", "blue", "blue", "green", "red", "red","red"]

x=statistics.multimode(data)    #rỗng

y=statistics.multimode(data\_1)

# blue xuất hiện 3 lần ->output blue

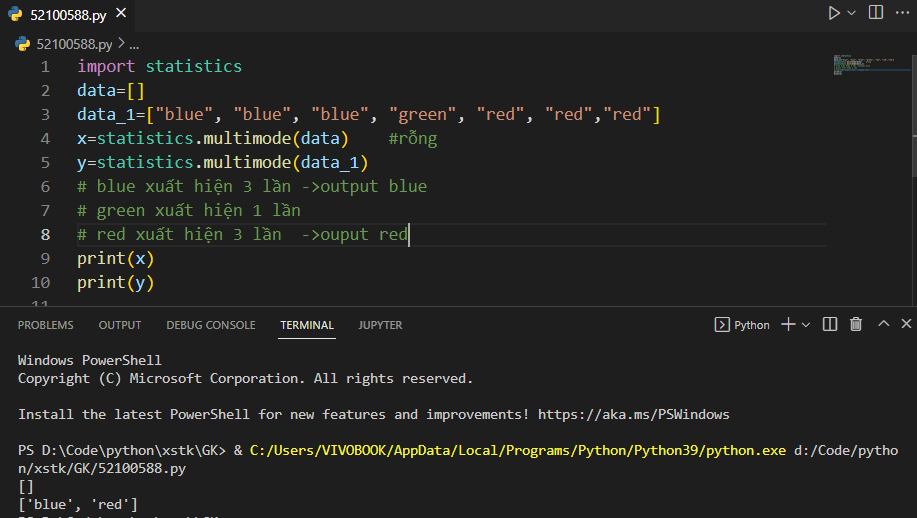
# green xuất hiện 1 lần

# red xuất hiện 3 lần  ->ouput red

print(x)

print(y)

**Kết Quả Demo:**

****

|  |  |
| --- | --- |
|  | quantiles() |
| **Cách sử dụng** | Hàm quantiles (data, \*, n=4, method = ‘exclusive’) chia dữ liệu thành n khoảng liên tục với xác suất bằng nhau |
| **Tham số đầu vào** | * List của (int, float) * Nếu dữ liệu ít hơn 2 giá trị, một StatisticsError sẽ được đưa ra. |
| **Ý nghĩa đầu ra** | Trả về danh sách các điểm cắt tách các khoảng n-1 |

**Code:**

import statistics

data=[105, 129, 87, 86, 111, 111, 89, 81, 108, 92, 110,

        100, 75, 105, 103, 109, 76, 119, 99, 91, 103, 129,

        106, 101, 84, 111, 74, 87, 86, 103, 103, 106, 86,

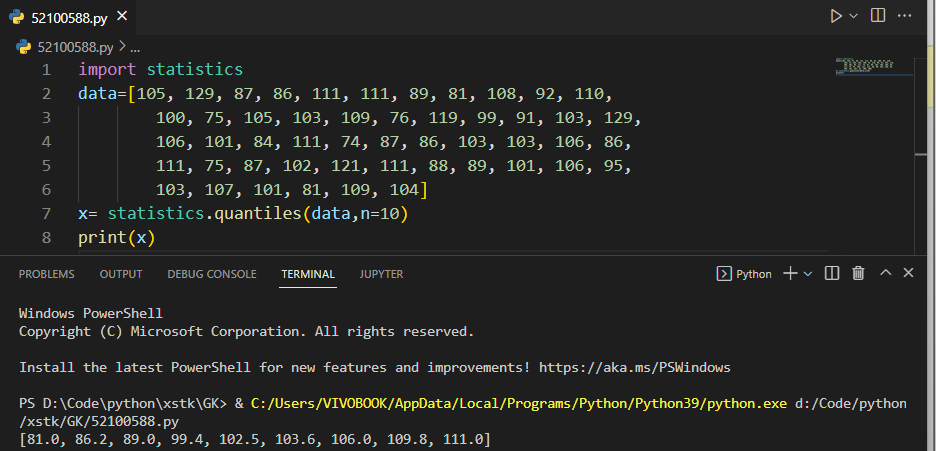
        111, 75, 87, 102, 121, 111, 88, 89, 101, 106, 95,

        103, 107, 101, 81, 109, 104]

x= statistics.quantiles(data,n=10)

print(x)

**Kết Quả Demo:**

****

## Measures of spread

|  |  |
| --- | --- |
|  | pstdev() |
| **Cách sử dụng** | Hàm pstdev(data) dùng để tính độ lệch chuẩn tổng thể (căn bậc 2 của phương sai tổng thể) |
| **Tham số đầu vào** | * List của (int, float) * Nếu dữ liệu rỗng, một StatisticsError sẽ được đưa ra. |
| **Ý nghĩa đầu ra** | Trả về độ lệch chuẩn tổng thể của tập dữ liệu. Trả về kiểu float |

**Code:**

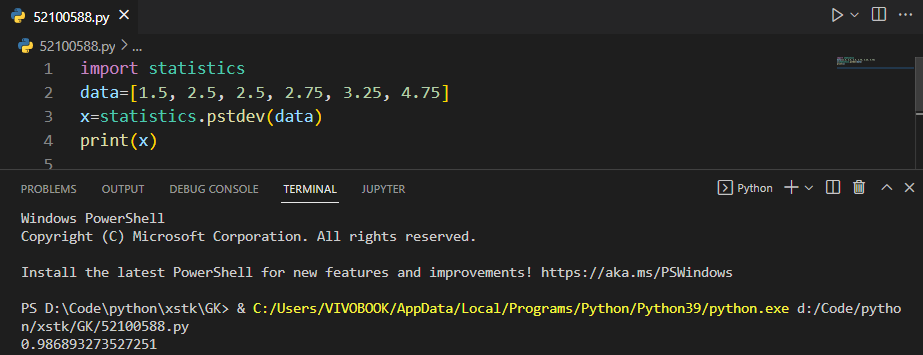
import statistics

data=[1.5, 2.5, 2.5, 2.75, 3.25, 4.75]

x=statistics.pstdev(data)

print(x)

**Kết Quả Demo:**

****

|  |  |
| --- | --- |
|  | pvariance() |
| **Cách sử dụng** | Hàm pvariance(data) dùng để tính phương sai tổng thể của dữ liệu |
| **Tham số đầu vào** | * List của (int, float, Decimal, Fraction) * Nếu dữ liệu rỗng, một StatisticsError sẽ được đưa ra. |
| **Ý nghĩa đầu ra** | Trả về phương sai tổng thể của tập dữ liệu. Kiểu dữ liệu phụ thuộc vào kiểu dữ liệu của tham số đầu vào |

**Code:**

import statistics

from fractions import Fraction as F

from decimal import Decimal as D

data=[1.5, 2.5, 2.5, 2.75, 3.25, 4.75]

data\_D=[D("27.5"), D("30.25"), D("30.25"), D("34.5"), D("41.75")]

data\_F=[F(1, 4), F(5, 4), F(1, 2)]

x=statistics.pvariance(data)

y=statistics.pvariance(data\_D)

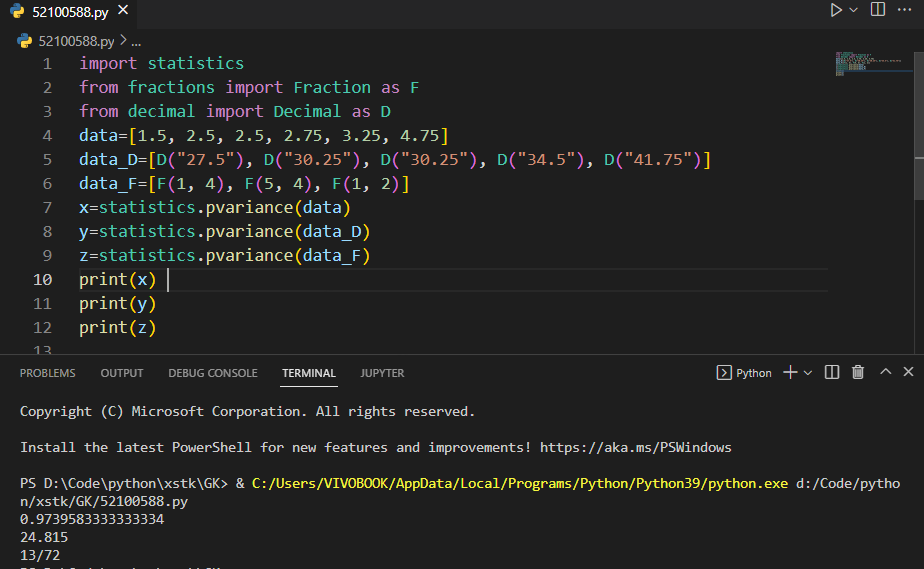
z=statistics.pvariance(data\_F)

print(x)

print(y)

print(z)

**Kết Quả Demo:**

****

|  |  |
| --- | --- |
|  | stdev() |
| **Cách sử dụng** | Hàm pvariance(data) dùng để tính độ lệch chuẩn của mẫu của dữ liệu( căn bậc 2 của phương sai mẫu) |
| **Tham số đầu vào** | * List của (int, float) * Nếu dữ liệu rỗng, một StatisticsError sẽ được đưa ra. |
| **Ý nghĩa đầu ra** | Trả về độ lệch chuẩn của mẫu của tập dữ liệu. Trả về kiểu float |

**Code:**

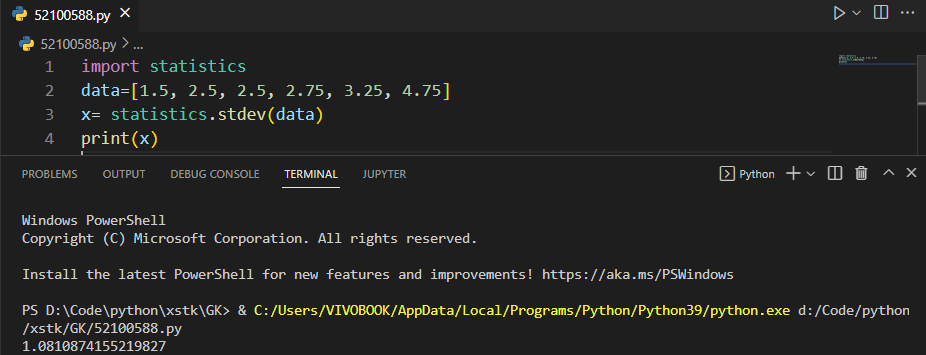
import statistics

data=[1.5, 2.5, 2.5, 2.75, 3.25, 4.75]  #input float

x= statistics.stdev(data)

print(x) #output float

**Kết Quả Demo:**

****

|  |  |
| --- | --- |
|  | variance() |
| **Cách sử dụng** | Hàm variance(data) dùng để tính phương sai mẫu của dữ liệu |
| **Tham số đầu vào** | * List của (int, float) * Nếu dữ liệu có ít hơn 2 giá trị, một StatisticsError sẽ được đưa ra. |
| **Ý nghĩa đầu ra** | Trả về phương sai mẫu của tập dữ liệu. Trả về kiểu float |

**Code:**

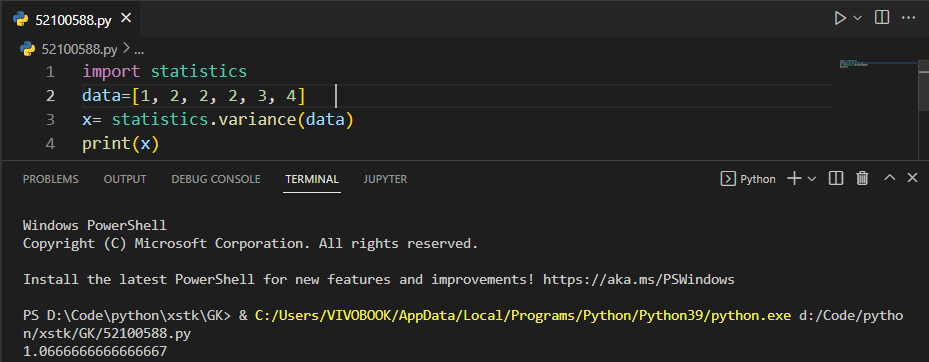
import statistics

data=[1, 2, 2, 2, 3, 4]

x= statistics.variance(data)

print(x)

**Kết Quả Demo:**

****

## Statistics for relations between two inputs

|  |  |
| --- | --- |
|  | covariance() |
| **Cách sử dụng** | Hàm covariance(data, data\_1) dùng để tính hiệp phương sai của dữ liệu( là thước đo thay đổi chung của 2 đầu vào) |
| **Tham số đầu vào** | * List của (int, float) * Cả 2 đầu vào phải có cùng độ dài(không ít hơn 2), nếu không thì một StatisticsError sẽ được đưa ra. |
| **Ý nghĩa đầu ra** | Trả về hiệp phương sai của tập dữ liệu. |

**Code:**

import statistics

data = [1, 2, 3, 4, 5, 6, 7, 8, 9]

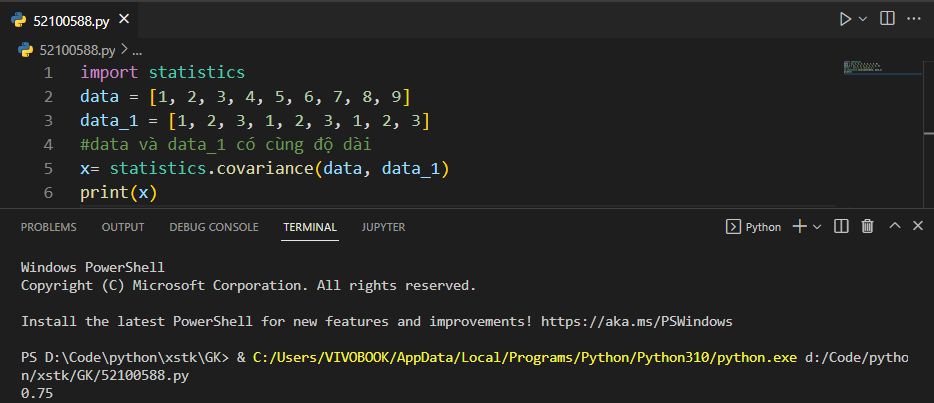
data\_1 = [1, 2, 3, 1, 2, 3, 1, 2, 3]

#data và data\_1 có cùng độ dài

x= statistics.covariance(data, data\_1)

print(x)

**Kết Quả Demo:**

****

|  |  |
| --- | --- |
|  | correlation() |
| **Cách sử dụng** | Hàm correlation (data, data\_1) dùng để tính hệ số tương quan Pearson |
| **Tham số đầu vào** | * List của (int, float) * Cả 2 đầu vào phải có cùng độ dài(không ít hơn 2), nếu không thì một StatisticsError sẽ được đưa ra. |
| **Ý nghĩa đầu ra** | Trả về hệ số tương quan Pearson. Có giá trị từ -1 đến 1 |

**Code:**

import statistics

data = [1, 2, 3, 4, 5, 6, 7, 8, 9]

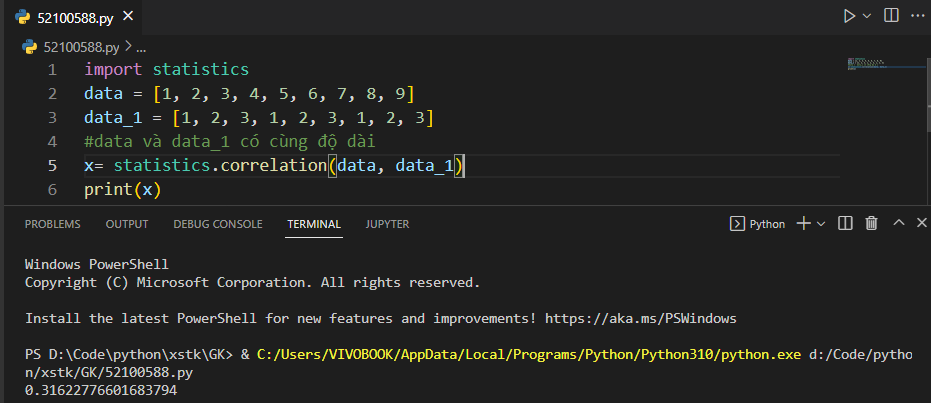
data\_1 = [1, 2, 3, 1, 2, 3, 1, 2, 3]

#data và data\_1 có cùng độ dài

x= statistics.correlation(data, data\_1)

print(x)

**Kết Quả Demo:**

****

|  |  |
| --- | --- |
|  | linear\_regression () |
| **Cách sử dụng** | Hàm linear\_regression(x, y) dùng để tính hồi quy tuyến tính theo công thức:  Trong đó:   * Slope và intercept: các tham số hồi quy được ước tính * Noise: đại diện cho sự biến thiên của dữ liệu |
| **Tham số đầu vào** | * List của (int, float) * Cả 2 đầu vào phải có cùng độ dài(không ít hơn 2),biến x không phải hằng số, nếu không thì một StatisticsError sẽ được đưa ra. |
| **Ý nghĩa đầu ra** | Trả về slope và intercept. |

**Code:**

import statistics

year = [1971, 1975, 1979, 1982, 1983]

films\_total = [1, 2, 3, 4, 5]

#data và data\_1 có cùng độ dài

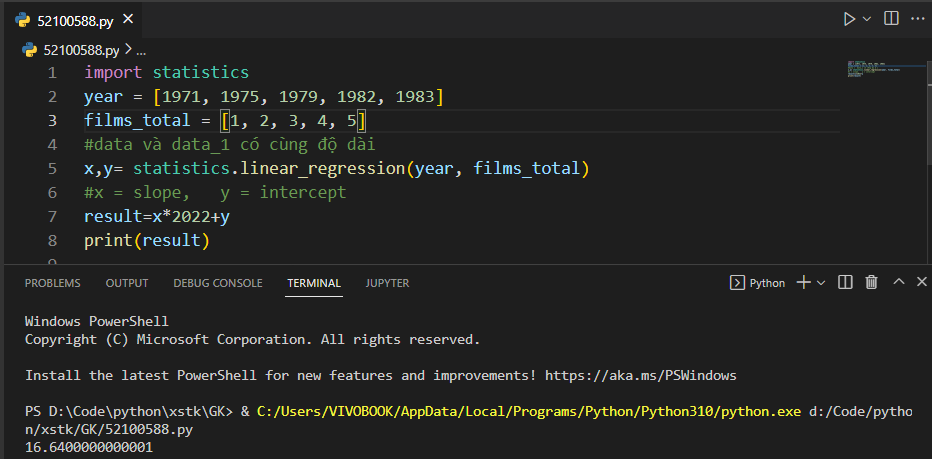
x,y= statistics.linear\_regression(year, films\_total)

#x = slope,   y = intercept

result=x\*2022+y

print(result)

**Kết Quả Demo:**

****

# Chapter 2: Histogram equalization algorithm

## Định nghĩa:

Thuật toán cân bằng biểu đồ là một phương pháp xử lí ảnh điều chỉnh độ tương phản bằng cách sử dụng biểu đồ của hình ảnh. Phương pháp này thường làm tăng độ tương phản toàn cục của nhiều hình ảnh, đặc biệt khi hình ảnh được biểu diễn bằng một dải giá trị cường độ đẹp. Thông qua điều chỉnh này, các cường độ có thể được phân phối tốt hơn trên biểu đồ bằng cách sử dụng đồng đều đầy đủ các cường độ. Điều này cho phép các khu vực có độ tương phản cục bộ thấp hơn để có được độ tương phản cao hơn. Phương pháp này hữu ích trong các hình ảnh có nền và nền vừa sáng hoặc vừa tối.

## Cách thực hiện:

Để thay đổi histogram, cách duy nhất chính là thay đổi intensity của các điểm ảnh. Nói một cách đơn giản, ta đi tạo một bảng màu mới (mapping giữa intensity của ảnh cũ và ảnh mới) sao cho khoảng màu mới rộng hơn khoảng màu cũ.  
Các điểm ảnh có cùng intensity trên ảnh cũ -> trên ảnh mới cũng có cùng intensity  
Từ equalization mang nghĩa sự chia đều, làm cho bằng nhau. Vậy histogram equalization có nghĩa là làm cho histogram đồng đều, bằng nhau. Cụ thể hơn, chính là làm cho histogram cho hình dáng về gần một đường ngang nhất có thể (làm số pixels của mỗi một intensity gần bằng nhau).

Ví dụ, với ảnh 8 bit (0->255) có độ phân giải 20x20, 400 pixels đó của ảnh chỉ tập trung trong khoảng từ 120->150, histogram equalization sẽ tìm cách **phân đều** 400 pixels đó sao cho **cường độ sáng nhỏ nhất là 0 và lớn nhất là 255**

## **Công thức:**

**B1**: Thống kê số lượng pixel cho từng mức sáng, ta được histogram *H*(*i*)

**B2**: Tính "hàm tích lũy" *Z* cho từng mức sáng theo công thức:

Trong đó chính là ổng số pixel có giá trị  nhỏ hơn hoặc bằng *i*

**B3**: Hàm biến đổi K tại một mức sáng i được tính như sau:

Công thức này có tác dụng **dãn** các khoảng phân bố dày đặc pixel và **co** các khoảng phân bố thưa pixel

# Chapter 3 Implementation

## Code:

import cv2

from matplotlib import pyplot as plt

import numpy as np

import os

# hàm read ảnh

def read\_images(path=None):

    images = []

    if path == None:

        path = os.getcwd()  #lấy đường dẫn ảnh

    for i in os.listdir(path):  #chạy từng đường dẫn ảnh

        if ".jpg" in i or ".png" in i or ".jpeg" in i:  #nếu đường dẫn thuộc các đuôi jpg, png, jpeg

            img = cv2.cvtColor(cv2.imread(os.path.join(path, i)), cv2.COLOR\_BGR2GRAY)

            images.append(img)  #lưu ảnh vào list images

    return images

#hàm biểu đồ thể hiện hiệu quả cải thiện ảnh

def plot\_hist(image, equalised, count):

    fig, axes = plt.subplots(2, 1)

    axes[0].hist(image.flatten(), 256, [0, 256])

    axes[0].set\_title("Before")

    axes[1].hist(equalised.flatten(), 256, [0, 256])

    axes[1].set\_title("After")

#hàm cải thiện độ tương phản ảnh

def hist\_equaliser(images):

    equalised\_image = []

    for i in images:

        equalised\_image.append(cv2.equalizeHist(i))

    return equalised\_image

#hàm show ảnh

def show\_images(images, equalised):

    count = 0

    for img1, img2 in zip(images, equalised):

        plot\_hist(img1, img2, count)

        cv2.imshow("Images", np.hstack((img1, img2))) # show ảnh củ và ảnh mới

        plt.show() # show biểu đồ

        cv2.waitKey(0)

if \_\_name\_\_ == '\_\_main\_\_':

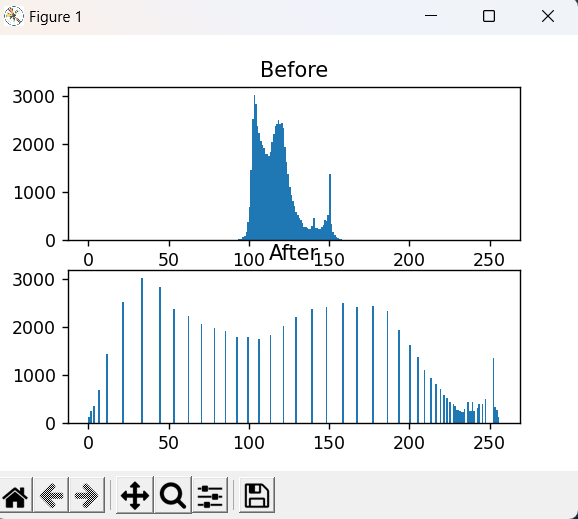
    images = read\_images()

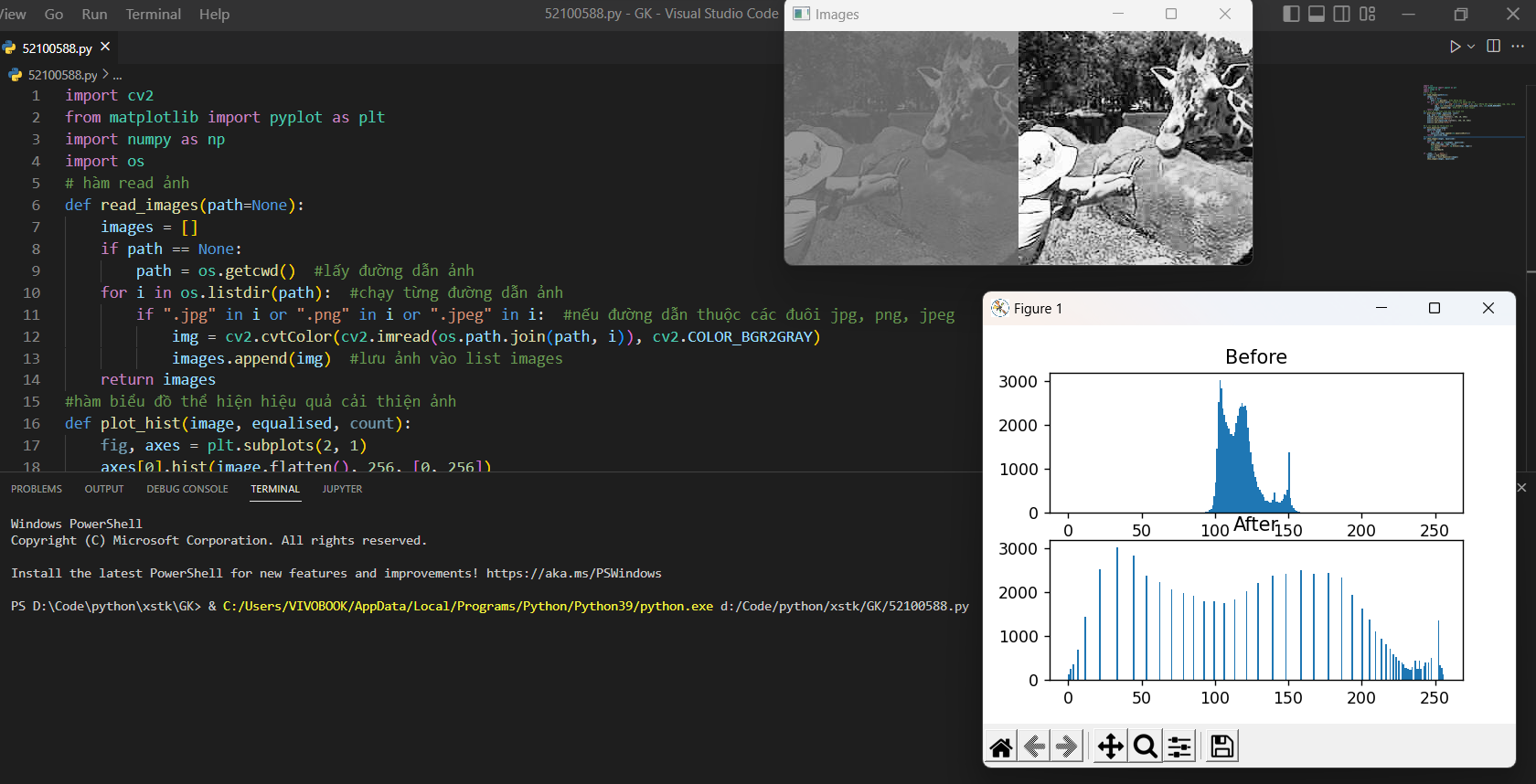
    equalised = hist\_equaliser(images)

    show\_images(images, equalised)

## Kết Quả Demo:

ảnh cũ ảnh mới

Biểu đồ trước và sau khi cân bằng sáng



# Tài liệu tham khảo

Tiếng việt

<https://blog.vietnamlab.vn/xu-ly-anh-voi-opencv-tut5-can-bang-histogram/#:~:text=Histogram%20equalization%20l%C3%A0%20m%E1%BB%99t%20k%C4%A9%20thu%E1%BA%ADt%20%C4%91%E1%BB%83%20t%C4%83ng,t%C6%B0%C6%A1ng%20ph%E1%BA%A3n%20%28constrast%29%20c%E1%BB%A7a%20%E1%BA%A3nh%20l%C3%A0%20g%C3%AC%20v%E1%BA%ADy>.

https://viblo.asia/p/xu-li-anh-thuat-toan-can-bang-histogram-anh-GrLZDOogKk0

Tiếng anh

<https://coderspacket.com/histogram-equalization-in-images-using-python-and-opencv#google_vignette>

https://en.wikipedia.org/wiki/Histogram\_equalization