TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**Báo cáo tiểu luân giữa kì**

**Môn giải tích ứng dụng**

*Người hướng dẫn*: **TS TRỊNH HÙNG CƯỜNG**

*Người thực hiện*: **TRƯƠNG ĐỨC ANH KHOA – 521H0459**

Lớp **: 21H50203**

Khoá  **: 25**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**Báo cáo tiểu luân giữa kì**

**Môn giải tích ứng dụng**

Người hướng dẫn: **TS TRỊNH HÙNG CƯỜNG**

Người thực hiện: **TRƯƠNG ĐỨC ANH KHOA**

Lớp **: 21H50203**

Khoá  **: 25**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2014**

LỜI CẢM ƠN

Em xin gửi lời cảm ơn đến thầy Lê Trung Nghĩa ở lớp lí thuyết và thầy Trịnh Hùng Cường ở lớp thực hành đã dạy cho em những kiến thức giúp em hoàn thành bài tiểu luận này.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi và được sự hướng dẫn của TS Trịnh Hùng Cường;. 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 đồ án 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 đồ án 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 10, tháng 12, năm 2021*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Text

Description automatically generated*

*Trương Đức Anh Khoa*

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

Trình bày tóm tắt về chức năng của 3 thư viện trong Python: Numpy, Sympy, Matplotlib. Nêu 1 số ví dụ về các hàm có trong thư viện.

MỤC LỤC

[CHƯƠNG 1 – PHẦN GIỚI THIỆU 2](#_Toc90204496)

[*1.1* *Chức năng của 3 thư viện: Numpy, Sympy, Matplotlib:* 2](#_Toc90204497)

[*1.1.1* *Thư viện Numpy:* 2](#_Toc90204498)

[*1.1.2* *Thư viện Sympy:* 2](#_Toc90204499)

[*1.1.3* *Thư viện Matplotlib:* 3](#_Toc90204500)

[*1.2* *Nội dung của chương này:* 3](#_Toc90204501)

[*1.2.1* *Thư viện Numpy:* 3](#_Toc90204502)

[*1.2.2* *Thư viện Sympy:* 10](#_Toc90204503)

[*1.2.3* *Thư viện Matplotlib:* 15](#_Toc90204504)

[CHƯƠNG 2 – PHẦN MÔ TẢ MÃ NGUỒN 30](#_Toc90204505)

[*2.1 Mô tả mã nguồn câu 1:* 31](#_Toc90204506)

[*2.2 Mô tả mã nguồn câu 2:* 31](#_Toc90204507)

[*2.3 Mô tả mã nguồn câu 3:* 31](#_Toc90204508)

[CHƯƠNG 3 – KẾT QUẢ 31](#_Toc90204509)

[*3.1 Câu 1:* 31](#_Toc90204510)

[*3.2 Câu 2:* 31](#_Toc90204511)

[*3.3 Câu 3:* 31](#_Toc90204512)

CHƯƠNG 1 – PHẦN GIỚI THIỆU

* 1. *Chức năng của 3 thư viện: Numpy, Sympy, Matplotlib:*

Các thông tin bên dưới được lấy từ nhiều nguồn tham khảo.

* + 1. *Thư viện Numpy:*
* Numpy hay còn được gọi là Numerical Python (Python số). Nó có nghĩa là các mô-đun xử lý số. NumPy là thư viện mã nguồn mở để tạo và quản lý các mảng và ma trận đa chiều. Thư viện này bao gồm một loạt các chức năng để xử lý các mảng phức tạp như vậy.
* Thư viện Numpy chuyên cung cấp các hàm toán học cấp cao để quản lý các mảng đa chiều. Bằng cách truy xuất các mô-đun từ NumPy, ta sẽ hoàn thành các tính toán đúng đắn và chính xác. Hơn thế nữa ta sẽ cải thiện đáng kể việc sử dụng Python với các cấu trúc dữ liệu này.
  + 1. *Thư viện Sympy:*
* Sympy là một thư viện Python để tính toán biểu tượng. Các mục tiêu đã nêu của thư viện là trở thành một hệ thống đại số máy tính đầy đủ tính năng và giữ một cơ sở mã đơn giản để thúc đẩy khả năng mở rộng và dễ hiểu. Sympy được viết hoàn toàn bằng Python và không phụ thuộc vào bất kỳ thư viện bổ sung nào. Nó có sẵn cả dưới dạng ứng dụng trên máy tính để bàn và trên web dưới dạng Sympy Live. Sympy bao gồm các tính năng từ số học biểu tượng cơ bản đến tính toán, đại số, toán học rời rạc và vật lý lượng tử. Nó có khả năng định dạng kết quả của các tính toán dưới dạng mã LaTeX.
* Tính năng Thư viện SymPy được chia thành một lõi với nhiều mô-đun tùy chọn. Hiện tại, lõi của SymPy có khoảng 13.000 dòng mã (bao gồm cả nhận xét và tài liệu) và các khả năng của nó bao gồm: Khả năng cơ bản Số học cơ bản : , /, +, -, \* Các hàm mở rộng đơn giản hóa: lượng giác, hyperbolic, hàm mũ, gốc, logarit, giá trị tuyệt đối, hài bậc cầu, hàm nhân tử và hàm gamma, hàm zeta, đa thức, siêu âm, hàm đặc biệt,…
  + 1. *Thư viện Matplotlib:*
* [Matplotlib là một thư viện Python](https://codelearn.io/sharing/ve-bieu-do-voi-thu-vien-matplotlib-p1) sử dụng Python Script để giúp chúng ta tạo ra các đồ thị 2D thường được ứng dụng trong toán học và khoa học dữ liệu. Thư viện này có hỗ trợ tạo nhiều giao điểm giữa hai trục số trong cùng một lúc. Bên cạnh đó, chúng ta cũng có thể dùng Matplotlib để thao tác trực tiếp đến các đặc điểm khác nhau của đồ thị.
* Matplotlib có thể tạo ra những đồ thị chất lượng và xuất ra một cách dễ dàng và thuận tiện, hoàn toàn đáp ứng nhu cầu của mọi ngành học. Các đồ thị được tạo ra bằng Matplotlib có sẵn bản sao cứng trên các nền tảng tương tác khác nhau.
* Ngoài ra, bạn còn có thể theo dõi bất kỳ lỗi nào phát sinh trong quá trình coding, các bản vá mới, đồng thời còn có thể đóng góp các tính năng mới tại GitHub. Đó là một trang chính thức để nêu ra các vấn đề liên quan đến Matplotlib và cùng giải quyết chúng.
  1. *Nội dung của chương này:*
     1. *Thư viện Numpy:*
        1. *Hàm np.array:*
* Hàm này được sử dụng để **tạo** một mảng với list giá trị được chỉ định trong các đối số của nó
* Ta có thể tạo mảng như sau:

#khai báo thư viện Numpy với bí danh là np  
import numpy as np  
#tạo mảng có giá trị từ 1 tới 5 với hàm array  
a = np.array([1, 2, 3, 4, 5])  
#in ra mảng  
print(a)

Kết quả đầu ra:



* Vì trong Numpy bắt buộc các phần tử phải cùng kiểu dữ liệu, nên nếu khác thì nó sẽ ép sao cho các phần tử đó cùng kiểu. Như ví ở dưới thì nó sẽ ép tất cả các phần tử thành kiểu số thực để đồng bộ với nhau:

# khai báo thư viện Numpy với bí danh là np  
import numpy as np  
# tạo mảng có 1 giá trị là số thực, còn lại là số nguyên  
a = np.array([1, 2, 3.2, 4, 5])  
# in ra mảng  
print(a)

Kết quả đầu ra:



* Ta còn có thể tạo mảng đa chiều với nhiều kiểu khác nhau như điền sẵn các phần tử trong chương trình:

# khai báo thư viện Numpy với bí danh là np  
import numpy as np  
# tạo mảng đa chiều với các phần tử được điền sẵn  
a = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])  
# in ra mảng  
print(a)

Kết quả đầu ra:

A screenshot of a video game

Description automatically generated with low confidence

* Tạo mảng đa chiều theo công thức:

# khai báo thư viện Numpy với bí danh là np  
import numpy as np  
# tạo mảng đa chiều với hàm range và vòng for  
a = np.array([range(i, i + 5) for i in [1, 2, 3, 4]])  
# in ra mảng  
print(a)

Kết quả đầu ra:

A picture containing text, electronics, keyboard

Description automatically generated

* + - 1. *Hàm np.arange:* Đây là hàm tạo mảng dựa trên phạm vi số mà ta đưa vào. Các giá trị trong mảng được tạo ra đều cách đều nhau bằng 1 giá trị nhất định. Ta có thể tạo mảng như thế với 4 yếu tố:
* Giá trị bắt đầu: Mảng sẽ bắt đầu với giá trị này, nếu không nhập thì nó mặc định bằng 0
* Giá trị kết thúc: Mảng sẽ kết thúc tại giá trị này từ đi 1. Đây là tham số bắt buộc phải nhập
* Bước nhảy: Đây là số xác định khoảng cách giữa 2 phần tử kế nhau, nếu không nhập thì mặc định sẽ là 1
* Kiểu dữ liệu(dtype = …): Xác định kiểu dữ liệu của tất cả các phần tử có trong mảng, nếu không nhập thì mặc định là none
* Các ví dụ cho hàm:

# khai báo thư viện Numpy với bí danh là np  
import numpy as np  
# tạo mảng cơ bản với các tham số và kiểu dữ liệu được nhập sẵn  
a = np.arange(1, 10, 2, dtype = int)  
# in ra mảng  
print(a)

Kết quả đầu ra:



Ta có thể kết hợp nó với các hàm khác:

# khai báo thư viện Numpy với bí danh là np  
import numpy as np  
# tạo mảng cơ bản với các tham số và kiểu dữ liệu được nhập sẵn  
a = np.arange(1, 10, 2, dtype = int)  
# in ra mảng  
print(a)  
# in ra mảng với các phần tử được bình phương lên  
print(np.square(a))

Kết quả đầu ra:

A picture containing text, watch

Description automatically generated

# khai báo thư viện Numpy với bí danh là np  
import numpy as np  
# tạo mảng cơ bản với các tham số và kiểu dữ liệu được nhập sẵn  
a = np.arange(-10, 10, 2, dtype = int)  
# in ra mảng  
print(a)  
# in ra mảng với các phần tử được lấy giá trị tuyệt đối  
print(np.abs(a))

Kết quả đầu ra:

A picture containing text, gauge, watch

Description automatically generated

* + - 1. *Hàm np.linspace:* Hàm này tương tự như hàm np.arange, nhưng ở đây ta không được điều chỉnh khoảng cách giữa các phần tử. Thay vào đó ta sẽ nhập tham số bắt đầu và kết thúc cùng với số lượng phần tử, sau đó hàm sẽ tự động đưa ra các phần tử có khoảng cách tự động cách đều nhau và có kiểu dữ liệu là số thực:

# khai báo thư viện Numpy với bí danh là np  
import numpy as np  
# tạo mảng cơ bản với các giá trị đầu và cuối cùng số lượng phần tử  
a = np.linspace(-10, 10, 11)  
# in ra mảng  
print(a)

Kết quả đầu ra:



* Hàm nà cũng có thể kết hợp với các hàm khác tương tự như hàm arrange:

# khai báo thư viện Numpy với bí danh là np  
import numpy as np  
# tạo mảng với các tham số và kiểu dữ liệu được nhập sẵn  
a = np.linspace(-10, 10, 21)  
# in ra mảng  
print(a)  
# in ra mảng với các phần tử bị lật ngược lại  
print(np.flip(a))

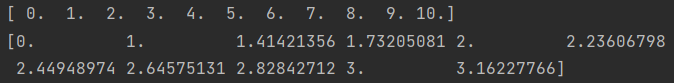
Kết quả đầu ra:

Graphical user interface, application

Description automatically generated

# khai báo thư viện Numpy với bí danh là np  
import numpy as np  
# tạo mảng cơ bản với các tham số và kiểu dữ liệu được nhập sẵn  
a = np.linspace(0, 10, 11)  
# in ra mảng  
print(a)  
# in ra mảng có giá trị là căn bậc 2 của các phần tử a  
print(np.sqrt(a))

Kết quả đầu ra:



* + - 1. *Hàm np.view:* Hàm này sẽ tạo ra 1 mảng bản sao của mảng trước đỏ. Các dữ liệu mà hàm view đã thực hiện sẽ ảnh hưởng đến mảng gốc và ngược lại

# khai báo thư viện Numpy với bí danh là np  
import numpy as np  
# tạo mảng cơ bản  
a = np.array([1, 2, 3, 4, 5, 6, 7])  
  
b = a.view()  
# thay đổi giá trị bên trong mảng b  
b[2] = 39  
# in ra mảng gốc và mảng sao chép  
print(a)  
print(b)

Kết quả đầu ra:

A picture containing text, gauge, watch

Description automatically generated

# khai báo thư viện Numpy với bí danh là np  
import numpy as np  
# tạo mảng cơ bản  
a = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])  
b = a.view()  
# thay đổi giá trị của cả 2 mảng  
a[2] = 13  
b[6] = 21  
# in ra mảng gốc và mảng sao chép  
print(a)  
print(b)

Kết quả đầu ra:

Graphical user interface

Description automatically generated with medium confidence

# khai báo thư viện Numpy với bí danh là np  
import numpy as np  
# tạo mảng cơ bản  
a = np.array([[1, 2, 3, 4, 5, 6], [7, 8, 9, 10, 11, 12]])  
  
b = a.view()  
# in ra mảng gốc và mảng sao chép  
print(a)  
print(b)

Kết quả đầu ra:

A picture containing text, electronics, keyboard

Description automatically generated

* + - 1. *Hàm np.soft:* Hàm này được sử dụng để sắp xếp mảng, nó trả về một bản sao của mảng đã sắp xếp có cùng hình dạng và kiểu. Thuật toán sắp xếp mặc định được sử dụng trong hàm này là sắp xếp nhanh, có thể thay đổi tùy theo nhu cầu bằng cách sử dụng kind = 'tên cách sắp xếp' chẳng hạn. kind = 'mergesort' sẽ sử dụng sắp xếp hợp nhất. **Cú pháp của các hàm sắp xếp: numpy.sort (arr, axis = - 1, kind = None, order = None)**
* Arr: mảng được sắp xếp
* Axis: trục chúng ta cần sắp xếp
* Kind: thuật toán sắp xếp
* Order: những trường hợp nào cần so sánh đầu tiên

# khai báo thư viện Numpy với bí danh là np  
import numpy as np  
# tạo mảng cơ bản  
arr1 = np.array([[6, 2, 4],  
 [5, 7, 3]])  
# sắp xếp theo cột  
arr2 = np.sort(arr1, axis = 0)  
# in mảng đã sắp xếp  
print(arr2)

Kết quả đầu ra:

Text

Description automatically generated

# khai báo thư viện Numpy với bí danh là np  
import numpy as np  
# tạo mảng cơ bản  
arr1 = np.array([[6, 2, 4],  
 [5, 7, 3]])  
# sắp xếp theo hàng với cách xếp là sắp xếp hợp nhất  
arr2 = np.sort(arr1, axis = 1, kind = 'mergesort')  
# in mảng đã sắp xếp  
print(arr2)

Kết quả đầu ra:

Text

Description automatically generated

# khai báo thư viện Numpy với bí danh là np  
import numpy as np  
# tạo mảng cơ bản  
arr1 = np.array([[6, 2, 4],  
 [5, 7, 3]])  
# sắp xếp theo cách mà lần trước đã dùng (-1) với cách xếp là sắp xếp nhanh

arr2 = np.sort(arr1, axis = -1, kind = 'quicksort')  
# in mảng đã sắp xếp  
print(arr2)

Kết quả đầu ra:

Text

Description automatically generated

* + 1. *Thư viện Sympy:*
       1. *Hàm Symbols:*
* Đây là 1 trong những hàm quan trọng nhất trong thư viện Sympy. Nó được gọi là hàm biểu tượng. Các phép tính toán trong thư viện Sympy như đã nói đều dùng đến biểu tượng:

import sympy as sp  
# khai báo thư viện Sympy với bí danh sp  
x = sp.Symbol('x')  
# tạo biến x là biểu tượng x  
fx = x\*\*2 + 2\*x + 9  
# tạo 1 biểu thức với biểu tượng x  
fx  
# in ra biểu thức

Kết quả đầu ra:

* Ta còn có thể tạo nhiều biến với hàm này:

import sympy as sp  
# khai báo thư viện Sympy với bí danh sp  
x = sp.Symbol('x')  
# tạo biến x là biểu tượng x  
y = sp.Symbol('y')  
# tạo biến y là biểu tượng y  
a = 2\*x + 3\*y - 6  
# tạo 1 biểu thức với biểu tượng x và y  
print(a)  
# in ra biểu thức

Kết quả đầu ra:



Hoặc gộp lại thành 1 câu như thế này:

import sympy as sp  
# khai báo thư viện Sympy với bí danh sp  
x, y = sp.symbols('x, y')  
a = 2\*x + 3\*y - 6  
# tạo 1 biểu thức với biểu tượng x và y  
print(a)  
# in ra biểu thức

* Lưu ý: giữa tạo 1 biến biểu tượng với tạo nhiều biến biểu tượng 1 có lệnh khác nhau đôi chút nên ta phải thông nhất sử dụng 1 lệnh.
  + - 1. *Hàm Solve:*
* Hàm này hỗ trợ chúng ta giải các phương trình từ đơn giản đến phức tạp với tốc độ nhanh chóng vào trả về giá trị chính xác:

import sympy as sp  
  
x = sp.symbols('x')  
y = x \*\* 2 - 36  
  
print("phương trình y = {}".format(y))  
  
a = sp.solve(y)  
  
print("Nghiệm của phương trình : {}".format(a))

Kết quả đầu ra:

Text

Description automatically generated

* Ta còn có thể phối hợp nó với hàm solver (Eq()) để tính các phương trình khác:

import sympy as sp  
  
x = sp.symbols('x')  
a = sp.Eq(x\*\*2 - 4, 0)  
  
ng = sp.solve(a)  
print(ng)

Kết quả đầu ra:



* Ta còn có thể tính các phương trình nâng cao như hệ phương trình:

import sympy as sp  
  
x, y = sp.symbols('x y')  
a1 = sp.Eq(2 \* x - 3 \* y - 9, 0)  
a2 = sp.Eq(3 \* x + 4 \* y + 6, 0)  
  
ng = sp.solve((a1, a2), (x, y))  
print("x =", ng[x])  
print("y =", ng[y])

Kết quả đầu ra:

Text

Description automatically generated

* + - 1. *Hàm Sqrt:*
* Đây là 1 hàm tính toán khá phổ biến trong toán học với kết quả trả về là căn bậc 2 của tham số nhận vào. Khác với căn bậc 2 trong thư viện Math là sẽ trả về giá trị tương đối chính xác thì với căn bậc 2 trong Sympy, nó sẽ trả về 1 biểu thức để ta dễ dàng quan sát và hiểu hơn.

import sympy as sp  
import math  
  
a1 = math.sqrt(16)  
a2 = sp.sqrt(16)  
  
print("Với math:", a1)  
print("Với Sympy:", a2)

Kết quả đầu ra của cả 2 xét về giá trị thì bằng nhau, nhưng về kiểu dữ liệu thì lại khác nhau:

Text

Description automatically generated

import sympy as sp  
import math  
  
a1 = math.sqrt(8)  
a2 = sp.sqrt(8)  
  
print("Với math:", a1)  
print("Với Sympy:", a2)

Với giá trị khác, ta sẽ thấy khác nhau rõ ràng:

Text

Description automatically generated

* + - 1. *Hàm Diff:*
* Cũng như hàm sqrt(), hàm diff cũng chiếm phần quan trọng trong việc tính toán với việc tìm và tính đạo hàm của 1 hàm số. Ví dụ:

import sympy as sp  
  
x = sp.Symbol('x')  
f = 2\*x\*\*2 + 3  
df = f.diff(x) # tìm đạo hàm của f(x)  
  
print(df)

Kết quả đầu ra: x\*4

import sympy as sp  
  
x = sp.symbols('x')  
f = (x + 1) / (x \*\* 2 + 1)  
df = f.diff(x)  
  
print(df)

Kết quả đầu ra:



Ta còn có thể tìm đạo hàm riêng của mỗi biến:

import sympy as sp  
  
x, y = sp.symbols('x, y')  
f = 2\*x + 3\*y  
dfx = f.diff(x) # tìm đạo hàm riêng đối với x của f(x)  
dfy = f.diff(y) # tìm đạo hàm riêng đối với y của f(x)  
  
print("df/dx =", dfx)  
print("df/dy =", dfy)

Kết quả đầu ra:

Text

Description automatically generated

* + - 1. *Hàm Subs:*
* Đây là hàm dùng để thay các tham số đầu được nhận vào bằng tham số thứ 2 vào biểu thức hoặc phương trình.Ta có thể thay giữa các biến với nhau, hoặc thay biến trong biểu thức thành các số để tính ra kết quả. Ví dụ:

import sympy as sp  
  
x = sp.symbols('x')  
y = sp.symbols('y')  
  
f = (x + 1) / (x \*\* 2 + 1)  
print("Trước khi thay:", f)  
f1 = f.subs(x, y)  
print("Sau khi thay:", f1)

Kết quả đầu ra:

Text

Description automatically generated

import sympy as sp  
  
x = sp.symbols('x')  
  
f = (x + 1) / (x \*\* 2 + 1)  
f1 = f.subs(x, 5) # thay x bằng 5  
print(f1)

Kết quả đầu ra: 3/13

import sympy as sp  
  
A = 459  
  
x = sp.symbols('x') # tạo biến x là 1 hàm biểu tượng  
fx = A \* x\*\*2 - 2 \* A \* x + A  
d\_fx = sp.diff(fx, x) # đạo hàm phương trình fx  
slope = d\_fx.subs(x, 2) # tính đạo hàm của fx khi thế x vào phương trình  
  
print(slope)

Kết quả đầu ra: 918

* + 1. *Thư viện Matplotlib:*
       1. *Hàm Plot:* Đây là hàm khá quan trọng cho việc vẽ đồ thị. Hàm sẽ vẽ vẽ cho ta không những các điểm, mà còn vẽ cho ta đồ thị của các phương trình mà sẽ rất khó khi vẽ bằng tay
* Tham số đầu tiên là các giá trị của trục hoành
* Tham số thứ hai sẽ là các giá trị của trục tung

Ví dụ:

import matplotlib.pyplot as plt  
import numpy as np  
x = np.array([0, 9])  
y = np.array([0, 8])  
  
plt.plot(x, y)  
  
plt.show()

Kết quả đầu ra:

Chart, line chart

Description automatically generated

Ta còn có thể thêm các chú thích cũng như đổi màu cho dồ thị để dễ dàng quan sát, khi đó ta dùng thêm hàm .legend() để hiển thị chú thích:

import numpy as np #khai báo thư viện   
import sympy as sp #khai báo thư viện   
import matplotlib.pyplot as plt #khai báo thư viện   
  
A = 459  
B = 1  
x = sp.symbols('x') #tạo biến x là 1 hàm biểu tượng  
fx = A \* x\*\*2 - 2 \* A \* x + A #khai báo phương trình fx  
d\_fx = sp.diff(fx, x) #đạo hàm phương trình fx  
slope = d\_fx.subs(x, 2) #tính đạo hàm của fx khi thế x vào phương trình  
  
fx = lambda x: A \* x\*\*2 - 2 \* A \* x + A #định nghĩa fx bằng hàm ẩn danh  
fx1 = lambda x: slope \* (x - 2) + A #định nghĩa phương trình tiếp tuyến của fx bằng hàm ẩn danh  
  
x = np.arange(-5, 5, 0.1, dtype=float) #Khởi tạo mảng của x từ -5 tới 5 với bước nhảy là 0.1  
plt.plot(x, fx(x), color = 'b', label="y = f(x)") #vẽ đồ thị phương trình fx  
plt.plot(x, fx1(x), color = 'g', label="y = f'(x)(x-2)+A") #vẽ đồ thị tiếp tuyến của fx  
  
plt.title("câu 3a") #in tiêu đề  
plt.legend() #hiện chú thích  
plt.grid() #vẽ lưới cho đồ thị  
plt.show() #hiển thị đồ thị

Kết quả đầu ra:

Chart, line chart

Description automatically generated

1 Ví dụ khác:

import numpy as np  
import matplotlib.pyplot as plt  
  
def f1(x):  
 return np.sqrt(1 - (abs(x) - 1)\*\*2)  
  
def f2(x):  
 return -3 \* np.sqrt(1 - np.sqrt(abs(x)/2))  
  
x = np.arange(-2, 2.1, 0.2)  
plt.plot(x, f1(x), color = 'm', label="f1(x)")  
plt.plot(x, f2(x), color = 'r', label="f2(x)")  
plt.legend()  
plt.grid()  
plt.show()

Kết quả đầu ra:

Chart, line chart

Description automatically generated

* + - 1. *Hàm Label và Title:* Đây là các hàm hỗ trợ tạo nhãn cho đồ thị, toạ độ trục hoành, trục tung bằng hàm .title(), .xlabel(), .ylabel()

import matplotlib.pyplot as plt  
import numpy as np  
x = np.array([0, 9])  
y = np.array([0, 8])  
  
plt.plot(x, y)  
plt.title("Tên đồ thị")  
plt.xlabel("Trục hoành")  
plt.ylabel("Trục tung")  
plt.show()

Kết quả đầu ra:

Chart, line chart

Description automatically generated

Ta có thể thay đổi phông chữ cùng cỡ chữ bằng cách lưu các cài đặt có sẵn vào 1 biến sau đó gọi biến đó với fontdict trong hàm. Ta còn có thể điều chỉnh vị trí xuất hiện với từ khoá loc:

import matplotlib.pyplot as plt  
import numpy as np  
x = np.array([0, 9])  
y = np.array([0, 8])  
  
font1 = {'family': 'serif', 'color': 'r', 'size': 30}  
font2 = {'family': 'serif', 'color': 'b', 'size': 15}  
font3 = {'family': 'serif', 'color': 'g', 'size': 15}  
  
plt.plot(x, y)  
plt.title("Tên đồ thị", fontdict=font1, loc='center')  
plt.xlabel("Trục hoành", fontdict=font2, loc='right')  
plt.ylabel("Trục tung", fontdict=font3, loc='top')  
plt.show()

Kết quả đầu ra:

Chart, line chart

Description automatically generated

import matplotlib.pyplot as plt  
import numpy as np  
def fx(x):  
 return x\*\*2 - 4\*x + 4  
x = np.arange(-5, 6, 1)  
y = list(map(fx, x))  
  
font1 = {'family': 'serif', 'color': 'r', 'size': 30}  
font2 = {'family': 'serif', 'color': 'b', 'size': 15}  
font3 = {'family': 'serif', 'color': 'g', 'size': 15}  
  
plt.plot(x, y, color='k')  
plt.title("Tên đồ thị", fontdict=font1, loc='left')  
plt.xlabel("Trục hoành", fontdict=font2, loc='left')  
plt.ylabel("Trục tung", fontdict=font3, loc='bottom')  
plt.show()

Kết quả đầu ra:

Chart

Description automatically generated

* + - 1. *Hàm Stackplot:* Đây là hàm hỗ trợ cho việc vẽ đồ thị dạng miền với tổng các giá trị trong 1 mốc sẽ bằng với tổng các giá trị ở mốc khác. Ta cần các tham số như sau để vẽ các đồ thị: plt.stackplot(S, A, B, [color = ‘r’, ‘b’])
* S: Giá trị của trục hoành. Nên để theo giá trị tăng dần
* A, B: đây là các thông số sẽ hiện lên trên dồ thị. Các giá trị phải theo quy luật như A[0] + B[0] = A[1] + B[1] = … để đồ thị được chuẩn xác hơn
* Color: điều chỉnh màu của các miền. Nếu không điền thì màu sẽ xuất hiện ngẫu nhiên

import matplotlib.pyplot as plt  
  
s = [1, 2, 3, 4, 5]  
  
a = [5, 7, 8, 6, 7]  
b = [2, 3, 1, 3, 2]  
c = [7, 6, 2, 5, 2]  
d = [6, 4, 9, 6, 9]  
  
plt.stackplot(s, a, b, c, d, colors=['b', 'c', 'r', 'k'])  
  
plt.xlabel('x')  
plt.ylabel('y')  
plt.title('Stack Plot')  
plt.show()

Kết quả đầu ra:

Chart

Description automatically generated

import matplotlib.pyplot as plt  
  
days = [1, 2, 3, 4, 5]  
  
sleeping = [7, 8, 6, 11, 7]  
eating = [2, 3, 4, 3, 2]  
working = [7, 8, 7, 2, 2]  
playing = [7, 4, 6, 7, 12]  
  
plt.stackplot(days, sleeping, eating, working, playing, colors=['m', 'c', 'r', 'k'])  
  
plt.xlabel('x')  
plt.ylabel('y')  
plt.title('Stack Plot')  
plt.legend()  
plt.show()

Kết quả đầu ra:

Chart

Description automatically generated

* Ta còn có thể sử dụng hàm plot để thêm các chú thích cho đồ thị của mình:

import matplotlib.pyplot as plt  
  
days = [1, 2, 3, 4, 5]  
  
sleeping = [7, 8, 6, 11, 7]  
eating = [2, 3, 4, 3, 2]  
working = [7, 8, 7, 2, 2]  
playing = [7, 4, 6, 7, 12]  
  
plt.stackplot(days, sleeping, eating, working, playing, colors=['b', 'c', 'r', 'k'])  
  
plt.plot([], [], color='k', label='Playing', linewidth=5)  
plt.plot([], [], color='r', label='Working', linewidth=5)  
plt.plot([], [], color='c', label='Eating', linewidth=5)  
plt.plot([], [], color='b', label='Sleeping', linewidth=5)  
  
plt.xlabel('x')  
plt.ylabel('y')  
plt.title('Stack Plot')  
plt.legend()  
plt.show()

Kết quả đầu ra:

Chart

Description automatically generated

* + - 1. *Hàm Grid:* Hàm này hỗ trợ chúng ta về việc vẽ các đường lưới ở dạng bất kì chúng ta muốn. Ví dụ:

import matplotlib.pyplot as plt  
import numpy as np  
x = np.array([0, 10])  
y = np.array([0, 10])  
  
plt.plot(x, y)  
plt.grid()  
plt.show()

Kết quả đầu ra:

Chart, line chart

Description automatically generated

Hoặc đổi thành các đường nét đứt với từ khoá linestyle:

import matplotlib.pyplot as plt  
import numpy as np  
x = np.array([0, 10])  
y = np.array([0, 10])  
  
plt.plot(x, y)  
plt.grid(linestyle="--")  
plt.show()

Kết quả đầu ra:

Chart, line chart

Description automatically generated

Hay đơn giản chỉ muốn xuất hiện các đường ở trục hoành hoặc trục tung:

import matplotlib.pyplot as plt  
import numpy as np  
x = np.array([0, 10])  
y = np.array([0, 10])  
  
plt.plot(x, y)  
plt.grid(axis='x')  
plt.show()

Kết quả đầu ra:

Chart

Description automatically generated

* + - 1. *Hàm Legend:* đây là hàm dùng để hiển thị chú thích cho các đồ thị. Ta có thể điều chỉnh chú thích đó theo nhiều hướng:
* Điều chỉnh độ trong suốt của khung chú thích bằng framealpha với 2 giá trị đưa vào là 0(Trong suốt) và 1(Không trong suốt):

import numpy as np  
import matplotlib.pyplot as plt  
  
def f1(x):  
 return np.sqrt(1 - (abs(x) - 1)\*\*2)  
  
def f2(x):  
 return -3 \* np.sqrt(1 - np.sqrt(abs(x)/2))  
  
x = np.arange(-2, 2.1, 0.2)  
plt.plot(x, f1(x), color = 'm', label="f1(x)")  
plt.plot(x, f2(x), color = 'r', label="f2(x)")  
plt.legend(framealpha=0)  
plt.grid()  
plt.show()

Kết quả đầu ra:

Chart, line chart

Description automatically generated

* Vị trí của khung chú thích trong biểu đồ với từ khoá loc cùng các giá trị như 'best', 'center', 'left center', 'right center', 'lower center', 'lower left', ...

import numpy as np  
import matplotlib.pyplot as plt  
  
def f1(x):  
 return np.sqrt(1 - (abs(x) - 1)\*\*2)  
  
def f2(x):  
 return -3 \* np.sqrt(1 - np.sqrt(abs(x)/2))  
  
x = np.arange(-2, 2.1, 0.2)  
plt.plot(x, f1(x), color = 'm', label="f1(x)")  
plt.plot(x, f2(x), color = 'r', label="f2(x)")  
plt.legend(loc='center')  
plt.grid()  
plt.show()

Kết quả đầu ra:

Chart, line chart

Description automatically generated

* Viền có đổ bóng hay không với shadow(True hoặc False):

import numpy as np  
import matplotlib.pyplot as plt  
  
def f1(x):  
 return np.sqrt(1 - (abs(x) - 1)\*\*2)  
  
def f2(x):  
 return -3 \* np.sqrt(1 - np.sqrt(abs(x)/2))  
  
x = np.arange(-2, 2.1, 0.2)  
plt.plot(x, f1(x), color = 'm', label="f1(x)")  
plt.plot(x, f2(x), color = 'r', label="f2(x)")  
plt.legend(shadow=True)  
plt.grid()  
plt.show()

Kết quả đầu ra:

Chart, line chart

Description automatically generated

CHƯƠNG 2 – PHẦN MÔ TẢ MÃ NGUỒN

*2.1 Mô tả mã nguồn câu 1:*

import numpy as np  *#khai báo thư viện*

import sympy as sp  *#khai báo thư viện*

import matplotlib.pyplot as plt  *#khai báo thư viện*

A = 459

B = 1

fx = **lambda** x: A \* x\*\*2 - 2 \* A \* x + A *#định nghĩa hàm f(x) bằng hàm ẩn danh*

gx = **lambda** x: -A \* x\*\*2 + A\*\*2 \* x + A *#định nghĩa hàm g(x) bằng hàm ẩn danh*

**def** cau\_1a():

    print("Câu 1a:")

    print("f(0) = " + str(fx(0))) *#giá thị của f(x) khi x bằng 0*

    print("f(1) = " + str(fx(1))) *#giá thị của f(x) khi x bằng 1*

    print("f(A) = " + str(fx(A))) *#giá thị của f(x) khi x bằng A*

**def** cau\_1b():

    print("Câu 1b:")

    print("g(0) = " + str(gx(0))) *#giá thị của g(x) khi x bằng 0*

    print("g(1) = " + str(gx(1))) *#giá thị của g(x) khi x bằng 1*

    print("g(A) = " + str(gx(A))) *#giá thị của g(x) khi x bằng A*

**def** cau\_1c():

    fx = **lambda** x: A \* x\*\*2 - 2 \* A \* x + A *#định nghĩa hàm f(x) bằng hàm ẩn danh*

    gx = **lambda** x: -A \* x\*\*2 + A\*\*2 \* x + A *#định nghĩa hàm g(x) bằng hàm ẩn danh*

    x = np.arange(-100, 300, dtype=np.float) *#Khởi tạo mảng của x từ -100 tới 300 với bước nhảy là 1*

    plt.plot(x, fx(x), color = 'b', label="fx") *#vẽ đồ thị của fx*

    plt.plot(x, gx(x), color = 'g', label="gx") *#vẽ đồ thị của gx*

    plt.title("Câu 1c") *#tiêu đề*

    plt.legend() *#hiện chú thích*

    plt.grid() *#vẽ lưới cho dồ thị*

    plt.show() *#hiển thị dồ thị*

**def** cau\_1d():

    x = np.arange(-10, 10.1, 0.1)

    fx = **lambda** x: A \* x\*\*2 - 2 \* A \* x + A *#định nghĩa hàm f(x) bằng hàm ẩn danh*

    y = list( map( fx, x)) *#tạo trục tung thêo từng giá trị x của fx*

    y\_keo\_dan = list( map( fx, x / (B + 2))) *#tạo trục tung thêo từng giá trị x của fx bị kéo dãn*

    plt.plot(x,y,label="Đồ thị ban đầu") *#vẽ đồ thị fx*

    plt.plot(x,y\_keo\_dan,label="Đồ thị kéo giãn") *#vẽ đồ thị fx bị kéo dãn*

    plt.title("câu 1d") *#hiện tiêu đề*

    plt.legend() *#hiện chú thích*

    plt.show() *#hiển thị dồ thị*

**def** cau\_1e():

    fx = **lambda** x: A \* x\*\*2 - 2 \* A \* x + A *#định nghĩa hàm f(x) bằng hàm ẩn danh*

    gx = **lambda** x: -A \* x\*\*2 + A\*\*2 \* x + A *#định nghĩa hàm g(x) bằng hàm ẩn danh*

    x = np.arange(-100, 300, 1, dtype=np.float) *#Khởi tạo mảng của x từ -100 tới 300 với bước nhảy là 1*

    plt.plot(x, fx(x), color = 'k', label="fx ") *#vẽ đồ thị của fx*

    plt.plot(x, gx(x), color = 'y', label="gx ") *#vẽ đồ thị của gx*

    x = sp.symbols('x') *#tạo biến x là 1 hàm biểu tượng*

    fx = A \* x\*\*2 - 2 \* A \* x + A *#khai báo phương trình fx*

    gx = -A \* x\*\*2 + A\*\*2 \* x + A *#khai báo phương trình gx*

    x\_root = sp.solve(fx - gx) *#giải phương trình fx - gx tìm nghiệm*

    y\_root0 = fx.subs(x, x\_root[0]) *#thay nghiệm x1 vào phương trình fx*

    y\_root1 = fx.subs(x, x\_root[1]) *#thay nghiệm x2 vào phương trình gx*

    print("Toạ độ X1:(",x\_root[0],",",y\_root0,")") *#in ra toạ độ của x1*

    print("Toạ độ X2:(",x\_root[1],",",y\_root1,")") *#in ra toạ độ của x2*

    plt.plot(x\_root[0], y\_root0, 'ro') *#vẽ giao điểm thứ nhất*

    plt.plot(x\_root[1], y\_root1, 'ro') *#vẽ giao điểm thứ hai*

    plt.title("Câu 1e") *#in tiêu đề*

    plt.legend() *#hiện chú thích*

    plt.grid() *#vẽ lưới cho đồ thị*

    plt.show() *#hiển thị đồ thị*

cau\_1a() *#gọi câu a*

print("\n") *#xuống dòng*

cau\_1b() *#gọi câu b*

print("\n") *#xuống dòng*

cau\_1c() *#gọi câu c*

cau\_1d() *#gọi câu d*

print("Câu e") *#in "Câu e"*

cau\_1e() *# gọi câu e*

*2.2 Mô tả mã nguồn câu 2:*

import numpy as np *#khai báo thư viện*

import sympy as sp *#khai báo thư viện*

A = 459

B = 1

**def** f(a): *#gọi hàm f(x)*

  if a == A:

    return A *#trả về giá trị A nếu x = A*

  else:

    return (x\*\*2 - A\*x - B\*x + A\*B) / (x - A) *#viết hàm trã về giá trị của fx khi thế x vào*

x = sp.symbols('x') *#tạo biến x là 1 hàm biểu tượng*

fx = (x\*\*2 - A\*x - B\*x + A\*B) / (x - A) *#khai báo phương trình fx bằng biểu thức bên*

lm\_fx = sp.limit(fx, x, 0) *#tính lim của fx khi x dần về 0*

lm\_fx1 = sp.limit(fx, x, B) *#tính lim của fx khi x dần về B*

lm\_fx2 = sp.limit(fx, x, A + B + 1) *#tính lim của fx khi x dần về A+B+1*

print("Cau a:") *#in "Cau a"*

print("Gioi han cua ham so khi x dan tien ve 0 la", lm\_fx) *#in ra giá trị của lim x->0*

print("Gioi han cua ham so khi x dan tien ve B la", lm\_fx1) *#in ra giá trị của lim x->B*

print("Gioi han cua ham so khi x dan tien ve A + B + 1 la", lm\_fx2) *#in ra giá trị của lim x->A+B+1*

print("\n") *#xuống dòng*

print("Cau b:") *#in"Cau b"*

fx = f(A) *#khai báo phương trình fx = A*

lm\_fA = sp.limit(fx, x, A) *#tính lim của fx khi x dần về A*

print("So sanh lim\_f(A) va f(A):", lm\_fA, A) *#so sánh lim fx và fx*

if lm\_fA == fx: *#xét điều kiện nếu lim của f(A) = a*

  print("Ham so lien tuc tai x=", A) *#in ra màn hình nếu điều kiện đúng*

else:

  print("Ham so khong lien tuc tai x =", A) *#in ra màn hình nếu điều kiện sai*

print("\n") *#xuống dòng*

print("Cau c:") *#in "Cau c"*

for n in np.arange(A - 10, A + 11, 1): *#chạy vòng lặp với n chạy từ A-10 tới A+10 với bước nhảy là 1*

  if n == A: *#nếu n = A thì thực hiện*

    fx = f(A) *#khai báo phương trình fx = A*

    lm\_f = sp.limit(fx, x, A) *#tính lim của fx khi x dần về A*

    if lm\_f == fx: *#nếu lim của fx khi x dần về A bằng với fx*

      print("c. Ham so lien tuc tai",A) *#in ra hàm số liên tục tại A*

    else:

      print("c. Ham so khong lien tuc tai",A) *#in ra hàm số không liên tục tại A*

  else:

    fx = f(n) *#gán hàm f(n) vào fx*

    lm\_f = sp.limit(fx, x, n) *#tính lim của fx khi dần về n*

    y = fx.subs(x, n) *# Thay biến x thành n vào trong fx rồi gán giá trị trả về thành y*

    if lm\_f == y: *#nếu lim của fx khi x dần về A bằng với y*

      print("c. Ham so lien tuc tai",n) *#in ra hàm số liên tục tại n*

    else:

      print("c. Ham so khong lien tuc tai",n) *#in ra hàm số không liên tục tại n*

*2.3 Mô tả mã nguồn câu 3:*

import numpy as np *#khai báo thư viện*

import sympy as sp *#khai báo thư viện*

import matplotlib.pyplot as plt *#khai báo thư viện*

A = 459

B = 1

**def** cau\_3a():

    x = sp.symbols('x') *#tạo biến x là 1 hàm biểu tượng*

    fx = A \* x\*\*2 - 2 \* A \* x + A *#khai báo phương trình fx*

    d\_fx = sp.diff(fx, x) *#đạo hàm phương trình fx*

    slope = d\_fx.subs(x, 2) *#tính đạo hàm của fx khi thế x vào phương trình*

    fx = **lambda** x: A \* x\*\*2 - 2 \* A \* x + A *#định nghĩa fx bằng hàm ẩn danh*

    fx1 = **lambda** x: slope \* (x - 2) + A *#định nghĩa phương trình tiếp tuyến của fx bằng hàm ẩn danh*

    x = np.arange(-5, 5, 0.1, dtype=np.float) *#Khởi tạo mảng của x từ -5 tới 5 với bước nhảy là 0.1*

    plt.plot(x, fx(x), color = 'b', label="y = f(x)") *#vẽ đồ thị phương trình fx*

    plt.plot(x, fx1(x), color = 'g', label="y = f'(x)(x-2)+A") *#vẽ đồ thị tiếp tuyến của fx*

    plt.title("câu 3a") *#in tiêu đề*

    plt.legend() *#hiện chú thích*

    plt.grid() *#vẽ lưới cho đồ thị*

    plt.show() *#hiển thị đồ thị*

cau\_3a() *#gọi câu 3a*

**def** cau\_3b():

    x = sp.symbols('x') *#tạo biến x là 1 hàm biểu tượng*

    fx = A \* x\*\*2 - 2 \* A \* x + A *#khai báo phương trình fx*

    d\_fx = sp.diff(fx, x) *#đạo hàm phương trình fx*

    ng = sp.solve(d\_fx\*(x - 0) + (-A) - fx) *#giải nghiệm phương trình (f'(x) \* (x - x0) + y0) - f(x)*

    ng1 = fx.subs(x, ng[0]) *#giá trị fx khi thế x1 vào phương trình*

    ng2 = fx.subs(x, ng[1]) *#giá trị fx khi thế x2 vào phương trình*

    d\_fx1 = d\_fx.subs(x, ng[0]) *#giá trị của f'x khi thế x1 vào phương trình*

    d\_fx2 = d\_fx.subs(x, ng[1]) *#giá trị của f'x khi thế x2 vào phương trình*

    fx = **lambda** x: A \* x\*\*2 - 2 \* A \* x + A *#định nghĩa phương trình fx bằng hàm ẩn danh*

    fx1 = **lambda** x: d\_fx1 \* (x - ng[0]) + ng1 *#định nghĩa phương trình tiếp tuyến thứ nhất bằng hàm ẩn danh*

    fx2 = **lambda** x: d\_fx2 \* (x - ng[1]) + ng2 *#định nghĩa phương trình tiếp tuyến thứ nhất bằng hàm ẩn danh*

    x = np.arange(-5, 5, 0.1, dtype=np.float) *#Khởi tạo mảng của x từ -5 tới 5 với bước nhảy là 0.1*

    plt.plot(x, fx(x), color = 'b', label="y = f(x)") *#vẽ đồ thị phương trình fx*

    plt.plot(x, fx1(x), color = 'r', label="y = f'({0})(x {0}) - {1}".format(ng[0],ng1)) *#vẽ đồ thị tiếp tuyến thứ nhất của fx*

    plt.plot(x, fx2(x), color = 'g', label="y = f'({0})(x {0}) - {1}".format(ng[1],ng2)) *#vẽ đồ thị tiếp tuyến thứ hai của fx*

    plt.title("câu 3b") *#in tiêu đề*

    plt.legend() *#hiện chú thích*

    plt.grid() *#vẽ lưới cho đồ thị*

    plt.show() *#hiển thị đồ thị*

cau\_3b() *#gọi câu 3b*

CHƯƠNG 3 – KẾT QUẢ

*3.1 Câu 1:*

*Text, letter

Description automatically generated*

*Chart, line chart

Description automatically generated*

*Chart

Description automatically generated*

*Chart, line chart

Description automatically generated*

*3.2 Câu 2:*

*Text

Description automatically generated*

*Text

Description automatically generated*

*Text

Description automatically generated*

*Text

Description automatically generated*

*3.3 Câu 3:*

*Chart, line chart

Description automatically generated*

*Chart, line chart

Description automatically generated*