THỰC HÀNH TOÁN CAO CẤP

TÀI LIỆU PHỤC VỤ SINH VIÊN NGÀNH KHOA HỌC DỮ LIỆU

Nhóm biên soạn: TS. Hoàng Lê Minh – Khưu Minh Cảnh – Hoàng Thị Kiều Anh – Lê Thị Ngọc Huyên – …

TP.HCM – Năm 2019

MỤC LỤC

CHƯƠNG 2: ĐẠO HÀM

Mục tiêu:

- *Cơ bản về Python và Sympy trong các ứng dụng tính toán: hàm eval, hàm subs;*
- *Danh sách (list) và vẽ đồ thị trong Python;*
- *Tính toán đạo hàm.*

Nội dung chính:

1. Một số vấn đề xử lý với Python và Sympy

Giới thiệu về một số vấn đề xử lý bổ sung với Python và gói Sympy. Các bổ trợ này sẽ hỗ trợ cho các tính toán, đặc biệt tính toán và xử lý hình thức.

1.1. Giới thiệu hàm eval trong Python

Hàm eval trong Python có chức năng ước tính một biểu thức số học cho một chuỗi. Như các dạng bảng tính Excel, biểu thức sẽ được tính toán theo các giá trị nhập. Ví dụ:

Thực hành 1: Sử dụng hàm eval

>>> **chuoitinhtoan** = "a*b+c"

>>> a = 2

>>> b = 5

>>> c = 8

>>> **eval**(chuoitinhtoan)

…….. ← Sinh viên điền giá trị vào

1.2. Giới thiệu hàm subs trong Sympy

Mạnh mẽ hơn hàm eval() trong Python, hàm **subs()** của Sympy không những vừa thay thế các biến để tính toán vừa có khả năng thực hiện tính toán hình thức. Chúng ta xét thực hành minh họa về hàm subs như sau:

Thực hành 2: Cơ bản về sử dụng hàm subs

>>> import sympy

>>> x = Symbol('x')

```
>>> y = Symbol('y')
>>> bieuthuc = x+y
>>> thaytheso = bieuthuc.subs({x:10, y:5})
>>> thaytheso
```

…… ← sinh viên điền kết quả vào

Rõ ràng đến đây, ta thấy được hàm **subs()** cũng tương tự hàm eval() khi tính toán. Và dưới đây là một ưu điểm khác của hàm **subs()** trong Sympy:

```
>>> u = Symbol('u')
>>> v = Symbol('v')
>>> bieuthuc_theo_uv = bieuthuc.subs({x:u, y:v})
>>> bieuthuc_theo_uv
```

…… ← sinh viên điền kết quả vào

Chúng ta có thể thử nghiệm các ví dụ khác:

```
>>> thaythe_tinhtoan = bieuthuc.subs({x:2*u*v, y:u**2+v**2})
>>> thaythe_tinhtoan
u**2 + 2*u*v + v**2
>>> thaythe_tinhtoan.factor()
```

…… ← Sinh viên điền kết quả

Ví dụ khác:

```
>>> import sympy
>>> x = Symbol('x')
>>> y = Symbol('y')
>>> bieuthuc = x + y
>>> bieuthuc2 = x**2 + y**2
```

```
>>> u = Symbol('u')

>>> v = Symbol('v')

>>> a = Symbol('a')

>>> from sympy import sin, cos
```

Và các câu lệnh tiếp theo:

```
>>> bieuthuc_theo_uv = bieuthuc2.subs({x : a*sin(u), y : a*cos(u)})

>>> bieuthuc_theo_uv

……..……← sinh viên điền kết quả

>>> bieuthuc_theo_uv.simplify()

……..……← sinh viên điền kết quả
```

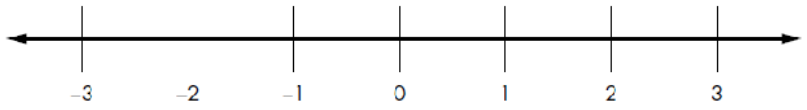
2. Vẽ biểu đồ với gói matplotlib

2.1. Trục số và mặt phẳng

Trong mục này, chúng ta sẽ học cách thể hiện dữ liệu dạng số: bằng cách vẽ đồ thị trong Python. Chúng ta sẽ bắt đầu với trục số, mặt phẳng Cartesian. Kế tiếp, chúng ta sẽ học cách thức vẽ bằng thư viện matplotlib và cách tạo các đồ thị. Sau đó, chúng ta sẽ học cách thể hiện đồ thị từ dữ liệu.

2.1.1. Trục số:

Xét trục số như hình bên dưới thể hiện một đoạn các số nguyên từ -3 đến 3 được đánh dấu trên trục. Giữa hai số chúng ta luôn có một số được xác định. Như vậy, các giá trị như 1.1, 1.2, 1.3,… sẽ nằm ở một vị trí của trục số.

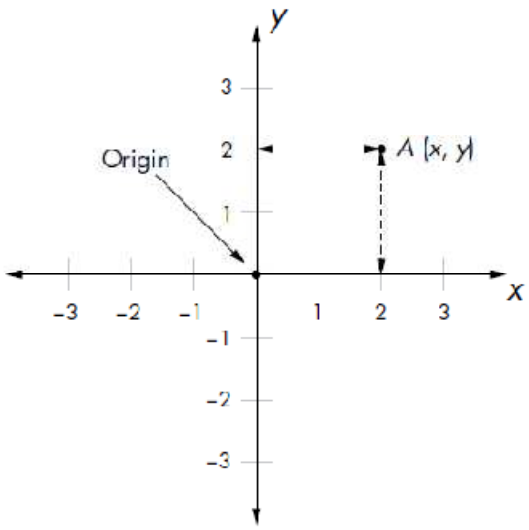


Trục số thể hiện một số thông tin thuộc tính một cách trực quan. Ví dụ như: các số bên phải số 0 là số dương và các số bên trái là số âm. Số bên phải lớn hơn số bên trái. Hai phía của trục số được kéo dài đến vô cực và bất kỳ các điểm nào trên trục số đều tương ứng với một giá trị thực.

2.1.2. Hệ tọa độ phẳng Cartesian:

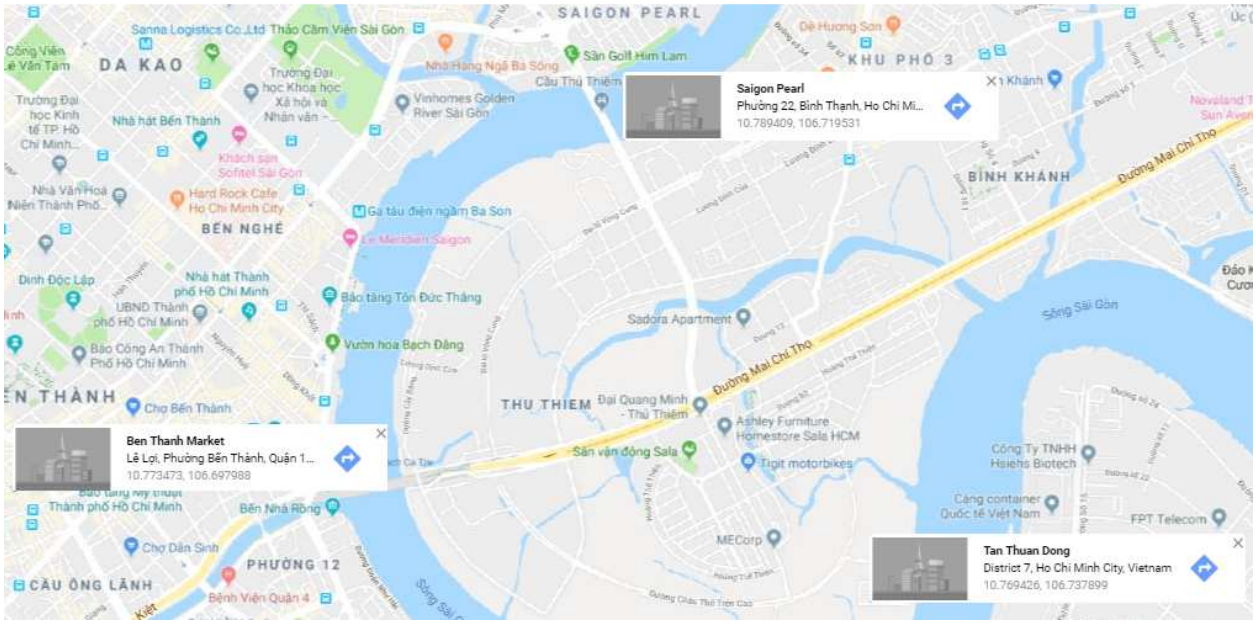
Bây giờ, xét hai trục số vuông góc nhau như hình bên dưới. Hai trục số cắt nhau tại điểm 0 của mỗi đường thẳng, điểm giao giữa hai đường thẳng gọi là điểm gốc (origin). Dạng hệ tọa độ này

gọi là hệ tọa độ phẳng Cartesian hoặc hệ x-y phẳng có hai trục gọi là trục ngang gọi là trục x và trục đứng gọi là trục y.



Chúng ta có thể mô tả một điểm bằng cặp số thay vì một giá trị số. Ví dụ: chúng ta có thể mô tả 1 điểm A trong hình bởi 2 số x và y; viết ở dạng (x,y), còn gọi là tọa độ của điểm. Gốc tọa độ được chọn là điểm O(0,0), là nơi giao giữa hai trục.

Theo đó, x là khoảng cách từ điểm A đến điểm gốc dọc theo trục X; tương tự, y là khoảng cách từ điểm A đến điểm gốc dọc theo trục Y. Hệ tọa độ phẳng Cartesian cho phép chúng ta trực quan hóa quan hệ giữa hai tập số.



Lưu ý: Hệ tọa độ toán học ứng dụng tương tự với các hệ tọa độ địa lý. Hệ tọa độ địa lý như chúng ta thấy trên Google Map sẽ có trục Long (X) là trục ngang (là các vĩ tuyến, viết tắt

longitude, 0 ở đài thiên văn Greenwich ở nước Anh) và trục Lat (Y) là trục đứng (là trục kinh tuyến, nghĩa là latitude, 0 là đường xích đạo). Dưới đây là minh họa về vị trí theo hệ tọa độ địa lý trong bản đồ TP.HCM:

- Tại Chợ Bến Thành: (lat, long) = (**10.77**3473, **106.69**7988): giá trị long (vĩ độ) thấp nhất;
- Tại Cảng container: (lat, long) = (**10.76**9426, **106.73**7899): lat (kinh độ) thấp nhất; long cao nhất;
- Tại SAIGON PEARL: (lat, long) = (**10.78**9409, 106.719531): lat cao nhất;

2.2. Danh sách List và bộ Tuple

Để vẽ đồ thị, chúng ta phải làm việc với kiểu dữ liệu danh sách list và bộ tuple trong Python. Trong Python, có nhiều cách để lưu trữ nhóm các giá trị. Trong đó, list và tuple là hai dạng phổ biến để xử lý. Sau khi tạo danh sách, chúng ta có thể thêm giá trị vào nó và thay đổi thứ tự cũng như giá trị của nó. Ngược lại, giá trị của tuple sẽ cố định khi khởi tạo, nghĩa là không thay đổi được. Chúng ta sẽ sử dụng list để lưu trữ tọa độ x và y các điểm để vẽ.

Lưu ý: với tuple, do đặc thù không thể bổ sung nên kiểu dữ liệu tuple sẽ hỗ trợ vẽ các đồ thị có miền cố định.

Trước tiên, chúng ta xem xét lại kiểu dữ liệu **list**. Một danh sách **list** chứa các số được tạo bằng lệnh liệt kê các con số trong dấu ngoặc vuông và mỗi số cách nhau bằng dấu phẩy (,). Ví dụ dưới đây là tạo danh sách list của 3 số: 10, 15, 20:

Thực hành 3: Tạo danh sách

```
>>> danhsach_so = [10, 15, 20]
```

Danh sách trên vừa được tạo có thứ tự được gọi là chỉ mục (*index*). Chỉ mục của kiểu list bắt đầu từ số 0 để tham chiếu đến giá trị đầu tiên và kế tiếp là chỉ mục thứ 1, 2,… Như vậy, tương ứng các giá trị là:

```
>>> danhsach_so[0]

10

>>> danhsach_so[1]

15

>>> danhsach_so[2]

20
```

Lưu ý 1: Các danh sách chuỗi cũng được khai báo tương tự như vậy. Các chuỗi sẽ được thay thế các số. Ví dụ: >>> ds_chuoi = [‘truc x’ , ‘truc y’, ‘truc z’]

Lưu ý 2: Danh sách rỗng là danh sách chưa có phần tử. Ví dụ: >>> ds_rong = []. Danh sách và danh sách rỗng nói riêng đều có thể thêm phần tử bằng lệnh **append**.

Ví dụ: >>> ds_rong.**append**(1)

Lưu ý 3: Việc tạo tuple cũng tương tự như việc tạo danh sách. Chỉ khác là dữ liệu thay vì nằm trong dấu [ ] thì sẽ được thay bằng dấu ( ).

Lưu ý 4: Cả hai kiểu dữ liệu **list** và **tuple** đều có thể sử dụng chỉ mục âm. Chỉ mục âm từ -1, -2, … tương ứng với các giá trị ở cuối, kế cuối,…

Lưu ý 5: tuple không có phương thức append và nội dung tuple không thể thay thế, cập nhật.

2.2.1. Duyệt các phần tử trong danh sách List và Tuple

Chúng ta có thể lặp trên khắp list và tuple bằng việc sử dụng lệnh lặp như sau:

Thực hành 4: Duyệt danh sách

```
>>> ds = [1, 2, 3]
>>> for so in ds:
        print(so)
……. ← sinh viên điền  kết quả.
…….
…….
```

Chúng ta có thể sử dụng hàm enumerate để duyệt

```
>>> ds = [10, 11, 12]
>>> for chiso, giatri in enumerate(ds):
        print(chiso, giatri)
……. ← sinh viên điền  kết quả.
…….
…….
```

2.3. Vẽ đồ thị với Matplotlib

Chúng ta sẽ sử dụng thư viện matplotlib để tạo đồ thị với Python. Matplotlib là một gói thư viện của Python bao gồm tập các module liên quan đến tính năng vẽ đồ thị. Matplotlib không phải là thư viện chuẩn của Python do đó nếu chưa có gói này trong hệ thống Python thì chúng ta phải cài đặt thêm.

Sau khi được cài đặt, chúng ta có thể sử dụng thư viện trên trình IDLE hoặc tương tự. Chúng ta sẽ tạo đồ thị đầu tiên với 3 giá trị: (1,2), (6, 5) và (8, 9). Để tạo đồ thị này, chúng ta phải tạo hai dãy số tương ứng với 1 dãy lưu trữ các giá trị trong hệ tọa độ x và 1 dãy lưu trữ các giá trị trong hệ tọa độ y:

Thực hành 5: Vẽ đồ thị đầu tiên từ 2 danh sách số (list)

```
>>> x_numbers = [1, 6, 8]
```

```
>>> y_numbers = [2, 5, 9]
```

Để tạo đối tượng vẽ đồ thị, chúng ta sử dụng phương thức/hàm **plot**() với lần lượt tham số đầu tiên là tập giá trị x và tập các giá trị y tương ứng như lệnh như sau:

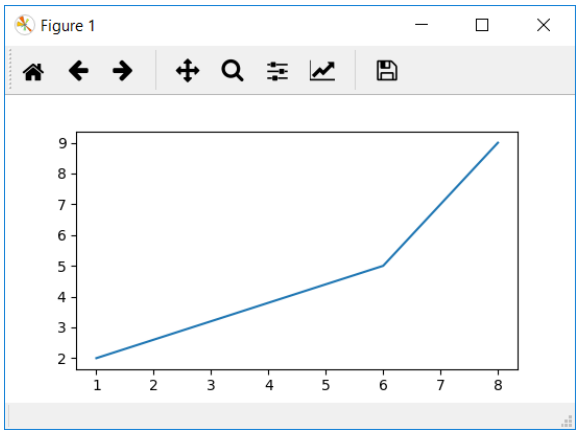
```
>>> from pylab import plot, show
```

```
>>> plot(x_numbers, y_numbers)
```

```
[<matplotlib.lines.Line2D object at 0x000001BEE2587668>]
```

Cuối cùng, chúng ta hiển thị đồ thị bằng 2 phương thức **plot**() đi kèm là **show**(). Đây là các chức năng của gói pylab (một phần của gói matplotlib).

```
>>> show()
```



Lưu ý rằng: thay vì bắt đầu từ điểm gốc O(0,0), trục x bắt đầu từ 1 và trục y bắt đầu từ 2 là điểm đầu tiên của đồ thị (1 và 2 là các giá trị nhỏ nhất của trục x và trục y từ các giá trị nhập vào).

Lưu ý: một số trình shell Python sẽ không tương tác được đến khi chúng ta đóng cửa sổ matplolib chứa đồ thị.

2.3.1. Tạo điểm trên đồ thị

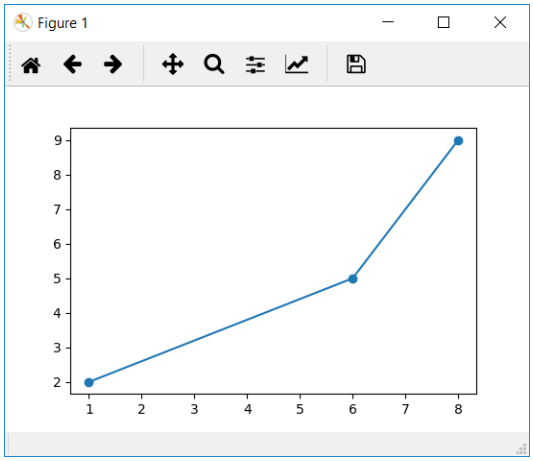
Bên cạnh việc tạo đồ thị, chúng ta có thể tạo các điểm phục vụ việc vẽ. Để tạo điểm trên đồ thị, chúng ta có thể chọn một marker (như: 'o', 'x', '*', '+').

Thực hành 6: Vẽ đồ thị có marker

```
>>> plot(x_numbers, y_numbers, marker = 'o')

[<matplotlib.lines.Line2D object at 0x000001BEE0FE9320>]

>>> show()
```



Lưu ý 1: Nếu chúng ta chỉ muốn vẽ các điểm và không cần vẽ các đường nối thì trong câu lệnh không cần chữ 'marker', cụ thể:

```
>>> plot(x_numbers, y_numbers, 'x')

>>> show()
```

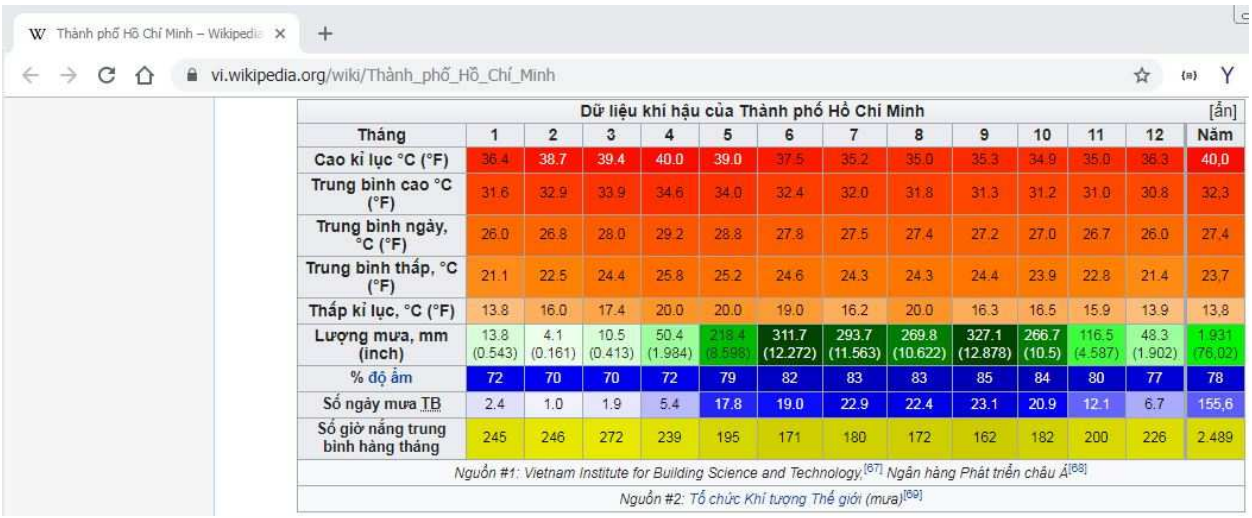
Lưu ý 2: Không phải mọi kí tự đều được xem như marker, như: 'a', 'b', 'c',… không được chấp thuận là những marker. Khi chúng ta chọn những kí tự không được xem là marker, lỗi sẽ phát sinh.

```
>>> plot(x_numbers, y_numbers, marker = 'c')

…… ← Yêu cầu sinh viên ghi nhận tên lỗi.
```

2.3.2. Vẽ đồ thị một số thông tin khí hậu theo tháng tại thành phố Hồ Chí Minh

Số liệu về khí hậu (như nhiệt độ, lượng mưa, độ ẩm, số ngày nắng,...) theo tháng ở TP.HCM được cập nhật lên trang Wikipedia. Dưới đây là dữ liệu được lấy từ một thời điểm truy cập năm 2019:



Thành phố Hồ Chí Minh – Wikipedia

vi.wikipedia.org/wiki/Thành_phố_Hồ_Chí_Minh

Dữ liệu khí hậu của Thành phố Hồ Chí Minh [ẩn]

Tháng	1	2	3	4	5	6	7	8	9	10	11	12	Năm
Cao kỉ lục °C (°F)	36.4	38.7	39.4	40.0	39.0	37.5	35.2	35.0	35.3	34.9	35.0	36.3	40,0
Trung bình cao °C (°F)	31.6	32.9	33.9	34.6	34.0	32.4	32.0	31.8	31.3	31.2	31.0	30.8	32,3
Trung bình ngày, °C (°F)	26.0	26.8	28.0	29.2	28.8	27.8	27.5	27.4	27.2	27.0	26.7	26.0	27,4
Trung bình thấp, °C (°F)	21.1	22.5	24.4	25.8	25.2	24.6	24.3	24.3	24.4	23.9	22.8	21.4	23,7
Thấp kỉ lục, °C (°F)	13.8	16.0	17.4	20.0	20.0	19.0	16.2	20.0	16.3	16.5	15.9	13.9	13,8
Lượng mưa, mm (inch)	13.8 (0.543)	4.1 (0.161)	10.5 (0.413)	50.4 (1.984)	218.4 (8.598)	311.7 (12.272)	293.7 (11.563)	269.8 (10.622)	327.1 (12.878)	266.7 (10.5)	116.5 (4.587)	48.3 (1.902)	1.931 (76,02)
% độ ẩm	72	70	70	72	79	82	83	83	85	84	80	77	78
Số ngày mưa TB	2.4	1.0	1.9	5.4	17.8	19.0	22.9	22.4	23.1	20.9	12.1	6.7	155,6
Số giờ nắng trung bình hàng tháng	245	246	272	239	195	171	180	172	162	182	200	226	2.489

Nguồn #1: Vietnam Institute for Building Science and Technology,[67] Ngân hàng Phát triển châu Á[68]

Nguồn #2: Tổ chức Khí tượng Thế giới (mưa)[69]

Trong đó với số liệu lượng mưa trung bình theo 12 tháng (theo mm), chúng ta có thể vẽ đồ thị như sau:

- Trục x là trục về tháng: 1 năm có 12 tháng.
- Trục y là trục về lượng mưa trung bình.

Thực hành 7: Vẽ đồ thị lượng mưa trung bình tại TP.HCM theo tháng.

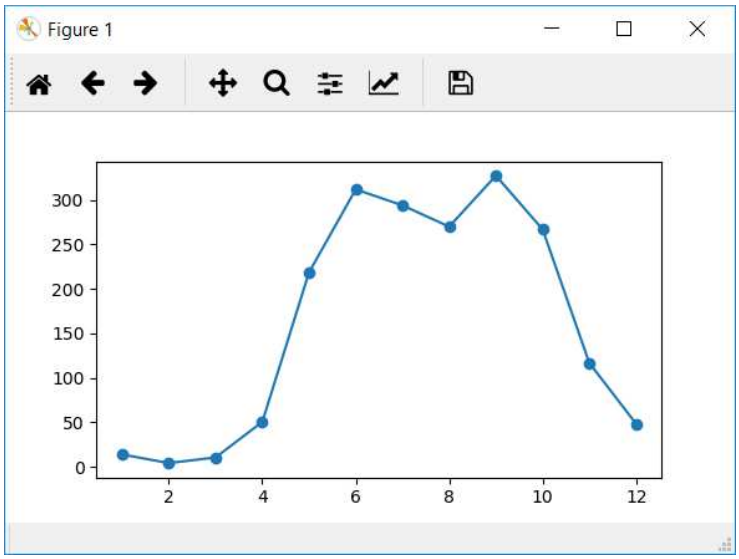
```
>>> hcm_rain = [13.8, 4.1, 10.5, 50.4, 218.4, 311.7, 293.7, 269.8, 327.1, 266.7, 116.5, 48.3]

>>> months = range(1, 13)

>>> plot(months, hcm_rain, marker = 'o')

[<matplotlib.lines.Line2D object at 0x000001BEE4E14F98>]

>>> show()
```



3. Đạo hàm

Đạo hàm của hàm số $y = f(x)$ thể hiện tỉ lệ thay đổi trong biến phụ thuộc, theo đó y phụ thuộc vào biến x và được kí hiệu là $f'(x)$ hoặc $^{dy}/_{dx}$. Chúng ta có thể tìm đạo hàm của một hàm bằng việc tạo đối tượng của lớp Derivative. Xét hàm về chuyển động của chiếc ô tô chuyển động theo phương trình $5t^2 + 2t + 8$ (sinh viên xem mô tả trong Mục 4 Chương 1):

Thực hành 8: Tính đạo hàm với Derivative

```
>>> from sympy import Symbol, Derivative
>>> t = Symbol('t')
>>> st = 5*t**2 + 2*t + 8
>>> Derivative(st, t)
…… ← Sinh viên điền kết quả
```

Với các câu lệnh trên, chúng ta vừa nhập đối tượng Derivative và tạo ra đối tượng của lớp Derivative. Hai tham số được truyền vào khi tạo đối tượng là hàm st và biểu tượng t, tương ứng với biến t. Sau đó, chúng ta sử dụng hàm dựng để đối tượng được trả về là một đối tượng thuộc lớp Derivative. Đối tượng lúc này chưa thực sự được tính toán. Để tính toán, chúng ta sử dụng phương thức doit() để tìm giá trị đạo hàm.

```
>>> d = Derivative(st, t)
>>> d.doit()
…… ← Sinh viên điền kết quả
```

Biểu thức của đạo hàm tính toán được là $10 * t + 2$. Bây giờ, chúng ta có thể tính toán cụ thể giá trị của đạo hàm tại các vị trí $t = t_1$ hoặc $t = 1$ bằng phương thức thay thế subs().

+ Về thay thế $t = 1$:

```
>>> d.doit().subs({t:1})
…… ← Sinh viên điền kết quả
```

+ Đối với việc thay thế biến t thành một biến khác thì biến đó phải được khai báo trước. Nếu không lỗi sẽ xảy ra, cụ thể là lỗi không biết tên (do chưa đặt biến) khi hệ thống không biết biến đó. Ví dụ:

```
>>> d.doit().subs({t:t1})
Traceback (most recent call last):
  File "<pyshell#43>", line 1, in <module>
    d.doit().subs({t:t1})
NameError: name 't1' is not defined
```

Do đó, chúng ta phải khai báo trước biến:

```
>>> t1 = Symbol('t1')
>>> d.doit().subs({t:t1})
…… ← Sinh viên điền kết quả
```

Nếu biến đã có giá trị, thủ tục subs() sẽ tính toán giá trị cụ thể, như trường hợp thay thế $t = 1$

```
>>> t2 = 10
>>> d.doit().subs({t:t2})
…… ← Sinh viên điền kết quả
```

Xét một hàm dạng uv, khi đó, ta có: $(uv)' = u'v + v'u$. Ví dụ: tính đạo hàm của hàm số sau:

$$f = (x^3 + x^2 + x) \times (x^2 + x)$$

Thực hành 9: Tính đạo hàm hàm hợp

```
>>> from sympy import Derivative, Symbol
>>> x = Symbol('x')
>>> f = (x**3+x**2+x)*(x**2+x)
>>> Derivative(f, x).doit()
```

```
(2*x + 1)*(x**3 + x**2 + x) + (x**2 + x)*(3*x**2 + 2*x + 1)
```

Rõ ràng hàm f ở đây là tích của hai hàm độc lập với nhau. Tuy nhiên, lớp Derivative sẽ hỗ trợ chúng ta xử lý những hàm số phức tạp.

Thực hành 10: Tính đạo hàm với các hàm lượng giác

Chúng ta phải import thư viện sympy, sau đó, chúng ta có thể tính toán bằng lớp Derivative:

a. **f(x)=sin(2x)**

```
>>> f = sympy.sin(2*x)
>>> Derivative(f, x).doit()
…… ← Sinh viên điền kết quả
```

b. **f(x) = sin(x)cos(x)**

```
>>> f = sympy.sin(x)*sympy.cos(x)
>>> Derivative(f, x).doit()
…… ← Sinh viên điền kết quả
```

Lưu ý: phải nhớ khai báo biến x trước bằng khai báo: x = **Symbol**('x')

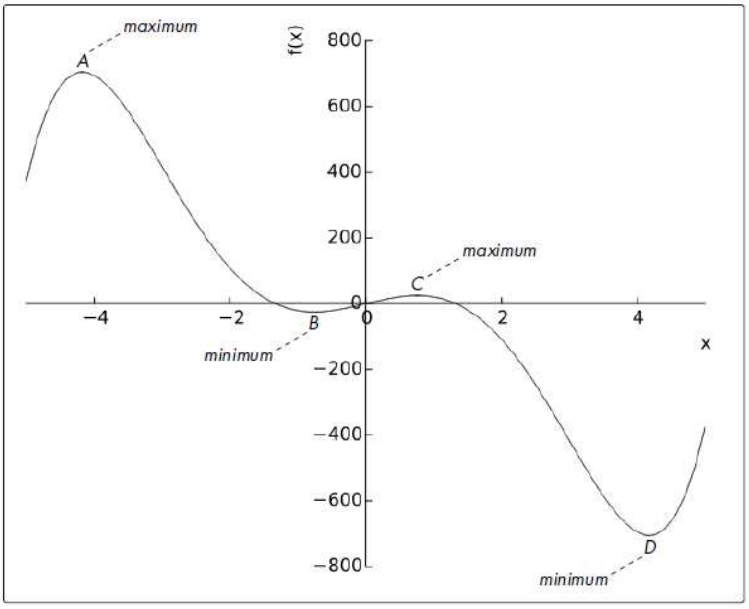
4. Đạo hàm cấp cao và bài toán cực trị

Theo mặc định, việc tạo ra đối tượng đạo hàm bằng cách sử dụng lớp Derivative để tính đạo hàm bậc 1. Để tính toán các đạo hàm cấp cao cơn, đơn giản chỉ việc xác định cấp đạo hàm muốn tính vào tham số thứ 3 khi tạo đối tượng Derivative. Trong phần này, chúng ta sẽ sử dụng đạo hàm thứ 1 và 2 để tìm cực đại và cực tiểu trên một khoảng.

Giả sử, xét hàm $x^5 - 30x^3 + 50x$ định nghĩa trên miền xác định $[-5, 5]$. Chúng ta có thể dễ dàng vẽ đồ thị của hàm số.

Từ đồ thị của hàm số, chúng ta thấy rằng nó gồm các giá trị cực tiểu tại điểm B trong khoảng $-2 \leq x \leq 0$. Tương tự, nó có giá trị cực đai tại điểm C thuộc khoảng $0 \leq x \leq 2$. Tuy nhiên, giá trị cực đại và cực tiểu của toàn bộ miền xác định nằm ở hai điểm được đặt tên là A và D tương ứng. Do vậy, điểm B và C được xem như cực tri địa phương, mà cụ thể là cực tiểu địa phương (local minimum) và cực đại địa phương (local maximum). Và điểm A và D được gọi là cực đại toàn cục (global maximum) và cực tiểu toàn cục (global minimum).

Thuật ngữ cực trị (*extramum*, số nhiều là *extrama*) đề cập đến những điểm là cực đại hoặc cực tiểu toàn cục cũng như địa phương. Đặc điểm là: nếu x_0 là cực trị của hàm $f(x)$ thì đạo hàm bậc 1 của $f(x)$ tại x_0 sẽ bị triệt tiêu (vanish, bằng 0), nghĩa là: $f'(x_0) = 0$.



Đồ thị của hàm số được vẽ

Điều này có nghĩa là để tìm được các điểm cực trị, chúng ta phải giải phương trình $f'(x) = 0$.

Lưu ý: để giải phương trình, Sympy hỗ trợ lệnh ***solve****(phương trình).*

Chúng ta thực hiện các lệnh sau:

Thực hành 11: Tìm cực đại, cực tiểu của hàm số

```
>>> from sympy import Symbol, solve, Derivative
>>> x = Symbol('x')
>>> f = x**5-30*x**3+50*x
>>> d1 = Derivative(f, x).doit()
```

Với các lệnh trên, chúng ta tính được đạo hàm bậc 1 của hàm f là $d1$. Từ đó, giải phương trình $d1 = 0$ để tìm tập hợp các điểm cực trị. Các lệnh tiếp theo như sau:

```
>>> d1 = Derivative(f, x).doit()
>>> cuctri = solve(d1)
……..…… ← Sinh viên viết các nghiệm!
……..
```

Sau đó, chúng ta lần lượt lấy các giá trị của nó gán vào các biến A, B, C, D tương ứng với 4 nghiệm:

```
>>> A = cuctri[0] # nghia la gia tri -sqrt(-sqrt(71) + 9)
>>> A = cuctri[2] # nghia la gia tri -sqrt(sqrt(71) + 9)
>>> B = cuctri[0] # nghia la gia tri -sqrt(-sqrt(71) + 9)
>>> C = cuctri[1] # nghia la gia tri sqrt(-sqrt(71) + 9)
>>> D = cuctri[3] # nghia la gia tri sqrt(sqrt(71) + 9)
```

Khi này, chúng ta có thể tính toán để xác định giá trị cực đại và cực tiểu cũng như địa phương và toàn cục. Điều này được xử lý bằng việc kiểm đạo hàm cấp 2 (second derivative test). Chúng ta thực hiện như sau:

```
>>> d2 = Derivative(d1, x, 2).doit()
```

Và sau đó, chúng ta có thể tìm lại các giá trị cực tiêu hoặc cực đại bằng biểu thức thay thế (subs) và tính toán (sử dụng hàm evalf) như sau:

```
>>> d2.subs({x:B}).evalf()
```

…….. ← Sinh viên ghi kết quả

Tương tự:

```
>>> d2.subs({x:C}).evalf()
```

…….. ← Sinh viên ghi kết quả

```
>>> d2.subs({x:A}).evalf()
```

…….. ← Sinh viên ghi kết quả

```
>>> d2.subs({x:D}).evalf()
```

…….. ← Sinh viên ghi kết quả

Giá trị cực đại sẽ có đạo hàm cấp 2 âm. Trong khi đó, giá trị cực tiểu sẽ có đạo hàm cấp 2 dương.

Ngoài ra, chúng ta phải tìm giá trị tại 2 biên x_min = -5 và x_max= 5 để tìm giá trị cực trị toàn cục và cục bộ. Chúng ta tính toán như sau:

```
>>> x_min = -5
```

```
>>> x_max = 5

>>> f.subs({x:A}).evalf()

705.959460380365

>>> f.subs({x:C}).evalf()

25.0846626340294

>>> f.subs({x:x_min}).evalf()

375.000000000000

>>> f.subs({x:x_max}).evalf()

-375.000000000000
```

Tương tự với 2 điểm cực tiểu B và D, chúng ta cũng tính toán các giá trị:

```
>>> f.subs({x:B}).evalf()

…….. ← sinh viên điền vào

>>> f.subs({x:D}).evalf()

…….. ← sinh viên điền vào

>>> f.subs({x:x_min}).evalf()

…….. ← sinh viên điền vào

>>> f.subs({x:x_max}).evalf()

…….. ← sinh viên điền vào
```

Từ những giá trị trên, chúng ta có thể xác định giá trị nhỏ nhất tương ứng với cực tiểu toàn cục và điểm cực tiểu cục bộ.

Phương pháp tìm giá trị cực trị, cụ thể hơn là tìm cực đại và cực tiểu chỉ áp dụng được đối với các hàm có đạo hàm cấp 2.

Với hàm như e^x, việc áp dụng phương pháp trên vẫn hiệu quả, chúng sẽ không có giá trị cực trị:

```
>>> g = sympy.E ** x

>>> g1 = Derivative(g, x)
```

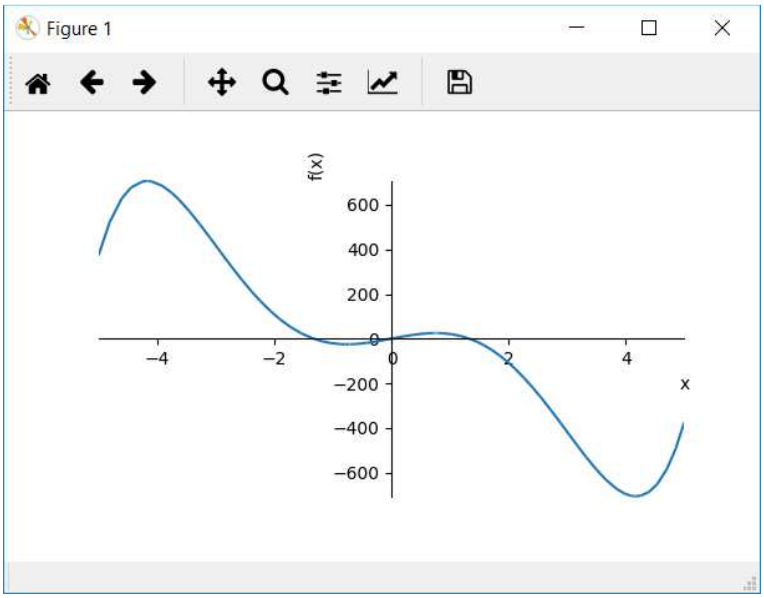
```
>>> g1
Derivative(exp(x), x)
>>> solve(g1)
[]
```

Và điều này có nghĩa là giá trị lớn nhất sẽ ở các biên của miền xác định!

Lưu ý: để vẽ đồ thị của hàm $f = x^5 - 30x^3 + 50x$, chúng ta sử dụng lệnh như sau:

```
>>> import sympy
>>> sympy.plot(f, (x, -5, 5))
```

Trong đó, tham số (x, -5, 5) là giới hạn hàm số chỉ vẽ trong khoảng [-5, 5]. Và kết quả là đồ thị được tạo thành như bên dưới:



./.

BÀI TẬP CHƯƠNG 2

Những bài tập này được yêu cầu sinh viên nộp bài trong tuần học kế tiếp

Bài tập 1: Vẽ biểu đồ số ngày mưa trung bình và số giờ nắng trung bình hằng tháng theo số liệu khí hậu của TP.HCM

Bài tập 2: Chọn 10 hàm số trong sách/giáo trình Toán Cao cấp để thực hiện việc tính đạo hàm bằng Python.