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 3: ĐẠO HÀM VÀ ỨNG DỤNG 3](#_Toc17246)

[1. Bổ túc kiến thức cơ bản Python 3](#_Toc17247)

[1.1. Viết hàm/phương thức trong Python 3](#_Toc17248)

[1.2. Cơ chế bắt lỗi trong Python 4](#_Toc17249)

[2. Vẽ đồ thị 7](#_Toc17250)

[2.1. Vẽ đồ thị về chuyển động của vật được ném lên 7](#_Toc17251)

[3. Đạo hàm: Xây dựng chương trình tính đạo hàm 14](#_Toc17252)

[4. Đạo hàm và ứng dụng: Tìm hiểu và ứng dụng phương pháp Gradient Ascent 18](#_Toc17253)

[BÀI TẬP CHƯƠNG 3 23](#_Toc17254)

# CHƯƠNG 3: ĐẠO HÀM VÀ ỨNG DỤNG

Mục tiêu:

* Bổ túc cơ bản về lập trình Python: viết phương thức, cơ chế bắt lỗi trong Python. (30 phút)
* Vẽ đồ thị tiếp theo: vẽ đồ thị theo công thức. (30 phút)
* Đạo hàm và ứng dụng của đạo hàm. (90 phút)

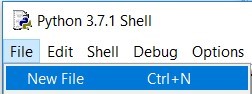
Nội dung chính:

# Bổ túc kiến thức cơ bản Python

## Viết hàm/phương thức trong Python

Trong Python, một hàm hay phương thức xử lý đơn giản được khai báo thông qua từ khóa def với tên hàm và kết quả trả về bằng từ khóa return. Nếu trả về nhiều giá trị thì mỗi giá trị cách nhau bởi dấu phẩy (,). Ví dụ: return masv, tensv là trả về 2 biến mssv và tensv.

Để bắt đầu, từ IDLE của Anaconda, chúng ta có thể viết hàm bằng việc mở một tập tin



Ví dụ: Hàm tính lũy thừa số nguyên 𝑥 như sau trong dấu nhắc lệnh >>> của IDLE

Thực hành 1: Viết hàm từ dấu nhắc lệnh >>> của trình IDLE

>>> def luythua(x, n):

ketqua= 1

for i in range(n):

ketqua = ketqua \*x return ketqua

>>> luythua(2,1)

………………………………1…………………………….  sinh viên điền kết quả

>>> luythua(2,0)

………………………………2…………………………….  sinh viên điền kết quả

Lưu ý: Một số đặc điểm nổi bật trong Python về hàm:

 Hàm “main” của một tập tin .py:

Là đoạn chương trình nằm trong một khối lệnh, thường ở vị trí cuối của tập tin .py. Khi khối lệnh này tồn tại, chúng ta có thể “thực thi” tập tin Python đó từ dòng lệnh của hệ điều hành, như:

C:\> python abc.py

Khối lệnh bắt đầu từ việc so sánh 2 từ khóa riêng \_\_name\_\_ và \_\_main\_\_ của Python như sau:

if \_\_name\_\_ == "\_\_main\_\_":

#... thực hiện gì đó….

Chúng ta sẽ làm việc rõ hơn hàm main này trong phần sau của chương

## Cơ chế bắt lỗi trong Python

Trong khi viết chương trình, nếu không tính đến lỗi thuật toán, hai loại lỗi xảy ra là:

Thực hành 2: Lỗi và cơ chế bắt lỗi trong Python

* Lỗi cú pháp (syntax errors). Ví dụ: viết dòng lệnh while thiếu dấu : hoặc điều kiện bằng trong dòng lệnh while phải là dấu “==” thay vì dấu “=”.

Ví dụ:

>>> while 1 = 2:

……SyntaxError: cannot assign to literal here. Maybe you meant '==' instead of '='?……………….  Sinh viên điền lỗi vào

* Ngoại lệ (exceptions): Ví dụ: mở tập tin không tồn tại, thực hiện phép chia 0, thực hiện phép cộng (operand type for +) theo thứ tự số + chuỗi hoặc phép nối (concatenate) theo thứ tự chuỗi + số. Ví dụ:

>>> 10 \* (1/0)

………………ZeroDivisionError: division by zero……………….  Sinh viên ghi nhận lỗi

>>> 3 \* bien\_chua\_khai\_bao / 4

NameError: name bien\_chua\_khai\_bao is not defined…………….  Sinh viên ghi nhận lỗi

>>> ‘ba’ + 7

………SyntaxError: invalid character '‘' (U+2018)………….  Sinh viên ghi nhận lỗi

>>> 10 \* (1/0)

……………ZeroDivisionError: division by zero……………….  Sinh viên ghi nhận lỗi

Tóm lại, các ngoại lệ bên trên báo là: ZeroDivisionError, NameError và TypeError.

Một ví dụ khác cho thấy việc tính toán của Python là từ trái sang phải và có ưu tiên các phép toán. Thực hành:

>>> 'ba' + (1/0)

……………IndentationError: unexpected indent……………….  Sinh viên ghi nhận lỗi

>>> 'ba'+7+(1/0)

…TypeError: can only concatenate str (not "int") to str….  Sinh viên ghi nhận lỗi

Phương pháp xử lý:

Đối với lỗi cú pháp (syntax error), chúng ta phải kiểm tra lại câu lệnh và sửa. Đối với lỗi ngoại lệ (exception) chúng ta phải xử lý và hạn chế bằng các cơ chế thông báo để tránh trường hợp chương trình bị kết thúc bất ngờ.

Thực hành 3: Xử lý ngoại lệ với cơ chế try… except

 Sử dụng cơ chế try… except… để xử lý lỗi:

>>> x = int(input(“Vui long nhap so: ”))

Vui long nhap so: a

Sinh viên điền tên ngoại lệ:……………teddy……………………..

>>> while True:

try:

x = int(input(“Vui long nhap so: ”))

break

except <sinh viên điền vào tên ngoại lệ tìm được bên trên vào đây>…nhập số đúng với điều kiện try nên break ra khỏi vòng lặp… Vui long nhap so. Khong nhap chu…..: print (‘Vui long nhap so. Khong nhap chu!’)

Cơ chế try… except có cấu trúc else:

Thực hành 4: Xử lý ngoại lệ với cơ chế try… except… else… finally

Cơ chế else trong try except để thực hiện tiếp các lệnh trong try nếu không có lỗi/ngoại lệ xảy ra. Lưu ý 1: Nếu except mà không chỉ ra lỗi/ngoại lệ thì xem như bắt tất cả các lỗi và ngoại lệ.

Lưu ý 2: Nếu có thêm finally thì để xử lý tiếp các câu lệnh còn lại. Thường là các thông báo để người sử dụng biết kết thúc một chương trình.

Ví dụ:

>>> while True:

try:

x = int(input('Vui long nhap so: '))

except:

print ('So vua nhap sai')

else:

print ('so vua nhap la ' + str(x))

break

finally:

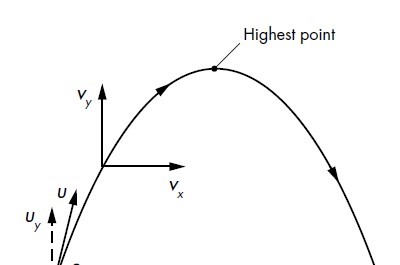
print ('Da hoan thanh chuong trinh')

Sinh viên thử nghiệm nhập chữ a. Rồi Enter và nhập số 1.

# Vẽ đồ thị

## Vẽ đồ thị về chuyển động của vật được ném lên

Trong mục này, chúng ta sẽ làm quen với bài toán về ném quả bóng và theo dõi quỹ đạo của nó (như hình mô tả bên dưới).



Trong hình trên, quả bóng được ném từ điểm A và rơi xuống điểm B. Đây là dạng quỹ đạo của đạn đại bác hoặc của một vật thể được ném lên. Mục tiêu ở đây là sử dụng các phương trình đạn đạo để vẽ đồ thị của vật thể bay và cho thấy các vị trí của quả bóng từ lúc ném đến khi chạm vào điểm B.

Khi ném quả bóng, chúng ta xác định được vận tốc và hướng ban đầu của quả bóng. Hướng bay là góc tạo bởi mặt đất và vector chỉ hướng của quả bóng. Ta lần lượt gọi 2 đại lượng này là vận tốc 𝑢 và góc 𝜃 (đọc là theta) (như hình). Quả bóng gồm 2 thành phần vận tốc: dọc theo trục 𝑥 được tính theo công thức 𝑢 = 𝑢𝑐𝑜𝑠𝜃 và dọc theo trục y là 𝑢 = 𝑢𝑠𝑖𝑛𝜃.

Khi quả bóng bay, vận tốc của nó sẽ thay đổi và chúng ta sẽ thể hiện sự thay đổi vận tốc bằng giá trị 𝑣, với hai thành phần ngang là 𝑣 và chiều dọc là 𝑣 . Để đơn giản, giả định thành phần 𝑣 không thay đổi trong suốt quá trình bay của vật thể. Trong khi đó, thành phần 𝑣 sẽ bị tác động bởi trọng trường theo phương trình 𝑣 = 𝑢 − 𝑔𝑡. Trong phương trình này, 𝑔 là gia tốc trọng trường và 𝑡 là thời gian mà vận tốc được đo. Vì 𝑢 = 𝑢𝑠𝑖𝑛𝜃 nên ta có thể thay thế:

𝑣 = 𝑢𝑠𝑖𝑛𝜃 − 𝑔𝑡.

Vì thành phần ngang của vận tốc được xem như không đổi, do đó, thành phần ngang sẽ đi được một khoảng là 𝑆 = 𝑢(𝑐𝑜𝑠𝜃)𝑡. Thành phần theo chiều đứng có vận tốc thay đổi và do đó, khoảng cách di chuyển được cho bởi công thức:

𝑆 = 𝑢(𝑠𝑖𝑛𝜃)𝑡 −  𝑔𝑡 .

Nói cách khác, 𝑆 và 𝑆 sẽ cho chúng ta tọa độ của quá bóng tại bất kỳ thời điểm nào trong khi bay. Chúng ta sẽ sử dụng phương trình để vẽ quỹ đạo. Theo đó, chúng ta sử dụng phương trình với các đại lượng được tính như sau: thời gian 𝑡 được tính bằng giây, vận tốc được tính bằng 𝑚⁄𝑠, góc chiếu ban đầu là 𝜃 theo đơn vị độ và gia tốc trọng trường được thể hiện là 𝑚 𝑠 .

Tuy nhiên, trước khi chúng ta viết chương trình, chúng ta cần tìm khoảng thời gian quả bóng bay trước khi nó tiếp đất để chúng ta vẽ được quỹ đạo của quả bóng. Để làm việc này, đầu tiên, chúng ta tìm tính điểm cao nhất của quả bóng, khi đó, vận tốc của quả bóng theo chiều y (𝑣 ) là 0. Nghĩa là ta có phương trình toán 𝑣 = 𝑢𝑠𝑖𝑛𝜃 − 𝑔𝑡 = 0. Từ đó, chúng ta có thể tìm được nghiệm 𝑡 như sau:

𝑢𝑠𝑖𝑛𝜃

𝑡 =

### 𝑔

Ta gọi thời điểm này là thời điểm đỉnh 𝑡\_peak. Sau khi đến vị trí cao nhất, quả bóng cần một số thời gian như lúc quả bóng rơi xuống nơi nào đó. Khi đó, ta có tổng thời gian bay là:

𝑢𝑠𝑖𝑛(𝜃)

𝑡 = 2𝑡 = 2 ×

### 𝑔

Với giả định vận tốc ban đầu là 5m/giây và góc bay là 90 độ. Khi đó, việc tính thời gian bằng cách thay thế các giá trị 𝑢 = 5, 𝜃 = 45𝜃. Và chúng ta có thể thay thế các số vào trong phương trình trên, ta được:

### 𝑡 = 2𝑡 = 2 ×

Trong trường hợp này, thời gian bay trên không của quả bóng là 0.7251 giây (làm tròn đến trường số 5 số thập phân). Quả bóng trên không trung trong thời gian này. Do đó, để vẽ được quỹ đạo, chúng ta sẽ tính tọa độ x và y theo các khoảng thời gian. Câu hỏi đặt ra là chúng ta sẽ tính tọa độ như thế nào? Một cách lý tưởng, chúng ta có thể tính tọa độ đều. Trong phần này, chúng ta sẽ tính tọa độ cho mỗi 0.001 giây.

2.1.1. Phát sinh dãy số các số thực đều nhau

Hàm range() để phát sinh dãy các số nguyên đều nhau. Nghĩa là nếu ta muốn liệt kê số nguyên giữa 1 và 10, mỗi số các nhau 2 đơn vị thì chúng ta có thể sử dụng lệnh range(1, 10, 2). Tuy nhiên, với số thực, giả định: chúng ta muốn phát sinh các số từ 0 đến 0.32 với khoảng cách 0.02. Nghĩa là chúng ta phải tạo ra các số: 0.00, 0.02, 0.04,. 0.06, 0.08,…, 0.28, 0.30, 0.32.

Khi đó, chúng ta phải thực hiện lệnh lặp để tạo dãy, như sau:

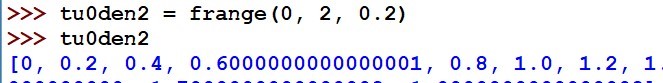
Thực hành 5: Tạo hàm phát sinh dãy số thực đều

>>> def frange(batdau, ketthuc, buocnhay):

day\_ketqua = [] while batdau < ketthuc:

day\_ketqua.append(batdau) batdau = batdau + buocnhay return day\_ketqua

Sau khi xây dựng hàm frange, chúng ta có thể sử dụng. Ví dụ: để tạo dãy số từ 0 đến 2 với bước nhảy 0.2:



2.1.2. Vẽ quỹ đạo bay của vật được ném lên

Chương trình dưới đây sẽ vẽ quả bóng bay với vận tốc và gốc ban đầu được cho. Lưu ý: cả hai tham số này được nhập trong chương trình.

Thực hành 6: Vẽ quỹ đạo bay của vật được ném lên

Chúng ta thực hiện tạo mới một tập tin và đưa vào các đoạn mã sau:

Phân đoạn 1: Nhập các thư viện xử lý:

>>> from matplotlib import pyplot as plt

>>> import math

Phân đoạn 2: Xây dựng các hàm xử lý:

+ Hàm 1: Hàm vẽ đồ thị def draw\_graph(x, y):

plt.plot(x, y) plt.xlabel('Truc X') plt.ylabel('Truc Y') plt.title('Do thi bai toan nem qua bong') + Hàm 2: Hàm xử lý về tạo các khoảng số thực đều nhau def frange(start, final, interval):

numbers = [] while start < final:

numbers.append(start) start = start + interval return numbers

+ Hàm 3: Hàm tính toán các dãy X và Y để vẽ đồ thị tương ứng với hàm đạn đạo:

def draw\_trajectory(u, theta): # Gia toc trong truong g = 9.8 # Goc bay:

theta = math.radians(theta)

# Thoi gian bay:

t\_flight = 2\*u\*math.sin(theta)/g # Tinh toan khoang cach thoi gian intervals = frange(0, t\_flight, 0.001) # Danh sach toa do x va y

x = [] y = [] for t in intervals:

x.append(u\*math.cos(theta)\*t)

y.append(u\*math.sin(theta)\*t - 0.5\*g\*t\*t) draw\_graph(x, y)

Phân đoạn 3: Xây dựng hàm main if \_\_name\_\_ == '\_\_main\_\_':

try:

u = float(input('Nhap van toc ban dau (m/s): ')) theta = float(input('Nhap goc bay (degrees): ')) except ValueError:

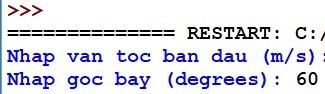
print (' Nhap cac gia tri sai! ') else:

draw\_trajectory(u, theta) plt.show() finally:

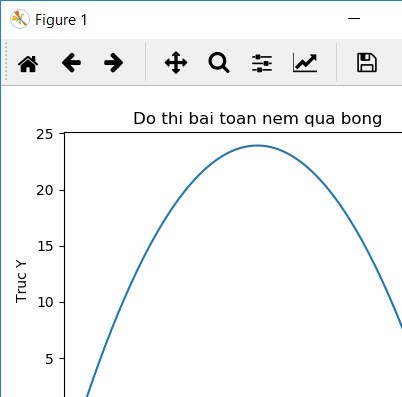
print ('Hoan thanh chuong trinh!')

Trong đoạn lệnh này, chúng ta sử dụng cơ chế bắt lỗi try except của Python để loại trừ những sai sót hoặc cố ý nhập không phải số. Việc nhập các số được hàm input xử lý và chuyển sang các số thực bằng hàm float() để tính toán và vẽ đồ thị. Sau khi hoàn thành, chương trình sẽ kết thúc và in ‘Hoan thanh chuong trinh!’.

Sau đó chúng ta lưu tập tin và thực hiện thực thi bằng cách nhấn F5 hoặc vào trong menu Run  Run Module (trên môi trường IDLE)



Sau khi nhập thông số, chương trình sẽ vẽ đồ thị:



Trên đây là đồ thị của vật thể được ném lên với vận tốc ban đầu là 25m/s và góc hướng lên là 60 độ.

2.1.3. Vẽ nhiều đồ thị với các vận tốc ban đầu khác nhau

Trong chương trình bên trên, chúng ta thực hiện thử nghiệm. Chúng ta có thể điều chỉnh chương trình để thể hiện nhiều tốc độ ban đầu:

Thực hành 7: Vẽ nhiều quỹ đạo trên cùng một đồ thị

Cụ thể là thay đổi hàm main, nghĩa là phân đoạn 3 trong chương trình bên trên:

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

# Danh sach cac toc do ban dau khac nhau ds\_vantoc = [20, 40, 60] goc\_bandau = 45 # 45 do for v in ds\_vantoc:

draw\_trajectory(v, goc\_bandau)

# Tao chu dan trong do thi:

chudan = []

for i in ds\_vantoc:

chudan.append(str(i)) # Xem chu dan duoc tao:

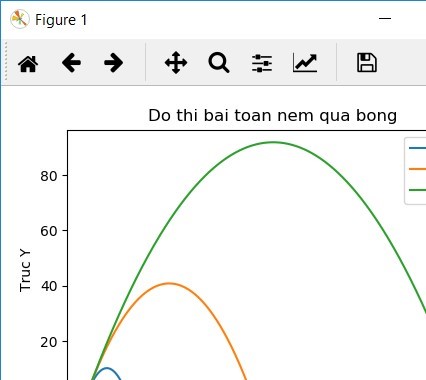
print ("Chu dan duoc tao la: ") print (chudan)

# Them chu dan vao do thi plt.legend(chudan)

# Hien thi do thi:

plt.show()

Kết quả thực hiện là đồ thị và các giá trị vận tốc ban đầu được thể hiện trong màn hình IDLE:

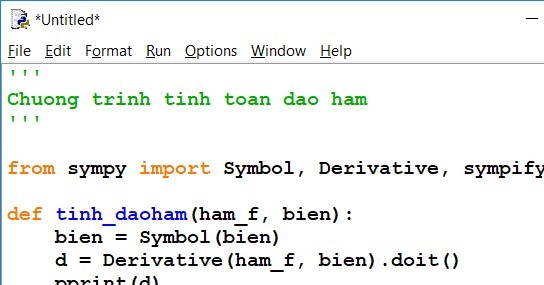


# Đạo hàm: Xây dựng chương trình tính đạo hàm

Việc viết một hàm số lượng giác sin(), cos() cần thể hiện cả tên thư viện sympy sẽ gây rất bất lợi trong các ứng dụng tính toán mà người sử dụng có thể không hề biết đến gói SymPy. Do đó, bây giờ, chúng ta sẽ tìm cách viết một chương trình máy tính để tính toán đạo hàm của một hàm số theo một biến với những ưu việt hơn. Tất nhiên, phần lõi của chương trình vẫn là phần triệu gọi Derivative cho hàm f theo một biến và thực hiện tính toán bằng phương thức doit().

Trong khung cửa sổ soạn thảo (bằng cách vài menu File  New File), chúng ta viết hàm tinh\_daoham như sau:

Thực hành 8: Xây dựng chương trình tính đạo hàm



Đoạn mã:

'''

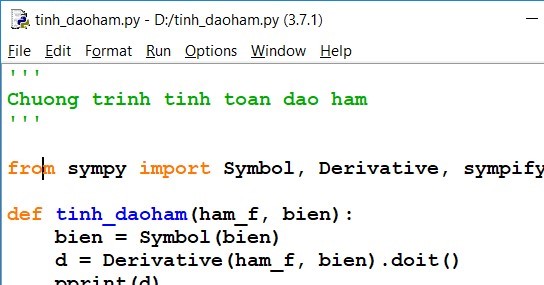
Chuong trinh tinh toan dao ham

'''

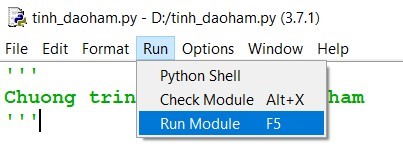
from sympy import Symbol, Derivative, sympify, pprint

def tinh\_daoham(ham\_f, bien): bien = Symbol(bien) d = Derivative(ham\_f, bien).doit() pprint(d)

Khi chưa lưu, tên cửa sổ là \*Untitled\*. Chúng ta có thể lưu vào đĩa D với tên tập tin là tinh\_daoham.py.



Sau khi lưu vào đĩa, chúng ta vào menu Run  Run Module hoặc nhấn F5 để sử dụng hàm trên:



Khi được nhấn F5 để chạy, dấu nhắc được RESTART như sau:



Khi đó, chúng ta có thể thử nghiệm việc thực thi hàm tinh\_daoham với các tham số hàm được nhập như các chuỗi (kí tự) như sau:

Lưu ý: trong Python, chuỗi kí tự được đặt trong 2 dấu nháy đơn ‘’ hoặc nháykép “”

>>> tinh\_daoham('x\*\*3+x\*\*2+x', 'x')

…………………………………………………  Sinh viên điền kết quả



Rõ ràng, với chương trình này, chúng ta thấy một sự khác biệt với câu lệnh thô sơ. Với chương trình này, ngoài việc tính toán, chúng ta thấy kết quả được trình bày thông qua hàm pprint() của SymPy. Hàm pprint() hỗ trợ chúng ta định dạng kết quả một cách toán học tốt hơn so với lệnh print kết xuất thông thường của Python. Thể hiện kết quả trực quan hơn cũng là một ưu điểm của gói SymPy làm chúng ta cảm giác sự thân thiện của một gói tính toán toán học.

Với hàm trên, chúng ta không thể tránh được những lỗi phát sinh do vô tình của sự bất cẩn hoặc cố ý. Ví dụ: với lệnh dưới đây, lỗi sẽ xuất hiện:

>>> tinh\_daoham('x\*/3+x\*\*2+r', 'x')

………………………………………………………..  Sinh viên ghi nhận lỗi

Bên cạnh đó, để “giao tiếp” với hệ thống tốt hơn, chúng ta có thể bổ sung thêm các trao đổi giữa hệ thống và người sử dụng.

Từ đó, chúng ta phải nâng cấp chương trình theo 2 hướng:

* Vấn đề 1: Quản lý được lỗi phát sinh (bằng cơ chế try…except).
* Vấn đề 2: Thực hiện việc giao tiếp với người sử dụng chương trình tốt hơn.

Để thực hiện vấn đề thứ 2, chúng ta có thể sử dụng cơ chế nhập input chuẩn của Python. Cơ chế nhập cho phép nhập một hàm và nhập tên biến để gọi hàm. Bên cạnh đó, bản thân SymPy cũng cung cấp 1 đối tượng để chuyển đổi giá trị từ input hàm thông thường thành hàm của SymPy, đó là hàm sympify. Lớp sympify sẽ hỗ trợ chúng ta kể cả việc bắt lỗi với mã lỗi SympifyError khi chuyển đổi hàm sang dạng hàm của SymPy() không thành công. Để từ đó, chúng ta có thể sử dụng được các hàm một cách tự nhiên, như hàm sin(), cos() mà không cần phải thể hiện khó hiểu như: sympy.sin(…).

Đối với vấn đề 1, chúng ta có thể sử dụng cơ chế kiểm soát lỗi và ngoại lệ với khối try… except.

Nghĩa là chúng ta sẽ triệu gọi phương thức tinh\_daoham() bên trong khối lệnh try:… except:… Thông qua đó, chúng ta cũng có thể viết thành một chương trình độc lập. Một chương trình độc lập được đặt trong khối lệnh bắt đầu bằng trong chương trình if \_\_name\_\_ == ‘\_\_main\_\_’:

Cụ thể như sau:

Đoạn mã:

from sympy import Symbol, Derivative, sympify, pprint from sympy.core.sympify import SympifyError

def tinh\_daoham(ham\_f, bien): bien = Symbol(bien) d = Derivative(ham\_f, bien).doit() pprint(d)

if \_\_name\_\_ == '\_\_main\_\_': # luu y 2 dau gach duoi \_ o moi chu ham\_input = input('Nhap 1 ham so: ') bien = input('Nhap bien so: ')

try:

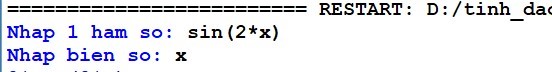
ham\_sympy = sympify(ham\_input) except SympifyError:

print('Ham nhap bi loi!') else:

tinh\_daoham(ham, bien)

Lưu ý: dòng if \_\_name\_\_ … là 2 dấu gạch dưới \_ trước và sau các chữ name và chữ main.

Chúng ta thực hiện việc thử nghiệm thực thi bằng cách nhấn F5 (hoặc menu Run  Run Module và nhập vào hàm sin(2\*x). Minh họa bằng hình: khi nhập hàm và kết quả tính đạo hàm:



Chúng ta có thể thực thi lần nữa chương trình (bằng F5) với hàm sin (𝑥 )

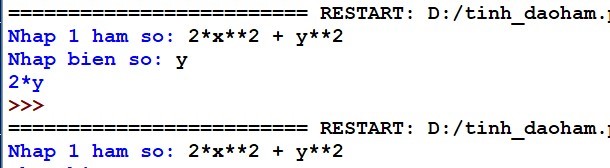
Nhap 1 ham so: sin(x\*\*2)

Nhap bien so: x

………………………………………………………  sinh viên điền kết quả.

Một ví dụ khác, với hàm nhiều biến, khi tính đạo hàm với 1 biến thì các biến khác được xem là các hằng số trong phép đạo hàm theo 1 biến. Cụ thể:

Với hàm 𝑓 = 2𝑥 + 𝑦 , chúng ta tính đạo hàm theo 𝑦 và theo 𝑥 sẽ được kết quả như sau:



# Đạo hàm và ứng dụng: Tìm hiểu và ứng dụng phương pháp Gradient Ascent

Đôi khi, chúng ta sẽ thấy thú vị trong bài toán tìm giá trị lớn nhất toàn cục hơn là việc chỉ tìm các cực trị của hàm số. Ví dụ: xét bài toán ném/bắn đạn vừa vẽ đồ thị bên trên, chúng ta muốn khám phá góc ném/“bắn” mà quả bóng sẽ đạt được độ cao nhất với một vận tốc ban đầu không đổi. Chúng ta sẽ tiếp cận một phương pháp sẽ hỗ trợ cho chúng ta tính toán, giải quyết đạo hàm với những vấn đề như vậy, đặc biệt trong những ứng dụng thực tế cuộc sống mà các “hàm” chỉ có thể tính được đạo hàm bậc nhất.

Phương pháp này gọi là “Phương pháp gradient ascent”. Là phương pháp lặp để tìm giá trị lớn nhất toàn cục. Vì vậy, phương pháp gradient ascent là phương pháp cần thực hiện nhiều tính toán và người ta thường sử dụng máy tính hơn so với việc tính tay. Trong các bài trước, chúng ta đã có được phương trình thời gian bay của vật thể là:

𝑢𝑠𝑖𝑛𝜃

𝑡 = 2

### 𝑔

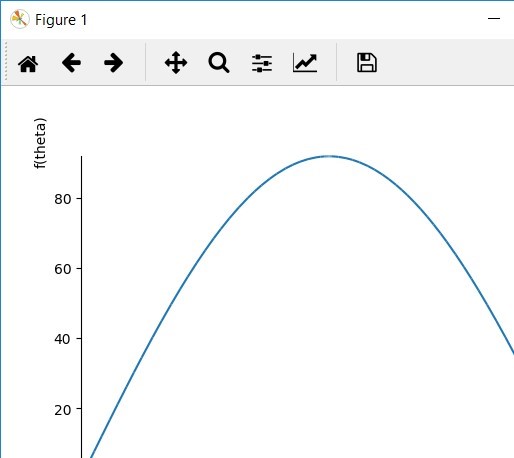
Để tính toán được thời gian bay của một vật thể trong chuyển động “đạn đạo” (projectile) được ném với vận tốc ban đầu là 𝑢 theo một góc 𝜃. Miền projectile 𝑅 là chiều dài vật thể đi xa được và được cho bởi tích 𝑢 × 𝑡 . Ở đây 𝑢 là thành phần ngang bao gồm vận tốc ban đầu và có giá trị là 𝑢𝑐𝑜𝑠𝜃. Thay thế công thức vào, chúng ta được:

2𝑢𝑠𝑖𝑛𝜃 𝑢 𝑠𝑖𝑛2𝜃

𝑅 = 𝑢 𝑡 = 𝑢𝑐𝑜𝑠𝜃 × =

### 𝑔 𝑔

Hình dưới đây là vẽ giá trị 𝜃 trong khoảng từ 0 đến 90 độ và vùng tương ứng cho mỗi góc. Từ đồ thị (hình bên dưới) biểu diễn giá trị 𝑅 theo sự thay đổi của góc 𝜃 với vận tốc ban đầu cố định là 𝑢 = 30𝑚/𝑠, chúng ta có thể nhận thấy miền đạt cực đại khi góc chiếu gần bằng 𝜃 = 45 , là góc có giá trị gần bằng = 0.785 (radian). Tuy nhiên, chúng ta sẽ đi tìm góc 𝜃 tối ưu nhất bằng phương pháp Gradient Ascent.



Thực hành: Vẽ đồ thị hàm 𝑹 để thấy được khoảng cách bay của “đạn” theo góc bắn.

>>> from sympy import Symbol, sin, pi

>>> u = Symbol('u')

>>> u = 30.0 # van toc 30m/s

>>> g = 9.81 # hang so gia toc trong truong

>>> theta = Symbol('theta')

>>> R = (u\*\*2)\*sin(2.0\*theta)/g

>>> R

91.743119266055\*sin(2.0\*theta)

>>> plot(R, (theta, 0, pi/2)) # xet goc ban tu 0 den 90 (do), tuong ung tu 0 den pi/2 (radian) <sympy.plotting.plot.Plot object at 0x000002481F642278>

Phương pháp Gradient ascent (GA) là phương pháp lặp: chúng ta bắt đầu với giá trị 𝜃 khởi đầu bằng 0.001, ngay khi đó, nó còn gọi là 𝜃 ũ = 0.001 và sau đó tăng dần để đạt được giá trị tương ứng vùng lớn hơn. Các bước được thể hiện theo phương trình như sau:

𝑑𝑅

𝜃 ớ = 𝜃 ũ + 𝜆

𝑑𝜃

Với giá trị 𝜆 là kích thước của bước (step size) và

### 𝑑𝑅

𝑑𝜃

là đạo hàm của 𝑅 theo 𝜃. Một khi chúng ta đặt 𝜃 ũ = 0.001, chúng ta phải thực hiện các bước sau:

1. Bước 1: Tính toán 𝜃 ớ bằng phương trình trên.
2. Bước 2: Nếu giá trị của biểu thức |𝜃 ớ − 𝜃 ũ| > 𝜀, với giá trị epsilon 𝜀 cho trước thì chúng ta đặt giá trị cũ bằng giá trị mới, nghĩa là 𝜃 ũ = 𝜃 ớ và tiếp tục quay về Bước 1.

Ngược lại, chúng ta tiếp tục Bước 3 dưới đây:

1. Bước 3: 𝜃 ớ là giá trị xấp xỉ của 𝜃 mà ở đó 𝑅 đạt giá trị lớn nhất.

Từ quy trình xử lý trên, chúng ta nhận thấy vai trò quan trọng nằm ở việc chọn giá trị epsilon. Trong thuật toán xử lý bên dưới, chúng ta đặt giá trị epsilon là số 1 , thể hiện trong Python là giá trị 1e-6. [Vai trò của giá trị epsilon sẽ được đề cập trong bài Thực hành sau].

Dưới đây là đoạn code minh họa:

Thực hành 9: Tìm giá trị cực đại của hàm bằng phương pháp Gradient Ascent

>>> import math

>>> from sympy import Derivative, Symbol, sin

>>> def grad\_ascent(x0, f1x, x):

epsilon = 1e-6 step\_size = 1e-4 x\_cu = x0

x\_moi = x\_cu + step\_size \* f1x.subs({x: x\_cu}).evalf() while abs(x\_cu - x\_moi) > epsilon:

x\_cu = x\_moi

x\_moi = x\_cu + step\_size \* f1x.subs({x: x\_cu}).evalf() return x\_moi

>>> def tim\_max\_theta(R, theta):

R1theta= Derivative(R, theta).doit()

theta\_0 = 1e-3

theta\_max = grad\_ascent(theta\_0, R1theta, theta) return theta\_max

>>> g = 9.8

>>> u = 30

>>> theta = Symbol('theta')

>>> R = (u\*\*2) \* sin(2\*theta) / g

>>>

>>> theta\_max = tim\_max\_theta(R, theta)

>>> print ('Goc theta: {0}'.format(math.degrees(theta\_max)))

…………………………………………………………………………  Sinh viên điền kết quả

>>> print ('Khoang cach vat bay xa: {0}'.format(R.subs({theta:theta\_max})))

…………………………………………………………………………  Sinh viên điền kết quả

Sau đó, chúng ta có thể đưa vào chương trình giả định lưu vào tập tin grad\_asc.py trong đĩa D:

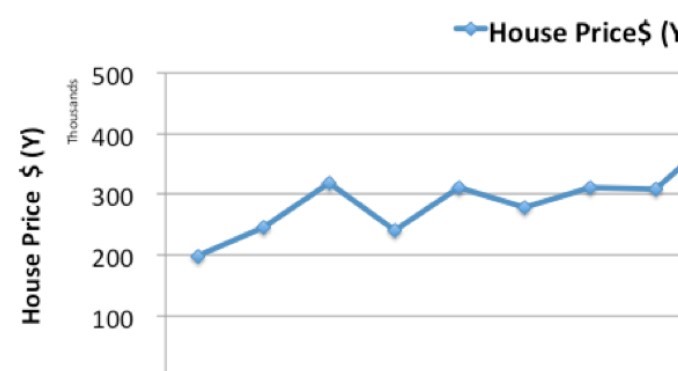
Kết quả thực thi chương trình:



Giá trị 𝜃 được in theo độ và tính ra gần bằng 45 . Nếu thay đổi vận tốc ban đầu thành giá trị khác, chúng ta sẽ thấy rằng giá trị 45 độ sẽ không thay đổi.

Bình luận:

* Tuy vậy, Giảng viên cần nói rõ với Sinh viên rằng: Trên thực tế, do các lực ma sát với không khí, sức cản của gió,… nên các ứng dụng về bắn đại bác hoặc vận động viên trong khi thi phóng lao ở các kì thế vận hội Olympic luôn phải chọn góc lớn hơn 45 độ, khoảng 50 – 55 độ!
* Phương pháp Gradient Ascent hoặc Gradient Descent được ứng dụng nhiều trong hệ thống Học Máy (Machine Learning), đặc biệt đối với các ứng dụng tối ưu và phân loại. Các dạng bài toán thường gặp là: từ dữ liệu trong quá khứ về các yếu tố ảnh hưởng đến giá cả hàng hóa, người ta sẽ tìm ra yếu tố nào thay đổi trong cuộc sống thì sẽ tác động đến giá cả của hàng hóa nào: như cháy rừng Amazon sẽ làm mạnh hàng hóa nào trên thị trường: kim loại quý, tiền tệ, dầu mỏ,…?; hoặc xác định chi phí thuê nhà hợp lý theo diện tích (như hình bên dưới).



# BÀI TẬP CHƯƠNG 3

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

Bài tập 1: Vẽ đồ thị ném vật thể với các góc ban đầu khác nhau, vận tốc không đổi.

Bài tập 2: Vẽ đồ thị định luật hấp dẫn Newton giữa hai vật thể

Biết rằng phương trình của định luật vạn vật hấp dẫn Newton là:

𝐺𝑚 𝑚

𝐹 =

𝑟

Giả sử chúng ta có hai vật có khối lượng 𝑚 = 0.5𝑘𝑔; 𝑚 = 1.5𝑘𝑔

Hằng số hấp dẫn 𝐺 = 6.674 × 10 𝑁𝑚 𝑘𝑔 .

Hãy vẽ đồ thị lực hấp dẫn của 2 vật thể ở các khoảng cách: 100, 150, 200, 250, 300,… đến 1000 mét.

Kết quả đồ thị để Sinh viên tham khảo:

