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 4: TÍCH PHÂN HÀM SỐ 3
1. Hàm linspace của numpy 3
2. Giới thiệu cơ bản về lập trình hàm 7
2.1. Hàm ẩn/vô danh lambda 7
2.2. Ứng dụng cơ bản của hàm lambda 7
2.3. Giới thiệu cơ bản về lập trình hàm 8
3. Tích phân của các hàm số 10
3.1. Việc tính tích phân 10
3.2. Tích phân với gói phần mềm scipy 13
3.3. Tích phân với gói sympy 14
3.4. Một ví dụ về hàm mật độ xác suất 16
BÀI TẬP CHƯƠNG 4 19

CHƯƠNG 4: TÍCH PHÂN

Mục tiêu:

- *Sơ lược về numpy, giới thiệu thêm các tính năng của gói Anaconda, phong cách lập trình*
- *Bổ túc cơ bản về lập trình Python: vẽ đồ thị, hàm ẩn/vô danh lambda*
- *Giới thiệu về lập trình hàm dạng cơ bản*
- *Tích phân của hàm số.*

Nội dung chính:

1. Hàm linspace của numpy

Thư viện numpy và hàm linspace để tạo các số thực đều nhau

Trong các bài trước, chúng ta đã làm quen với hàm **range** cho để tạo 1 khoảng đều các số tự nhiên. Và sau đó, chúng ta làm quen với việc viết bổ sung hàm (**frange**) hỗ trợ cho việc tạo khoảng đều số thực. Trong bài này, chúng ta sẽ sử dụng hàm của linspace của gói tính toán **numpy**.

Cùng với các gói **sympy, scipy**,…, gói **numpy** là một trong những gói hỗ trợ nhiều cho tính toán và xử lý dữ liệu. Tuy vậy, khác với sympy, numpy không tập trung vào việc giải toán hình thức (công thức) mà là giải cụ thể ra các con số. **Numpy** còn được xem là thư viện trung gian và chuẩn về cấu trúc dữ liệu để các thư viện xử lý như sympy, scipy tương tác dữ liệu (ma trận, dãy số, lưới, đa thức, vector, số phức… cũng như nhiều hàm xử lý).

```
>>> import bokeh
>>> import seaborn
>>> import matplotlib
>>> import sklearn # goi ten la Scikit-Learn
>>> import pandas
>>> import scipy
>>> import numpy
>>>
```

Hình ảnh về các gói xử lý khoa học dữ liệu được gói Anaconda hỗ trợ

Thư viện **numpy** hỗ trợ hàm **linspace** để tạo ra các dãy số thực đều nhau trong một khoảng. Đầu ra của lệnh là một kiểu dữ liệu dạng **array** của numpy được các gói phần mềm trên hỗ trợ.

Ví dụ: Để chia miền 0 đến 2 thành 10 miền (11 giá trị), chúng ta thực hiện lệnh linspace như sau:

Thực hành 1: Chia khoảng dữ liệu

```
>>> import numpy as np
>>> mienchia = np.linspace(0, 2, 11)
>>> mienchia
…… ← sinh viên điền vào.
```

Kết hợp **linspace** của numpy để vẽ các đồ thị hình sin bằng matplotlib:

Thực hành 2: Vẽ các đồ thị

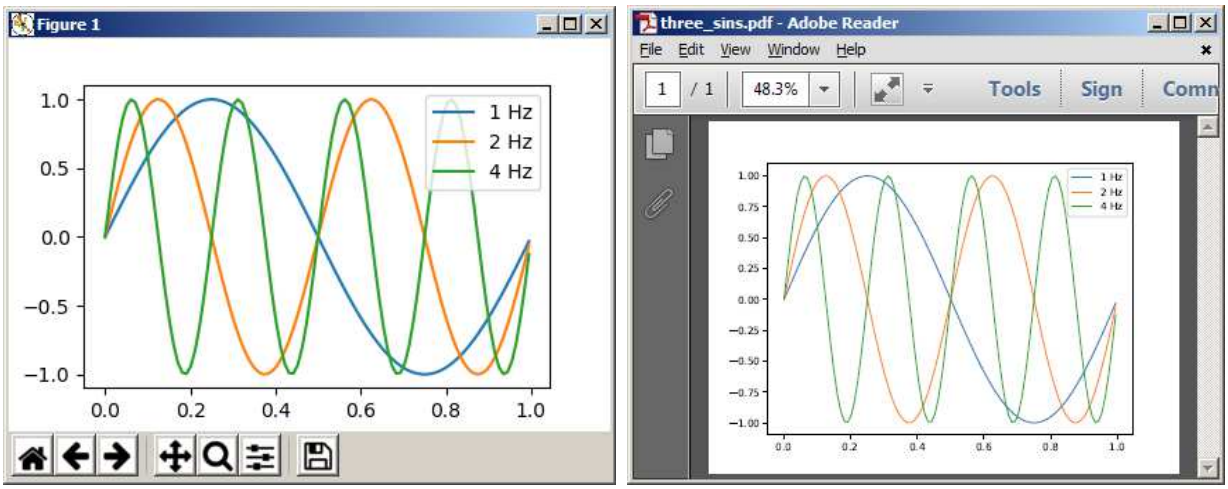
```
import numpy as np
import matplotlib.pyplot as plt

fs = [1, 2, 4]
all_time = np.linspace(0, 2, 200)
t = all_time[:100]

for f in fs:
        y = np.sin(2 * np.pi * f * t)
        plt.plot(t, y, label='{} Hz'.format(f))

plt.legend()
import os
os.chdir('d:\\')
plt.savefig('basics_python.pdf')
plt.show()
```

Kết quả là đồ thị và tập tin pdf được xuất ra:



Lưu ý: trong một số báo cáo khoa học, việc sử dụng tập tin pdf là bắt buộc do ảnh không bị vỡ khi phóng to hoặc thu nhỏ. Ngoài ra, định dạng pdf là định dạng của nhiều hệ điều hành như: Windows, Linux/Unix,…

Cũng nói thêm về phong cách/tư duy viết chương trình

Chúng ta hãy tưởng tượng rằng "nơi vẽ đồ thị" giống như một tấm bảng lớn. Như vậy, khi trình diễn đồ thị có hai cách khác nhau có thể có là:

- **Trình diễn dạng đối tượng (object-oriented approach):** vẽ các đồ thị lên các tờ giấy hoặc bảng con rồi dán chúng lên bảng lớn. Mỗi tờ giấy như một "bảng con". Ưu điểm của phương pháp là tách sự quản lý hình ảnh với bảng lớn. Trong một số trường hợp, chúng ta chỉ cần điều chỉnh các bảng nhỏ. Tư duy của phương pháp lập trình này có trong các ngôn ngữ lập trình hiện đại như: Java, C++, C#
- **Trình diễn dạng "máy trạng thái" (state-machine approach):** vẽ từng đồ thị lên bảng lần lượt đồ thị này đến đồ thị khác. Ưu điểm của phương pháp này là sự trực quan và tương thích với các ngôn ngữ lập trình kinh điển khác như: Matlab (để dễ dàng chuyển đổi code).

Như vậy, với hai cách bên trên là chúng ta cũng phải xem xét từng đồ thị. Nghĩa là chúng ta phải có một vòng lặp duyệt từng đồ thị. Nhưng khác nhau ở chỗ: giải pháp hướng đối tượng sẽ "vẽ" lên các bảng con còn phương pháp state-machine sẽ "vẽ" ngay vào "bảng". Ở đây, bảng chính là đối tượng **plt** của thư viện matplotlib.

Từ đó, xét 2 đoạn mã với điểm khác biệt chính là việc sử dụng vòng lặp for để vẽ:

- Giải pháp hướng đối tượng: Vẽ các hình như là các đối tượng "con" của **plt**:

…….

fig, ax = plt.subplots()

for f **in** fs:

y = **np.sin**(2 * np.pi * f * t)

ax.plot(t, y, label='{} Hz'.format(f))

……

- Giải pháp hướng máy trạng thái: Vẽ các hình trực tiếp lên trên đối tượng **plt**:

……

for f **in** fs:

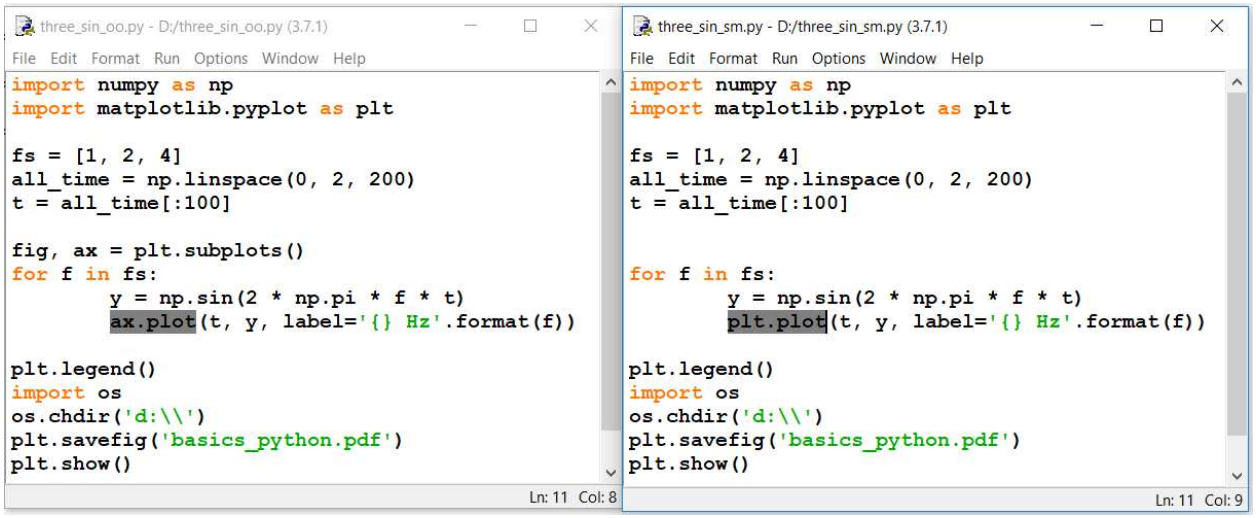
y = **np.sin**(2 * np.pi * f * t)

plt.plot(t, y, label='{} Hz'.format(f))

……

Sinh viên tham khảo hình ảnh hai chương trình:

Thực hành 3: Xây dựng chương trình hướng đối tượng



Tư duy **hướng đối tượng** cũng là một điểm mới mà người làm toán cần phải nắm để thể hiện mô hình. Đó là phương pháp ưu việt giúp các chương trình có thể kết hợp với nhau để tạo ra các phần mềm lớn.

2. Giới thiệu cơ bản về lập trình hàm

Trong bài trước, chúng ta làm quen với việc xử lý tính toán giá trị chuỗi bằng hàm **eval()**. Tuy nhiên, với một số mô hình tính toán, chúng ta cần phải mở rộng để xử lý.

2.1. Hàm ẩn/vô danh lambda

Hàm **lambda** trong Python: có từ phiên bản 2.2. Gọi là hàm "ẩn danh" (anonymous function) **trong lúc chạy**. Hàm lambda không có lệnh **return** trả về như các module của Python nhưng nó luôn luôn thực hiện các lệnh trong đó để trả về giá trị.

Ví dụ: xây dựng hàm tính x^2 theo 2 phương pháp: viết hàm def thông thường và viết hàm theo dạng hàm lambda:

Thực hành 4: Minh họa cơ bản về hàm lambda

```
>>> def f (x): return x**2
...
>>> print (f(8))
64
>>>
>>> g = lambda x: x**2
>>>
>>> print (g(8))
64
```

2.2. Ứng dụng cơ bản của hàm lambda

Khi xây dựng một ứng dụng nhiều cấp, một số tham số có thể thay đổi ở các cấp cụ thể. Tuy nhiên, về mặt công thức tính toán sẽ giữ như nhau. Ví dụ: việc tính thuế để mua xe,… là công thức chung áp dụng cho cả nước. Nhưng ở một số nơi như Hà Nội và TP.HCM có điều chỉnh theo các điều kiện như: để giảm tắt nghẽn giao thông hoặc để giảm ô nhiễm.

Xét ứng dụng cơ bản: *Bài toán tính thuế ở địa phương như sau: Giả sử thuế ở TPHCM là 0.012 (nghĩa là 1.2%); ở Hà Nội là 0.01 (nghĩa là 1%). Hãy viết hàm tính (lưu ý: hệ thống sẽ mở rộng nhiều địa phương).*

Thực hành 5: Tính thuế theo từng khu vực

>>> **def** thue(phan_tram): **return** ***lambda*** x**:** x * phan_tram

>>> hcm = thue(0.012)

>>> hn = thue(0.01)

>>> hcm(1000000)

…….. ← Sinh viên điền kết quả vào

>>> hn(1000000)

…….. ← Sinh viên điền kết quả vào

```
>>> def thue(phan_tram): return lambda x: x * phan_tram

>>> hcm = thue(0.012)
>>> hn = thue(0.01)
>>> hcm(1000000)
12000.0
>>> hn(1000000)
10000.0
```

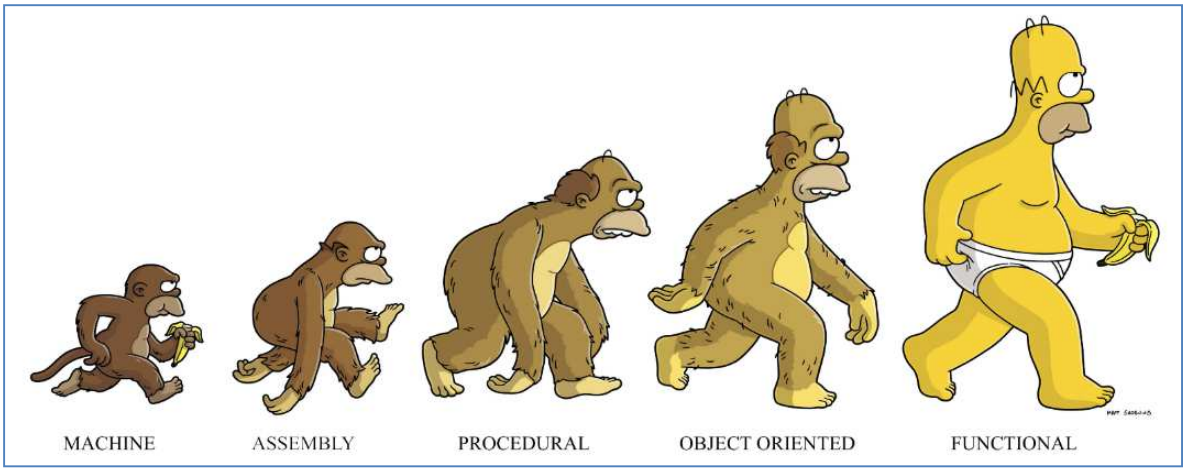
Cùng với cơ chế **lambda**, cơ chế **reduce**, **map**, **filter** cũng là những hàm rất quan trọng và đặc biệt của Python nói riêng và của một nhánh lập trình gọi là **lập trình hàm** (functional programming) nói chung. Lập trình hàm là một phương pháp lớn, cần nhiều thời gian để nghiên cứu. Nhìn chung, lập trình hàm và lập trình hướng đối tượng là những cách thức lập trình mới. Mục 2.3 chỉ mô tả sơ lược, dạng giới thiệu là chính.

Tài liệu tham khảo thêm về các hàm reduce, map, filter của Python:

- https://vimentor.com/vi/lesson/19-lambda-filter-reduce-and-map (truy cập 10/2019)

2.3. Giới thiệu cơ bản về lập trình hàm

Theo Wikipedia, trong ngành khoa học máy tính, lập trình hàm là một mô hình lập trình xem việc tính toán là sự đánh giá các hàm toán học và tránh sử dụng trạng thái và các dữ liệu biến đổi. Lập trình hàm nhấn mạnh việc *ứng dụng hàm số*, trái với phong cách lập trình mệnh lệnh, nhấn mạnh vào sự thay đổi trạng thái.



Một hình ảnh hài hước minh họa các giai đoạn xuất hiện các phương thức lập trình: từ ngôn ngữ máy (machine) đến lập trình hàm (functional).

Lập trình hàm xuất phát từ phép tính lambda, một hệ thống hình thức được phát triển vào những năm 1930 để nghiên cứu định nghĩa hàm số, ứng dụng của hàm số, và đệ quy. Nhiều ngôn ngữ lập trình hàm có thể được xem là những cách phát triển giải tích lambda. Một góc nhìn về lập trình hàm (functional programming) được trình bày như sau:

Functional Programming là phương pháp lập trình lấy function làm đơn vị thao tác cơ bản.

Từ đó, một số nguyên tắc của lập trình hàm:

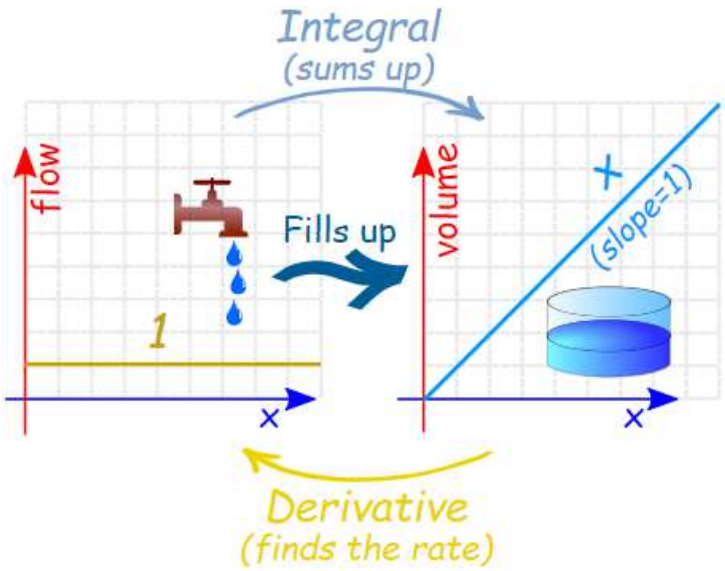
- Nguyên tắc thứ nhất trong Functional Programming là: Thứ gì đã khai báo một lần thì mãi mãi như vậy, không bao giờ thay đổi nữa. Các biến hoặc đối tượng trong kịch bản Functional Programming nếu có thì phải ***immutable*** (có tính bất biến).
- Nguyên tắc thứ hai: Trong Functional Programming: **tất cả các hàm** đều phải là ***pure function***, không có hiệu ứng phụ (side effect), không được tác động lên bất cứ giá trị nào bên ngoài nó, cũng nói không với chỉnh sửa tham số input. Lưu ý: purity là tính thuần khiết, thuần túy, sự trong sạch, không bị pha tạp. Ví dụ: hàm tính toán có sử dụng giá trị ngày giờ hệ thống sẽ không đảm bảo tính purity vì kết quả có thể sẽ khác nhau.

Tài liệu tham khảo:

- https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_h%C3%A0m
- https://vnoi.info/wiki/translate/functional-programming-part-1

3. Tích phân của các hàm số

Tích phân về mặt ý niệm là "góp nhặt" (tổng) các "lát cực nhỏ" để được tổng thể đối tượng (nguyên văn "Integration is a way of adding slices to find the whole", theo https://www.mathsisfun.com/calculus/integration-introduction.html). Như vậy, giữa đạo hàm với tích phân có quan hệ ngược nhau. Cụ thể là: Giả định có 1 vòi chảy nước vào trong hồ.



Khi đó chúng ta có thể phát biểu theo từng gốc độ:

- Đạo hàm: Nếu khối nước cứ tăng đều x trong một đơn vị thời gian thì nghĩa là hệ số nước đều đều không thay đổi (bằng) 1.
- Tích phân: Với tốc độ chảy nước cứ giữ bằng nhau tại mọi thời điểm trong một đơn vị thời gian thì nước sẽ tăng đều một giá trị khối lượng nước x nào đó. Hiển nhiên, để mô tả đầy đủ, chúng ta phải bổ sung thêm giá trị C (trong tích phân) là đối tượng không bắt buộc

Một vòi nước chảy nước đều vào trong hồ chứa, nghĩa là cứ mỗi đơn vị thời gian sẽ có 1 đơn vị nước chảy vào trong hồ chứa làm hồ chứa tăng đúng 1 khối lượng là x.

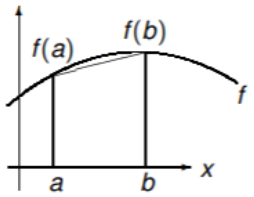
3.1. Việc tính tích phân

Tích phân vô định hoặc bất định (indefinite integral, có sách tiếng Anh ghi là antiderivative) của một hàm số $f(x)$ là là số $F(x)$, thỏa $F'(x) = f(x)$. Chúng ta có thể hiểu nôm na là tích phân của một hàm số là một hàm mà đạo hàm của nó chính là hàm ban đầu. Trong toán học, kí hiệu tích

phân là $F(x) = \int f(x)dx$. Với tích phân xác định (definite integral), chúng ta có thêm các cận (giả định là a và b):

$$\int_a^b f(x)dx = F(b) - F(a)$$

Với $F(b)$ và $F(a)$ là những giá trị tích phân tại các vị trí $x = b$ và $x = a$ tương ứng.



$$\int_a^b f(x)dx \approx \left(\frac{b-a}{2}\right)(f(a) + f(b))$$

Để tính toán tích phân, nguyên lý cơ bản là phương pháp tính diện tích hình thang (the trapezoidal rule). Chúng ta thực hiện việc xây dựng hàm tính toán **lambda** chung và sử dụng minh họa cho hàm $\boldsymbol{e^x}$:

Thực hành 6: Biểu diễn cài đặt phương pháp hình thang với hàm lambda

```
>>> hinhthang = lambda f, a, b: (b-a)*(f(a)+f(b))/2

>>> from math import exp

>>> hinhthang(exp,0, 1)

1.8591409142295225

>>> def hai_x(x):

        return 2*x


>>> hinhthang(hai_x, 1, 2)
```

…… ← Sinh viên điền kết quả.

Thực hành 7: Xây dựng hàm tính tích phân cho e^x bằng phương pháp hình thang

```
"""
Ham tinh theo luat hinh thang
cho ham(x) tu [trai, phai]
"""
def traprule(ham, trai, phai):
    return (phai-trai)*(ham(trai) + ham(phai))/2


import math
S = "Bieu thuc can tinh tich phan la e^x:"
print(S + '[a,b]')
A = float(input('Nhap a : '))
B = float(input('Nhap b : '))
Y = traprule(math.exp, A, B)
print(S + "[%.1E,%.1E] : " % (A, B))
print('Gia tri xap xi : %.15E' % Y)
E = math.exp(B) - math.exp(A)
print(' Gia tri chinh xac : %.15E' % E)
```

Kết quả thực thi:

```
========== RESTART: D:/traprule.py ==========
Bieu thuc can tinh tich phan la e^x:[a,b]
Nhap a : 0
Nhap b : 1
```

Bieu thuc can tinh tich phan la e^x:[0.0E+00,1.0E+00] :

Gia tri xap xi : …….…….. ← *sinh viên nhập vào*

Gia tri chinh xac : …….. ← *sinh viên nhập vào*

Với phần mềm ngày nay, ngoài việc tự xây dựng chương trình, việc tính tích phân có thể thực hiện trên nhiều gói phần mềm khác nhau như scipy, sympy.

3.2. Tích phân với gói phần mềm scipy

Trong scipy, việc tính toán tích phân được sử dụng thông qua hàm **integrate**.

Thực hành 8: Viết các lệnh sau và thực thi cùng lúc trong tập tin scipy_in.py:

```
import scipy

from sympy import *

import sys

sys.displayhook = pprint

x = Symbol('x')

bt1 = integrate(x**2 + x + 1, x)

pprint (bt1)

bt2 = integrate(x/(x**2+2*x+1), x)

pprint (bt2)

bt3 = integrate(x**2 * exp(x) * cos(x), x)

pprint (bt3)

bt4 = integrate(exp(-x**2)*erf(x), x)

pprint (bt4)
```

Kết quả thực thi:

…… ← Sinh viên điền

……

……

……

3.3. Tích phân với gói sympy

Với gói SymPy, chúng ta có thể tìm thấy việc tính toán cả hai loại tích phân bằng cách tạo đối tượng **Integral**.

Dưới đây là minh họa việc tính tích phân $\int kxdx$ với k là một hằng số.

Thực hành 9: Tính tích phân đơn giản

>>> **from** sympy **import** Integral, Symbol

>>> x = Symbol('x')

>>> k = Symbol('k')

>>> Integral(k*x, x)

…… ← Sinh viên điền vào

```
>>> from sympy import Integral, Symbol
>>> x = Symbol('x')
>>> k = Symbol('k')
>>> Integral(k*x, x)
Integral(k*x, x)
```

Sau khi import các lớp Integral và Symbol và thực hiện việc tạo 2 đối tượng tương ứng k và x. Sau đó, chúng ta tạo đối tượng Integral với hàm kx và xác định biến lấy tích phân là x. Tương tự như các lớp Limit (tính giới hạn) và Derivative (tính đạo hàm), chúng ta sẽ phải thực hiện việc tính toán sử dụng phương thức **doit()**:

>>> Integral(k*x, x).doit()

…… ← Sinh viên điền vào

Hoặc chúng ta có thể "trình bày" kết quả đẹp hơn với lệnh **pprint** của Sympy:

Thực hành 10: Tính tích phân đơn giản

>>> **from** sympy **import** pprint**,** Integral

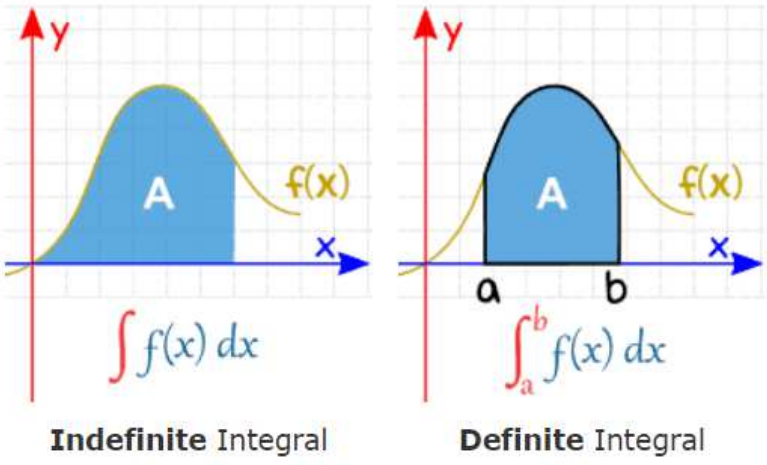
>>> F = **Integral**(k*x, x).**doit**()

>>> **pprint** (F)

…… ← Sinh viên điền vào

……

Giá trị tích phân trả về là một hàm số (kí hiệu). Nếu chúng ta tính đạo hàm, nó sẽ ra giá trị của hàm gốc là hàm kx.



Để tính tích phân xác định, chúng ta chỉ đơn giản thêm các giá trị biến xác định cận dưới và cận trên khi tạo đối tượng Integral, cụ thể như sau:

Thực hành 11: Tính tích phân đơn giản

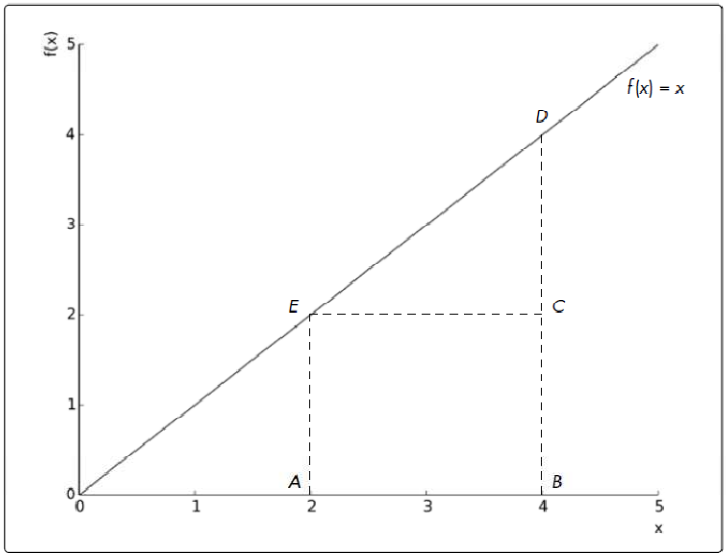
>>> **Integral**(k*x, (x, 0, 2)).doit()

…… ← Sinh viên điền vào

Giá trị trả về là tích phân xác định

$$\int_0^2 kxdx$$

Chúng ta có thể thể hiện trực quan các tích phân xác định bằng việc thể hiện hình học. Xét hình bên dưới của đồ thị $f(x) = x$ với giá trị x nằm giữa $[0, 5]$.



Xét vùng ABDE trong đồ thị trên nằm giữa 2 điểm từ $x = 2$ đến $x = 4$ tương ứng với điểm A và B. Diện tích ABDE có thể được tính bằng cách tính diện tích các vùng hình học, cụ thể:

$$S_{ABDE} = S_{ABCE} + S_{ECD} = 2 \times 2 + \left(\frac{1}{2}\right) \times 2 \times 2 = 6$$

Thử lại, chúng ta có thể tính tích phân $\int_2^4 xdx$ bằng hàm xử lý **Integral** của Sympy:

Thực hành 12: Tính tích phân đơn giản

```
>>> from sympy import Integral, Symbol
>>> x = Symbol('x')
>>> Integral(x, (x, 2, 4)).doit()
…… ← Sinh viên điền vào.
```

Giá trị tích phân tương đồng như việc tính toán diện tích ABDE.

Việc hiểu về tích phân xác định là diện tích "đóng" bởi hàm giữa các điểm xác định trên trục x là vấn đề chính yếu để hiểu được những tính toán về xác suất trong các sự kiện ngẫu nhiên liên quan các biến ngẫu nhiên liên tục.

3.4. Một ví dụ về hàm mật độ xác suất

Giả định lớp toán thực hiện một bài thi. Điểm trong bài thi đạt từ 0 đến 20, bao gồm các điểm phân số. Nếu chúng ta xem việc điểm là một sự kiện ngẫu nhiên, thì điểm được xem như là *biến*

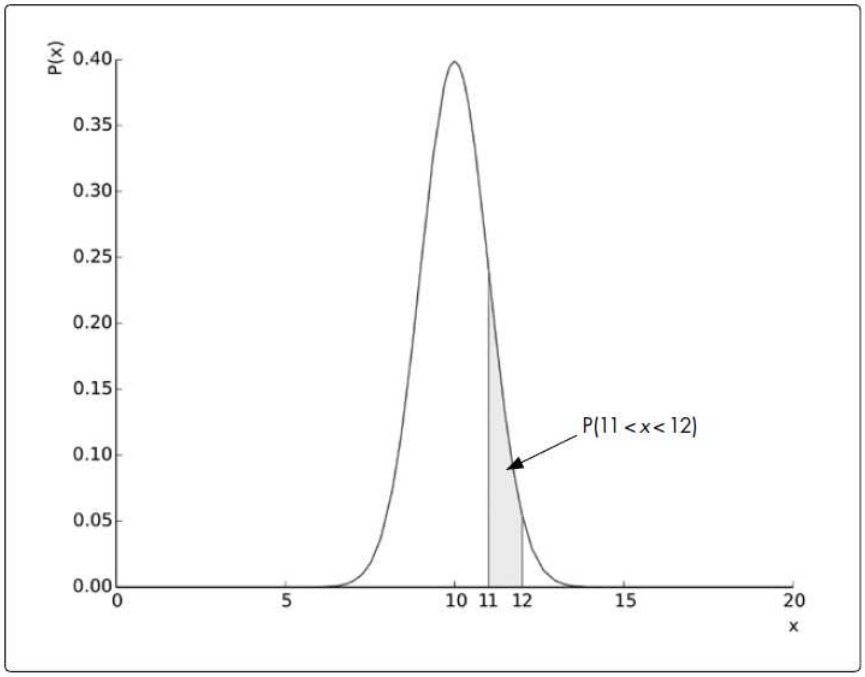
liên tục ngẫu nhiên vì điểm số bao gồm các giá trị từ 0 đến 20. Nếu chúng ta muốn tính toán xác suất của sinh viên có điểm từ 11 đến 12 thì chúng ta có thể vận dụng cách tính (trên lý thuyết):

$$P(11 < x < 12) = \frac{n(E)}{n(S)}$$

với E là tập các điểm có thể giữa 11 và 12 và S là tập tất cả các khả năng có thể có, nghĩa là tất cả các con số (điểm) có thể có giữa 1 và 20. Từ định nghĩa từ trước, $n(E)$ là không thể xác định vì không thể đếm hết tất cả các trường hợp thực sự điểm nằm giữa 11 và 12; điều tương tự như $n(S)$. Do đó, chúng ta cần giải pháp khác để tính xác suất.

Hàm mật độ xác suất $P(x)$ thể hiện xác suất của một giá trị ngẫu nhiên có mối quan hệ với x, với x tùy ý. Hàm có thể cho chúng ta biết xác suất để "rơi" vào một khoảng tùy ý. Rằng: nếu biết hàm mật độ xác suất thể hiện xác suất điểm của lớp thì việc tính toán xác suất $P(11 < x < 12)$ sẽ dễ dàng. Tuy nhiên, làm cách nào để tính toán xác suất đó?

Khi vẽ đồ thị, dễ dàng thấy rằng đó là diện tích của hàm mật độ xác suất và trục x từ điểm $x = 11$ đến điểm $x = 12$ như hình bên dưới:



Chúng ta biết rằng diện tích trên bằng với tích phân:

$$\int_{11}^{12} p(x)dx$$

do đó, chúng ta sẽ tìm xác suất giữa hai giá trị 11 và 12. Bằng phương pháp toán học, chúng ta có thể tìm ra xác suất này. Hàm mật độ xác suất **được giả định** trong hàm bên trên là hàm:

$$\frac{1}{\sqrt{2\pi}} e^{-\frac{(x-10)^2}{2}}$$

với x là điểm được có thể nhận được. Hàm này (giả định) được lựa chọn để tính xác suất của điểm gần với điểm 10 (hoặc lân cận 10) là cao nhưng sẽ có sự sụp giảm nhanh chóng khi tăng hoặc giảm về 2 phía.

Bây giờ, chúng ta có thể tính tích phân:

$$\int_{11}^{12} p(x)dx$$

với $p(x)$ được cho như hàm bên trên. Khi đó, các lệnh tính toán bằng Python là:

Thực hành 13: Tính tích phân hàm mật độ xác suất

```
>>> from sympy import Symbol, exp, sqrt, pi, Integral
>>> x = Symbol('x')
>>> p = exp(-(x - 10)**2/2)/sqrt(2*pi)
>>> Integral(p, (x, 11, 12)).doit().evalf()
```

…… ← sinh viên ghi lại kết quả.

Chúng ta tạo đối tượng **Integral** cho hàm số, với p thể hiện hàm mật độ xác suất và chúng ta tính toán tích phân xác định giữa 11 và 12 trên trục x.

Sau đó hàm **doit()** sử dụng để thực hiện tính toán giá trị và hàm **evalf()** để thể hiện bằng số. Giá trị tính toán được gần bằng 0.14.

BÀI TẬP CHƯƠNG 4

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

Bài tập 1: Sinh viên đọc, tìm hiểu và viết báo cáo về tài liệu trên mạng sau:

http://homepages.math.uic.edu/~jan/mcs507/sympyscipyintegration.pdf

(lưu ý: có thể sử dụng kết hợp thêm tài liệu khác để bài viết được phong phú)

Bài tập 2: Sinh viên đọc, tìm hiểu và viết báo cáo về tài liệu trên mạng sau:

https://vi.wikipedia.org/wiki/Ph%C3%A9p_t%C3%ADnh_lambda

(lưu ý: có thể sử dụng kết hợp thêm tài liệu khác để bài viết được phong phú)