TRƯỜNG ĐẠI HỌC TRẦN ĐẠI NGHĨA

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO

BÀI TIỂU LUẬN MÔN HỌC

NHẬP MÔN XỬ LÝ ẢNH

Đề tài: "Giải bài tập môn Lập trình căn bản bằng ngôn ngữ lập trình Python"

TP. HỒ CHÍ MINH, THÁNG 10 NĂM 2019

TRƯỜNG ĐẠI HỌC TRẦN ĐẠI NGHĨA

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO

BÀI TIỂU LUẬN MÔN HỌC

NHẬP MÔN XỬ LÝ ẢNH

Đề tài: "Giải bài tập môn Lập trình căn bản bằng ngôn ngữ lập trình Python"

GVHD: Ngô Thanh Tú

Sinh viên thực hiện:

Nguyễn Lê Xuân Phước

Tôn Nữ Nguyên Hậu

Đỗ Tống Quốc

Lớp: 16DDS06031

TP. HỒ CHÍ MINH, THÁNG 10 NĂM 2019

MỤC LỤC

LỜI MỞ ĐẦU

Nếu bạn làm việc nhiều với máy vi tính, một lúc nào đó bạn sẽ nhận thấy bạn muốn tự động hóa một số việc. Ví dụ, bạn muốn thực hiện một phép tìm kiếm và thay thế với nhiều tập tin văn bản, hoặc đổi tên và sắp xếp một loạt các tập tin ảnh theo một cách phức tạp. Có thể bạn muốn viết cơ sở dữ liệu tùy biến nho nhỏ, hoặc một ứng dụng với giao diện đồ họa đặc biệt, hay một trò chơi đơn giản.

Nếu bạn là một người chuyên viết phần mềm, bạn có thể làm việc với nhiều thư viện C/C++/Java nhưng bạn nhận thấy thường lặp đi lặp lại việc viết/biên dịch/thử/biên dịch là quá tốn thời gian. Có thể bạn viết một bộ các thử nghiệm cho các thư viện ấy và nhận ra rằng viết mã lệnh để thử nghiệm là một việc chán ngấy. Hoặc có thể bạn viết một chương trình cần sử dụng một ngôn ngữ mở rộng, và bạn không muốn thiết kế, xây dựng cả một ngôn ngữ mới cho ứng dụng của mình.

Python chính là ngôn ngữ lập trình bạn cần.

Python cho phép viết các chương trình nhỏ gọn và dễ hiểu. Các chương trình viết bằng Python thường ngắn hơn so với các chương trình viết bằng C, C++ hoặc Java, vì nhiều lý do:

- Các kiểu dữ liệu cao cấp cho phép bạn thực hiện nhanh các thao tác phức tạp chỉ với một lệnh đơn giản;
- Phát biểu lệnh được nhóm lại bằng khoảng cách thụt đầu dòng thay vì đóng mở với các dấu ngoặc;
- Không cần khai báo biến hoặc tham số trước khi sử dụng.

Python là ngôn ngữ hướng đối tượng được ứng dụng rất đa dạng. Vì vậy, nhóm chúng em sẽ làm báo cáo thực hiện giải các dạng bài tập lập trình căn bản bằng ngôn ngữ Python để có thể hiểu rõ hơn về cấu trúc, kiến trúc, các hàm, các câu lệnh, các thư viện được hỗ trợ và các tính năng khác mà ngôn ngữ lập trình Python đem lại cho chúng ta.

Do kinh nghiệm và kiến thức chưa được sâu sắc nên trong báo cáo về đề tài của nhóm mong thầy góp ý thêm để nhóm có thể hoàn thiện tốt hơn các đề tài nghiên cứu về sau.

Chúng em xin chân thành cảm ơn !

CHƯƠNG I. GIỚI THIỆU NGÔN NGỮ PYTHON

Python là một ngôn ngữ lập trình thông dịch do Guido van Rossum tạo ra năm 1990. Python hoàn toàn tạo kiểu động và dùng cơ chế cấp phát bộ nhớ tự động; do vậy nó tương tự như Perl, Ruby, Scheme, Smalltalk, và Tcl. Python được phát triển trong một dự án mã mở, do tổ chức phi lợi nhuận Python Software Foundation quản lý.

Theo đánh giá của Eric S. Raymond, Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình. Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu, như nhận định của chính Guido van Rossum trong một bài phỏng vấn ông.

Ban đầu, Python được phát triển để chạy trên nền Unix. Nhưng rồi theo thời gian, nó đã "bành trướng" sang mọi hệ điều hành từ MS-DOS đến Mac OS, OS/2, Windows, Linux và các hệ điều hành khác thuộc họ Unix. Mặc dù sự phát triển của Python có sự đóng góp của rất nhiều cá nhân, nhưng Guido van Rossum hiện nay vẫn là tác giả chủ yếu của Python. Ông giữ vai trò chủ chốt trong việc quyết định hướng phát triển của Python.

1. Lịch sử phát triển

Python được hình thành vào cuối những năm 1980, và việc thực hiện nó vào tháng 12 năm 1989 bởi Guido van Russom tại Centrum Wiskunde & Informatica (CWI) ở Hà Lan như là một kế thừa cho ngôn ngữ ABC (tự lấy cảm hứng từ SETL) có khả năng xử lý ngoại lệ và giao tiếp với Hệ điều hành Amoeba. Van Rossum là tác giả chính của Python, và vai trò trung tâm của ông trong việc quyết định hướng phát triển của Python.

Sự phát triển Python đến nay có thể chia làm các giai đoạn:

Python 1: bao gồm các bản phát hành 1.x. Giai đoạn này, kéo dài từ đầu đến cuối thập niên 1990. Từ năm 1990 đến 1995, Guido làm việc tại CWI (Centrum voor Wiskunde en Informatica - Trung tâm Toán-Tin học tại Amsterdam, Hà Lan). Vì vậy, các phiên bản Python đầu tiên đều do CWI phát hành. Phiên bản cuối cùng phát hành tại CWI là 1.2.

Vào năm 1995, Guido chuyển sang CNRI (Corporation for National Research Initiatives) ở Reston, Virginia. Tại đây, ông phát hành một số phiên bản khác. Python 1.6 là phiên bản cuối cùng phát hành tại CNRI.

Sau bản phát hành 1.6, Guido rời bỏ CNRI để làm việc với các lập trình viên chuyên viết phần mềm thương mại. Tại đây, ông có ý tưởng sử dụng Python với các phần mềm tuân theo chuẩn GPL. Sau đó, CNRI và FSF

(Free Software Foundation - Tổ chức phần mềm tự do) đã cùng nhau hợp tác để làm bản quyền Python phù hợp với GPL. Cùng năm đó, Guido được nhận Giải thưởng FSF vì Sự phát triển Phần mềm tự do (Award for the Advancement of Free Software).

Phiên bản 1.6.1 ra đời sau đó là phiên bản đầu tiền tuân theo bản quyền GPL. Tuy nhiên, bản này hoàn toàn giống bản 1.6, trừ một số sửa lỗi cần thiết.

Python 2: vào năm 2000, Guido và nhóm phát triển Python dời đến BeOpen.com và thành lập BeOpen PythonLabs team. Phiên bản Python 2.0 được phát hành tại đây. Sau khi phát hành Python 2.0, Guido và các thành viên PythonLabs gia nhập Digital Creations.

Python 2.1 ra đời kế thừa từ Python 1.6.1 và Python 2.0. Bản quyền của phiên bản này được đổi thành Python Software Foundation License. Từ thời điểm này trở đi, Python thuộc sở hữu của Python Software Foundation (PSF), một tổ chức phi lợi nhuận được thành lập theo mẫu Apache Software Foundation.

Python 3, còn gọi là Python 3000 hoặc Py3K: Dòng 3.x sẽ không hoàn toàn tương thích với dòng 2.x, tuy vậy có công cụ hỗ trợ chuyển đổi từ các phiên bản 2.x sang 3.x. Nguyên tắc chủ đạo để phát triển Python 3.x là "bỏ cách làm việc cũ nhằm hạn chế trùng lặp về mặt chức năng của Python". Trong PEP (Python Enhancement Proposal) có mô tả chi tiết các thay đổi trong Python. Các đặc điểm mới của Python 3.0 sẽ được trình bày phần cuối bài này.

2. Phiên bản Python

❖ *CPython*

Đây là phiên bản đầu tiên và được duy trì lâu nhất của Python, được viết trong C. Những đặc điểm của ngôn ngữ mới xuất hiện đầu tiên từ đây

❖ *Jython*

Là phiên bản Python trong môi trường Java. Bản này có thể được sử dụng như là một ngôn ngữ script cho những ứng dụng Java. Nó cũng thường được sử dụng để tạo ra những tests thử nghiệm cho thư viện Java

❖ *Python for .NET*

Phiên bản này thật ra sử dụng phiên bản Cpython, nhưng nó là một ứng dụng .NET được quản lý, và tạo ra thư viện .NET sẵn dùng. Bản này được xây dựng bởi Brian Lloyd.

❖ ***IronPython***

Là một bản Python tiếp theo của .NET, không giống như Python.NET đây là một bản Python hoàn chỉnh, tạo ra IL, và biên dịch mã Python trực tiếp vào một tập hợp .NET.

❖ ***PyPy***

PyPy được viết trên Python, thậm chí cả việc thông dịch từng byte mã cũng được viết trên Python. Nó sử dụng Cpython như một trình thông dịch cơ sở. Một trong những mục đích của dự án cho ra đời phiên bản này là để khuyến khích sự thử nghiệm bản thân ngôn ngữ này làm cho nó dễ dàng hơn để sửa đổi thông dịch (bởi vì nó được viết trên Python).

3. Sự khác biệt của phiên bản 3.x và 2.x

❖ **Division operator**

Thay đổi này đặc biệt nguy hiểm nếu bạn đang thực thi code Python 3 trong Python 2, vì thay đổi trong hành vi phân chia số nguyên thường có thể không được chú ý (nó không làm tăng cú pháp SyntaxError).

```
print 7 / 5
print -7 / 5
Output in Python 2.x:
1
-2
Output in Python 3.x:
1.4
-1.4
```

❖ **Print function**

Đây là một trong những sự thay đổi được biết đến nhiều nhất từ bản Python 2.x lên Python 3.x. Python 2.x không có vấn đề với các lệnh bổ sung, nhưng ngược lại, Python 3.x sẽ tăng cú pháp SyntaxError nếu chúng ta gọi hàm in mà không có dấu ngoặc đơn.

```
print 'Hello, Geeks'    # Python 3.x doesn't support
print('Hope You like these facts')
Output in Python 2.x:
Hello, Geeks
Hope You like these facts

Output in Python 3.x:
File "a.py", line 1
    print 'Hello, Geeks'
                        ^
SyntaxError: invalid syntax
```

❖ Unicode

Trong Python 2.x, kiểu mặc định của String là ASCII, nhưng ở Python 3.x kiểu mặc định của String là Unicode và 2 lớp byte: byte và bytearrays.

```
print(type('default string '))
print(type(u'string with b '))
Output in Python 2.x (Unicode and str are different):
<type 'str'>
<type 'unicode'>

Output in Python 3.x (Unicode and str are same):
<class 'str'>
<class 'str'>
```

❖ Xrange

Việc sử dụng xrange () rất phổ biến trong Python 2.x để tạo một đối tượng có thể lặp lại, ví dụ: trong vòng lặp for hoặc list / set-dictionary-comprehension. xrange-iterable không phải hoàn toàn có nghĩa là bạn có thể lặp lại nó vô hạn.

Trong Python 3, range () đã được thực hiện giống như hàm xrange () sao cho hàm xrange () chuyên dụng không còn tồn tại nữa (xrange () tăng một NameError trong Python 3).

```
for x in xrange(1, 5):
    print(x),

for x in range(1, 5):
    print(x),
Output in Python 2.x:
1 2 3 4 1 2 3 4

Output in Python 3.x:
NameError: name 'xrange' is not defined
```

❖ Error Handling

Đây là một thay đổi nhỏ trên phiên bản 3.x, từ khoá as đã trở thành bắt buộc Kiểm tra không có từ khoá as trong 2 phiên bản Python

```
try:
    trying_to_check_error
except NameError, err:
    print err, 'Error Caused'    # Would not work in Python 3.x
Output in Python 2.x:
name 'trying_to_check_error' is not defined Error Caused


Output in Python 3.x :
```

```
File "a.py", line 3
    except NameError, err:
                    ^
SyntaxError: invalid syntax
```

Kiểm tra khi có từ khoá as trong 2 phiên bản Python

```
try:
     trying_to_check_error
except NameError as err: # 'as' is needed in Python 3.x
     print (err, 'Error Caused')
Output in Python 2.x:
(NameError("name 'trying_to_check_error' is not defined",), 'Error
Caused')

Output in Python 3.x:
name 'trying_to_check_error' is not defined Error Caused
```

❖ The next() function and next() method

Next() hay (.next ()) là một hàm thường được sử dụng (method), đây là một thay đổi cú pháp khác (hay đúng hơn là thay đổi trong thực thi) sử dụng tốt trong python 2 nhưng bị loại bỏ trong python3

Vd: python 2.x:

```
my_generator = (letter for letter in 'abcdefg')

next(my_generator)
my_generator.next()
--------------------------------
result:
'a'
'b'
```

python3.x:

```
my_generator = (letter for letter in 'abcdefg')
next(my_generator)

--------------------------------
'a'

my_generator.next()

---------------------------------------------------------------------
--
AttributeError              Traceback (most recent call last)

<ipython-input-14-125f388bb61b> in <module>()
----> 1 my_generator.next()
```

```
AttributeError: 'generator' object has no attribute 'next'
```

❖ Các biến vòng lặp và rò rỉ namespace chung

Trong Python 3.x cho các biến vòng lặp không bị rò rỉ vào không gian tên chung nữa List comprehensions không còn hỗ trợ mẫu cú pháp [... for var in item1, item2, ...]. Sử dụng [... for var in (item1, item2, ...)] thay thế. Cũng lưu ý rằng việc hiểu danh sách có ngữ nghĩa khác nhau: chúng gần với cú pháp cho biểu thức máy hiểu bên trong một hàm tạo danh sách (và đặc biệt là các biến điều khiển vòng lặp không còn bị rò rỉ ra ngoài phạm vi. ”

Vd: Python 2.x

```
i = 1
print 'before: i =', i

print 'comprehension: ', [i for i in range(5)]

print 'after: i =', i
--------------------------------
before: i = 1
comprehension:  [0, 1, 2, 3, 4]
after: i = 4
```

Python 3.x:

```
i = 1
print('before: i =', i)

print('comprehension:', [i for i in range(5)])

print('after: i =', i)
--------------------------------

before: i = 1
comprehension: [0, 1, 2, 3, 4]
after: i = 1
```

❖ So sánh khác loại

Ở python 2 ,có thể so sánh list với string,... nhưng ở python 3 thì không

Vd: python 2.x

```
print "[1, 2] > 'foo' = ", [1, 2] > 'foo'
print "(1, 2) > 'foo' = ", (1, 2) > 'foo'
print "[1, 2] > (1, 2) = ", [1, 2] > (1, 2)
--------------------------------
[1, 2] > 'foo' =  False
(1, 2) > 'foo' =  True
```

```
[1, 2] > (1, 2) =  False
```

Python 3.x

```
print("[1, 2] > 'foo' = ", [1, 2] > 'foo')
print("(1, 2) > 'foo' = ", (1, 2) > 'foo')
print("[1, 2] > (1, 2) = ", [1, 2] > (1, 2))
--------------------------------
TypeError                      Traceback (most recent call last)

<ipython-input-16-a9031729f4a0> in <module>()
      1 print('Python', python_version())
----> 2 print("[1, 2] > 'foo' = ", [1, 2] > 'foo')
      3 print("(1, 2) > 'foo' = ", (1, 2) > 'foo')
      4 print("[1, 2] > (1, 2) = ", [1, 2] > (1, 2))
      TypeError: unorderable types: list() > str()
```

❖ Phân tích cú pháp đầu vào của người dùng thông qua input()

Hàm input () được cố định trong Python 3 để nó luôn lưu trữ đầu vào của người dùng làm các đối tượng str. Để tránh hành vi nguy hiểm trong Python 2 để đọc trong các loại khác so với chuỗi, chúng ta phải sử dụng raw_input () để thay thế.

Vd: Python 2.x

```
Python 2.7.6
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.

>>> my_input = input('enter a number: ')

enter a number: 123

>>> type(my_input)
<type 'int'>

>>> my_input = raw_input('enter a number: ')

enter a number: 123

>>> type(my_input)
<type 'str'>
```

Python 3.x

```
Python 3.4.1
[GCC 4.2.1 (Apple Inc. build 5577)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.

>>> my_input = input('enter a number: ')

enter a number: 123

>>> type(my_input)
<class 'str'>
```

Trả về các đối tượng có thể lặp lại thay vì danh sách

- **Một số hàm và phương thức trả về các đối tượng có thể lặp lại trong Python 3 ngay-thay vì các danh sách trong Python 2**

  Vd: Python 2.x

```
print range(3)
print type(range(3))
--------------------------------
[0, 1, 2]
<type 'list'>
```

Python 3

```
print(range(3))
print(type(range(3)))
print(list(range(3)))
--------------------------------
range(0, 3)
<class 'range'>
[0, 1, 2]
```

Một số hàm và phương thức thường được sử dụng không trả về danh sách nữa trong Python 3:

```
zip()
map()
filter()
dictionary's .keys() method
dictionary's .values() method
dictionary's .items() method
```

4. Đặc điểm của Python

. Dễ học, dễ đọc

Python được thiết kế để trở thành một ngôn ngữ dễ học, mã nguồn dễ đọc, bố cục trực quan, dễ hiểu, thể hiện qua các điểm sau:

Từ khóa

- Python tăng cường sử dụng từ khóa tiếng Anh, hạn chế các kí hiệu và cấu trúc cú pháp so với các ngôn ngữ khác.
- Python là một ngôn ngữ phân biệt kiểu chữ HOA, chữ thường.
- Như C/C++, các từ khóa của Python đều ở dạng chữ thường.

Khối lệnh

Trong các ngôn ngữ khác, khối lệnh thường được đánh dấu bằng cặp kí hiệu hoặc từ khóa. Ví dụ, trong C/C++, cặp ngoặc nhọn { } được dùng để bao bọc một khối lệnh. Python, trái lại, có một cách rất đặc biệt để tạo khối lệnh, đó là thụt các câu lệnh trong khối vào sâu hơn (về bên phải) so với các câu lệnh của khối lệnh cha chứa nó. Ví dụ, giả sử có đoạn mã sau trong C/C++:

```
#include <math.h>
    //...
    delta = b * b - 4 * a * c;
    if (delta > 0) {
    // Khoi lenh moi bat dau tu kí tu { den }
    x1 = (- b + sqrt(delta)) / (2 * a);
    x2 = (- b - sqrt(delta))/ (2 * a);
    printf("Phuong trinh co hai nghiem phan biet:\n");
    printf("x1 = %f; x2 = %f", x1, x2);
    }
```

Đoạn mã trên có thể được viết lại bằng Python như sau:

```
# -*- coding: utf-8 -*-
"""
Spyder Editor
This is a temporary script file.
"""
import math
  #...
  delta = b * b – 4 * a * c
  if delta > 0:
```

```
        # Khối lệnh mới, thụt vào đầu dòng
        x1 = (- b + math.sqrt(delta)) / (2 * a)
        x2 = (- b – math.sqrt(delta)) / (2 * a)
        print "Phuong trinh co hai nghiem phan biet:"
    print "x1 = ", x1, "; ", "x2 = ", x2
```

Ta có thể sử dụng dấu tab hoặc khoảng trống để thụt các câu lệnh vào.

Các bản hiện thực

Python được viết từ những ngôn ngữ khác, tạo ra những bản hiện thực khác nhau. Bản hiện thực Python chính, còn gọi là CPython, được viết bằng C, và được phân phối kèm một thư viện chuẩn lớn được viết hỗn hợp bằng C và Python. CPython có thể chạy trên nhiều nền và khả chuyển trên nhiều nền khác. Dưới đây là các nền trên đó, CPython có thể chạy.

- Các hệ điều hành họ Unix: AIX, Darwin, FreeBSD, Mac OS X, NetBSD, Linux, OpenBSD, Solaris,…
- Các hệ điều hành dành cho máy desktop: Amiga, AROS, BeOS, Mac OS 9, Microsoft Windows, OS/2, RISC OS.
- Các hệ thống nhúng và các hệ đặc biệt: GP2X, Máy ảo Java, Nokia 770 Internet Tablet, Palm OS, PlayStation 2, PlayStation Portable, Psion, QNX, Sharp Zaurus, Symbian OS, Windows CE/Pocket PC, Xbox/XBMC, VxWorks.
- Các hệ máy tính lớn và các hệ khác: AS/400, OS/390, Plan 9 from Bell Labs, VMS, z/OS.

Ngoài CPython, còn có hai hiện thực Python khác: Jython cho môi trường Java và IronPython cho môi trường .NET và Mono.

Khả năng mở rộng

Python có thể được mở rộng: nếu ta biết sử dụng C, ta có thể dễ dàng viết và tích hợp vào Python nhiều hàm tùy theo nhu cầu. Các hàm này sẽ trở thành hàm xây dựng sẵn (built-in) của Python. Ta cũng có thể mở rộng chức năng của trình thông dịch, hoặc liên kết các chương trình Python với các thư viện chỉ ở dạng nhị phân (như các thư viện đồ họa do nhà sản xuất thiết bị cung cấp). Hơn thế nữa, ta cũng có thể liên kết trình thông dịch của Python với các ứng dụng viết từ C và sử dụng nó như là một mở rộng hoặc một ngôn ngữ dòng lệnh phụ trợ cho ứng dụng đó.

Trình thông dịch

Python là một ngôn ngữ lập trình dạng thông dịch, do đó có ưu điểm tiết kiệm thời gian phát triển ứng dụng vì không cần phải thực hiện biên dịch và liên kết. Trình thông dịch có thể được sử dụng để chạy file script, hoặc

cũng có thể được sử dụng theo cách tương tác. Ở chế độ tương tác, trình thông dịch Python tương tự shell của các hệ điều hành họ Unix, tại đó, ta có thể nhập vào từng biểu thức rồi gõ Enter, và kết quả thực thi sẽ được hiển thị ngay lập tức. Đặc điểm này rất hữu ích cho người mới học, giúp họ nghiên cứu tính năng của ngôn ngữ; hoặc để các lập trình viên chạy thử mã lệnh trong suốt quá trình phát triển phần mềm. Ngoài ra, cũng có thể tận dụng đặc điểm này để thực hiện các phép tính như với máy tính bỏ túi.

Lệnh và cấu trúc điều khiển

Mỗi câu lệnh trong Python nằm trên một dòng mã nguồn. Ta không cần phải kết thúc câu lệnh bằng bất kì kí tự gì. Cũng như các ngôn ngữ khác, Python cũng có các cấu trúc điều khiển. Chúng bao gồm:

Cấu trúc rẽ nhánh: cấu trúc if (có thể sử dụng thêm elif hoặc else), dùng để thực thi có điều kiện một khối mã cụ thể.

Cấu trúc lặp, bao gồm:

- Lệnh while: chạy một khối mã cụ thể cho đến khi điều kiện lặp có giá trị false.
- Vòng lặp for: lặp qua từng phần tử của một dãy, mỗi phần tử sẽ được đưa vào biến cục bộ để sử dụng với khối mã trong vòng lặp.

Python cũng có từ khóa class dùng để khai báo lớp (sử dụng trong lập trình hướng đối tượng) và lệnh def dùng để định nghĩa hàm.

Hệ thống kiểu dữ liệu

Python sử dụng hệ thống kiểu duck typing, còn gọi là latent typing (tự động xác định kiểu). Có nghĩa là, Python không kiểm tra các ràng buộc về kiểu dữ liệu tại thời điểm dịch, mà là tại thời điểm thực thi. Khi thực thi, nếu một thao tác trên một đối tượng bị thất bại, thì có nghĩa là đối tượng đó không sử dụng một kiểu thích hợp.

Python cũng là một ngôn ngữ định kiểu mạnh. Nó cấm mọi thao tác không hợp lệ, ví dụ cộng một con số vào chuỗi kí tự.

Sử dụng Python, ta không cần phải khai báo biến. Biến được xem là đã khai báo nếu nó được gán một giá trị lần đầu tiên. Căn cứ vào mỗi lần gán, Python sẽ tự động xác định kiểu dữ liệu của biến. Python có một số kiểu dữ liệu thông dụng sau:

- int, long: số nguyên (trong phiên bản 3.x long được nhập vào trong kiểu int). Độ dài của kiểu số nguyên là tùy ý, chỉ bị giới hạn bởi bộ nhớ máy tính.
- float: số thực
- complex: số phức, chẳng hạn 5+4j

- list: dãy trong đó các phần tử của nó có thể được thay đổi, chẳng hạn [8, 2, 'b', -1.5] . Kiểu dãy khác với kiểu mảng (array) thường gặp trong các ngôn ngữ lập trình ở chỗ các phần tử của dãy không nhất thiết có kiểu giống nhau. Ngoài ra phần tử của dãy còn có thể là một dãy khác.
- tuple: dãy trong đó các phần tử của nó không thể thay đổi.
- str: chuỗi kí tự. Từng kí tự trong chuỗi không thể thay đổi. Chuỗi kí tự được đặt trong dấu nháy đơn, hoặc nháy kép.
- dict: từ điển, còn gọi là "hashtable": là một cặp các dữ liệu được gắn theo kiểu {từ khóa: giá trị}, trong đó các từ khóa trong một từ điển nhất thiết phải khác nhau. Chẳng hạn {1: "Python", 2: "Pascal"}
- set: một tập không xếp theo thứ tự, ở đó, mỗi phần tử chỉ xuất hiện một lần.

Ngoài ra, Python còn có nhiều kiểu dữ liệu khác. Xem thêm trong phần "Các kiểu dữ liệu" bên dưới.

Module

Python cho phép chia chương trình thành các module để có thể sử dụng lại trong các chương trình khác. Nó cũng cung cấp sẵn một tập hợp các modules chuẩn mà lập trình viên có thể sử dụng lại trong chương trình của họ. Các module này cung cấp nhiều chức năng hữu ích, như các hàm truy xuất tập tin, các lời gọi hệ thống, trợ giúp lập trình mạng (socket),…

Đa năng

Python là một ngôn ngữ lập trình đơn giản nhưng rất hiệu quả.

- So với Unix shell, Python hỗ trợ các chương trình lớn hơn và cung cấp nhiều cấu trúc hơn.
- So với C, Python cung cấp nhiều cơ chế kiểm tra lỗi hơn. Nó cũng có sẵn nhiều kiểu dữ liệu cấp cao, ví dụ như các mảng (array) linh hoạt và từ điển (dictionary) mà ta sẽ phải mất nhiều thời gian nếu viết bằng C.

Python là một ngôn ngữ lập trình cấp cao có thể đáp ứng phần lớn yêu cầu của lập trình viên:

- Python thích hợp với các chương trình lớn hơn cả AWK và Perl.
- Python được sử dụng để lập trình Web. Nó có thể được sử dụng như một ngôn ngữ kịch bản.
- Python được thiết kế để có thể nhúng và phục vụ như một ngôn ngữ kịch bản để tuỳ biến và mở rộng các ứng dụng lớn hơn.
- Python được tích hợp sẵn nhiều công cụ và có một thư viện chuẩn phong phú, Python cho phép người dùng dễ dàng tạo ra các dịch vụ Web, sử dụng các thành phần COM hay CORBA, hỗ trợ các loại

định dạng dữ liệu Internet như email, HTML, XML và các ngôn ngữ đánh dấu khác. Python cũng được cung cấp các thư viện xử lý các giao thức Internet thông dụng như HTTP, FTP,…

- Python có khả năng giao tiếp đến hầu hết các loại cơ sở dữ liệu, có khả năng xử lí văn bản, tài liệu hiệu quả, và có thể làm việc tốt với các công nghệ Web khác.
- Python đặc biệt hiệu quả trong lập trình tính toán khoa học nhờ các công cụ Python Imaging Library, pyVTK, MayaVi 3D Visualization Toolkits, Numeric Python, ScientificPython,…
- Python có thể được sử dụng để phát triển các ứng dụng desktop. Lập trình viên có thể dùng wxPython, PyQt, PyGtk để phát triển các ứng dụng giao diện đồ họa (GUI) chất lượng cao. Python còn hỗ trợ các nền tảng phát triển phần mềm khác như MFC, Carbon, Delphi, X11, Motif, Tk, Fox, FLTK, …
- Python cũng có sẵn một unit testing framework để tạo ra các các bộ test (test suites).

Multiple paradigms (đa biến hóa)

Python là một ngôn ngữ đa biến hóa (multiple paradigms). Có nghĩa là, thay vì ép buộc mọi người phải sử dụng duy nhất một phương pháp lập trình, Python lại cho phép sử dụng nhiều phương pháp lập trình khác nhau: hướng đối tượng, có cấu trúc, chức năng, hoặc chỉ hướng đến một khía cạnh. Python kiểu kiểu động và sử dụng bộ thu gom rác để quản lí bộ nhớ. Một đặc điểm quan trọng nữa của Python là giải pháp tên động, kết nối tên biến và tên phương thức lại với nhau trong suốt thực thi của chương trình.

Sự tương đương giữa true và một giá trị khác 0

Cũng như C/C++, bất kì một giá trị khác 0 nào cũng tương đương với true và ngược lại, một giá trị 0 tương đương với false. Như vậy:

```
if a != 0:
```

tương đương với:

```
if a:
```

Cú pháp

Sau đây là cú pháp cơ bản nhất của ngôn ngữ Python:

➢ ***Toán tử***

```
+ - * / // (chia làm tròn) % (phần dư) ** (lũy thừa)
    ~ (not) & (and) | (or) ^ (xor)
```

```
    << (left shift) >> (right shift)
    == (bằng) <= >= != (khác)
```

Python sử dụng kí pháp trung tố thường gặp trong các ngôn ngữ lập trình khác.

- ***Các kiểu dữ liệu***
  - *Kiểu số*

```
1234585396326 (số nguyên dài vô hạn) -86.12 7.84E-04
    2j 3 + 8j (số phức)
```

  - *Kiểu chuỗi (string)*

```
"Hello" "It's me" '"OK"-he replied'
```

  - *Kiểu bộ (tuple)*

```
(1, 2.0, 3) (1,) ("Hello",1,())
```

  - *Kiểu danh sách (list)*

```
[4.8, -6] ['a','b']
```

  - *Kiểu từ điển (dictionary)*

```
{"Vietnam":"Hanoi", "Netherlands":"Amsterdam", "France":"Paris"}
```

- ***Chú thích***

```
   # dòng chú thích
```

- ***Lệnh gán***

```
tên biến = biểu thức
    x = 23.8
    y = -x ** 2
    z1 = z2 = x + y
    loiChao = "Hello!"

i += 1 # tăng biến i thêm 1 đơn vị
```

- ***In giá trị***

```
print biểu thức
    print (7 + 8) / 2.0
    print (2 + 3j) * (4 - 6j)
```

- ***Nội suy chuỗi (string interpolation)***

```
print "Hello %s" %("world!")
    print "i = %d" %i
    print "a = %.2f and b = %.3f" %(a,b)
```

- ***Cấu trúc rẽ nhánh***

- *Dạng 1:*

```
if biểu_thức_điều_kiện:
    # lệnh ...
```

- *Dạng 2:*

```
if biểu_thức_điều_kiện:
    # lệnh ...
    else:
    # lệnh ...
```

- *Dạng 3:*

```
if biểu_thức_điều_kiện_1:
    # lệnh ... (được thực hiện nếu biểu_thức_điều_kiện_1 là đúng/true)
    elif biểu_thức_điều_kiện_2:
    # lệnh ... (được thực hiện nếu biểu_thức_điều_kiện_1 là sai/false,
nhưng biểu_thức_điều_kiện_2 là đúng/true)
    else:
    # lệnh ... (được thực hiện nếu tất cả các biểu thức điều kiện đi kèm
if và elif đều sai)
```

- ***Cấu trúc lặp***

```
while biểu_thức_đúng:
    # lệnh ...
    for phần_tử in dãy:
    # lệnh ...
    L = ["Ha Noi", "Hai Phong", "TP Ho Chi Minh"]
    for thanhPho in L:
    print thanhPho

for i in range(10):
    print i
```

- ***Hàm***

```
def tên_hàm (tham_biến_1, tham_biến_2, tham_biến_n):
    # lệnh ...
    return giá_trị_hàm
    def binhPhuong(x):
    return x*x
```

- ***Hàm với tham số mặc định:***

```
def luyThua(x, n=2):
    """Lũy thừa với số mũ mặc định là 2"""
    return x**n

print luyThua(3) # 9
    print luyThua(2,3) # 8
```

- ***Lớp***

```
class Tên_Lớp_1:
    # ...

class Tên_Lớp_2(Tên_Lớp_1):
    """Lớp 2 kế thừa lớp 1"""
    x = 3 # biến thành viên của lớp
    #
    def phương_thức(self, tham_biến):
    # ...

# khởi tạo
    a = Tên_Lớp_2()
    print a.x
    print a.phương_thức(m) # m là giá trị gán cho tham biến
    List Comprehension
```

- ***List Comprehension***

Là dạng cú pháp đặc biệt (syntactic sugar) (mới có từ Python 2.x) cho phép thao tác trên toàn bộ dãy (list) mà không cần viết rõ vòng lặp. Chẳng hạn y là một dãy mà mỗi phần tử của nó bằng bình phương của từng phần tử trong dãy x:

```
y = [xi**2 for xi in x]
```

- ***Xử lý lỗi***

```
try:
    câu_lệnh
    except Loại_Lỗi:
    thông báo lỗi
```

- ***Tốc độ thực hiện***

Là một ngôn ngữ thông dịch, Python có tốc độ thực hiện chậm hơn nhiều lần so với các ngôn ngữ biên dịch như Fortran, C, v.v... Trong số các ngôn ngữ thông dịch, Python được đánh giá nhanh hơn Ruby và Tcl, nhưng chậm hơn Lua.

➢ ***Các đặc điểm mới trong Python 3.x***

Nội dung phần này được trích từ tài liệu của Guido van Rossum. Phần này không liệt kê đầy đủ tất cả các đặc điểm; chi tiết xin xem tài liệu nói trên.

➢ ***Một số thay đổi cần lưu ý nhất***

Lệnh print trở thành hàm print. Theo đó sau print() ta cần nhớ gõ vào cặp ngoặc ():

```
print("Hello")
    print(2+3)
```

Trả lại kết quả không còn là list trong một số trường hợp.

- dict.keys(), dict.items(), dict.values() kết quả cho ra các "view" thay vì list.
- map và filter trả lại các iterator.
- range bây giờ có tác dụng như xrange, và không trả lại list.

✍ *So sánh*

Không còn hàm cmp, và cmp(a, b) có thể được thay bằng (a > b) - (a < b)

✍ Số nguyên

- Kiểu long được đổi tên thành int.
- 1/2 cho ta kết quả là số thực chứ không phải số nguyên.
- Không còn hằng số sys.maxint
- Kiểu bát phân được kí hiệu bằng 0o thay vì 0, chẳng hạn 0o26.

Phân biệt văn bản - dữ liệu nhị phân thay vì Unicode - chuỗi 8-bit

- Tất cả chuỗi văn bản đều dưới dạng Unicode, nhưng chuỗi Unicode mã hóa lại là dạng dữ liệu nhị phân. Dạng mặc định là UTF-8.
- Không thể viết u"a string" để biểu diễn chuỗi như trong các phiên bản 2.x

➢ ***Các thay đổi về cú pháp***

✍ *Cú pháp mới*

- Các tham biến chỉ chấp nhận keyword: Các tham biến phía sau *args phải được gọi theo dạng keyword.
- Từ khóa mới nonlocal. Muốn khai báo một biến x với có phạm vi ảnh hưởng rộng hơn, nhưng chưa đến mức toàn cục, ta dùng nonlocal x.

- Gán giá trị vào các phần tử tuple một cách thông minh, chẳng hạn có thể viết (a, *rest, b) = range(5) để có được a = 0; b = [1,2,3]; c = 4.
- Dictionary comprehension, chẳng hạn {k: v for k, v in stuff} thay vì dict(stuff).
- Kiểu nhị phân, chẳng hạn b110001.

✍ *Cú pháp được thay đổi*

- raise [biểu_thức [from biểu_thức]]
- except lệnh as biến
- Sử dụng metaclass trong đối tượng:

```
class C(metaclass=M):
    pass
```

Cách dùng biến __metaclass__ không còn được hỗ trợ.

➢ ***Cú pháp bị loại bỏ***

- Không còn dấu ``, thay vì đó, dùng repr.
- Không còn so sánh <> (dùng !=).

Không còn các lớp kiểu classic.

CHƯƠNG II. HƯỚNG DẪN CÀI ĐẶT

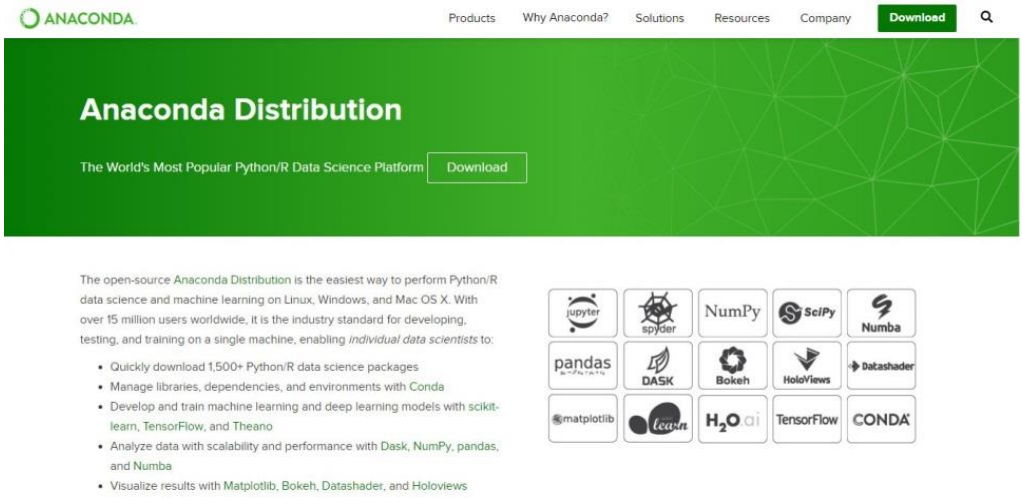
1. Hướng dẫn cài đặt Anaconda

1.1. Giới thiệu Anaconda

Anaconda là nền tảng mã nguồn mở về Khoa học dữ liệu trên Python thông dụng nhất hiện nay. Anaconda hướng đến việc quản lí các package một cách đơn giản, phù hợp với mọi người. Hệ thống quản lí package của Anaconda là Conda. Anaconda Với hơn 11 triệu người dùng, Anaconda là cách nhanh nhất và dễ nhất để học Khoa học dữ liệu với Python hoặc R trên Windows, Linux và Mac OS X. Lợi ích của Anaconda:

- Dễ dàng tải 1500+ packages về Python/R cho data science
- Quản lý thư viện, môi trường và dependency giữa các thư viện dễ dàng
- Dễ dàng phát triển mô hình machine learning và deep learning với scikit-learn, tensorflow, keras
- Xử lý dữ liệu tốc độ cao với numpy, pandas
- Hiện thị kết quả với Matplotlib, Bokeh

Trong khi đó Spyder là 1 trong những IDE (môi trường tích hợp dùng để phát triển phần mềm) tốt nhất cho data science và quang trọng hơn là nó được cài đặt khi bạn cài đặt Anaconda.



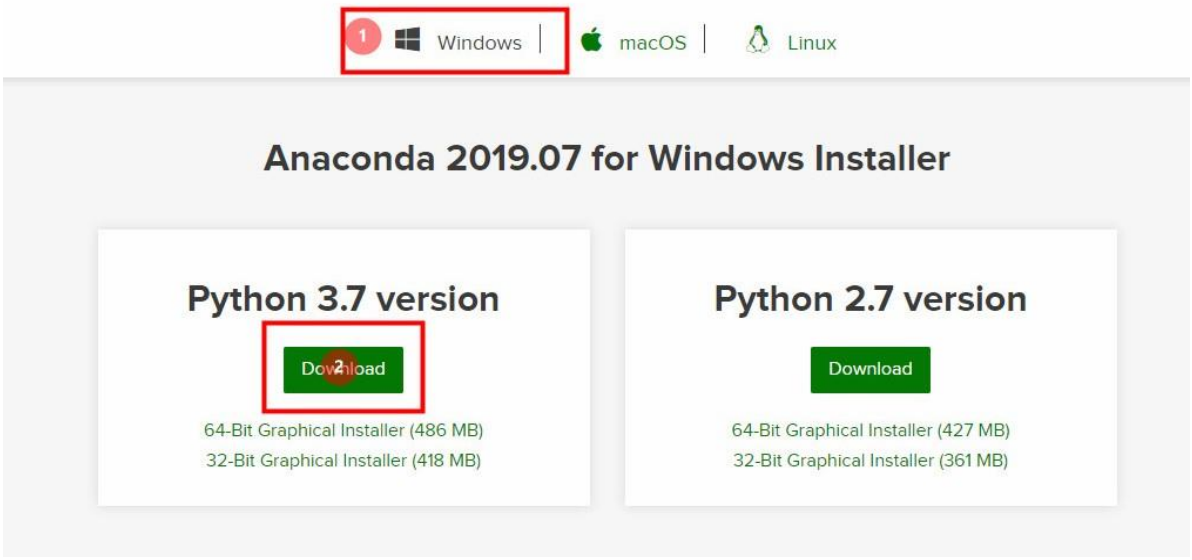
Hình 2.1. Giao diện trang chủ Anaconda

Yêu cầu phần cứng và phần mềm:

- Hệ điều hành: Win 7, Win 8/8.1, Win 10, Red Hat Enterprise Linux/CentOS 6.7, 7.3, 7.4, and 7.5, and Ubuntu 12.04+.
- Ram tối thiểu 4GB
- Ổ cứng trống tối thiểu 3GB để tải và cài đặt

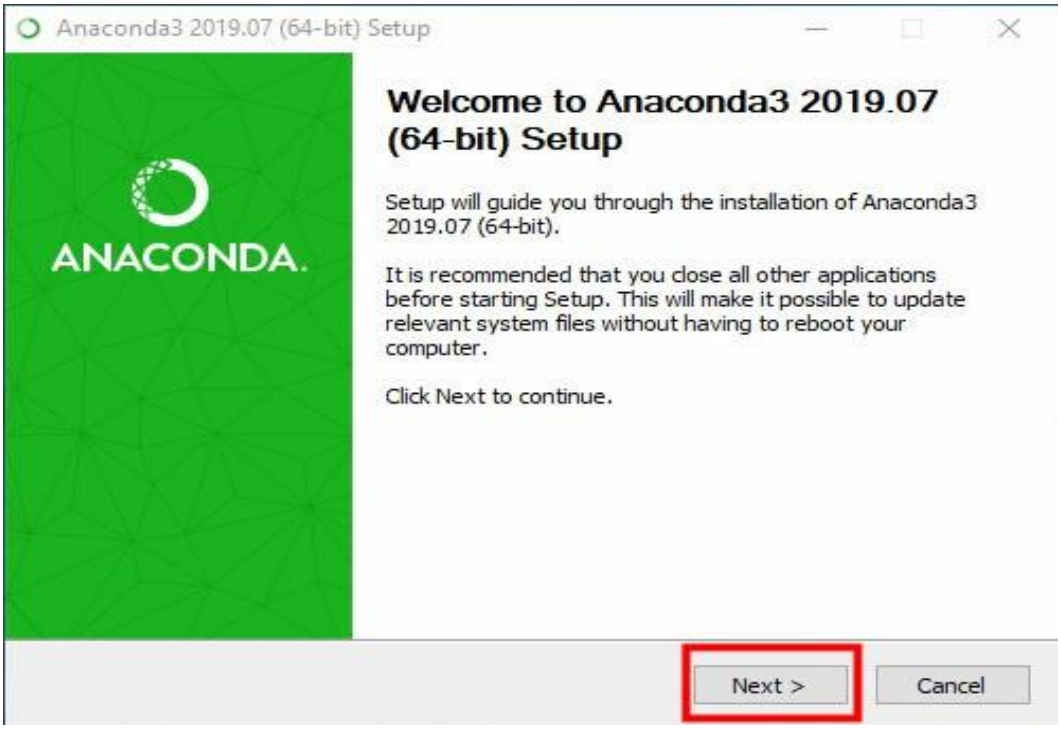
1.2. *Download Anaconda và hướng dẫn cài đặt trên HĐH Windown*

- Download Anaconda (python3) cho HĐH Windown về tại https://www.anaconda.com/distribution/
- Hiện tại phiên bản Python mà ta sử dụng là python3.7 bản phân phối Anaconda 2019.07

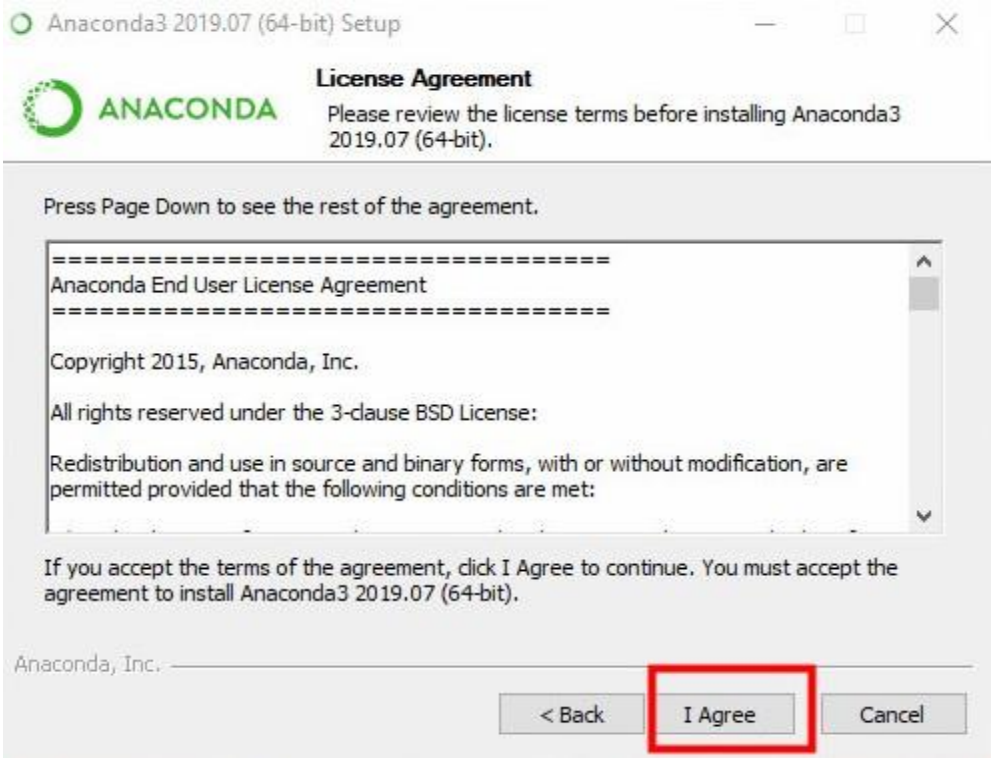


Hình 2.2: Trang tải Anaconda

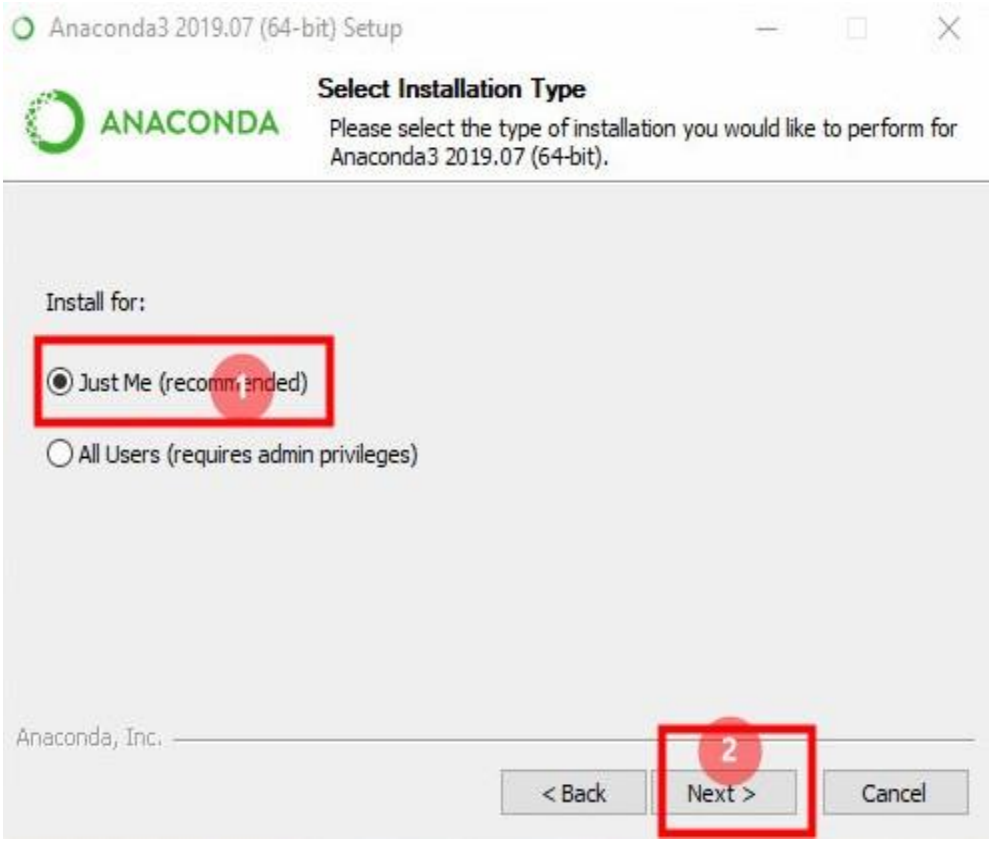
Sau khi tải về xong bạn chạy file "Anaconda3-2019.07-Windows-x86_64.exe"



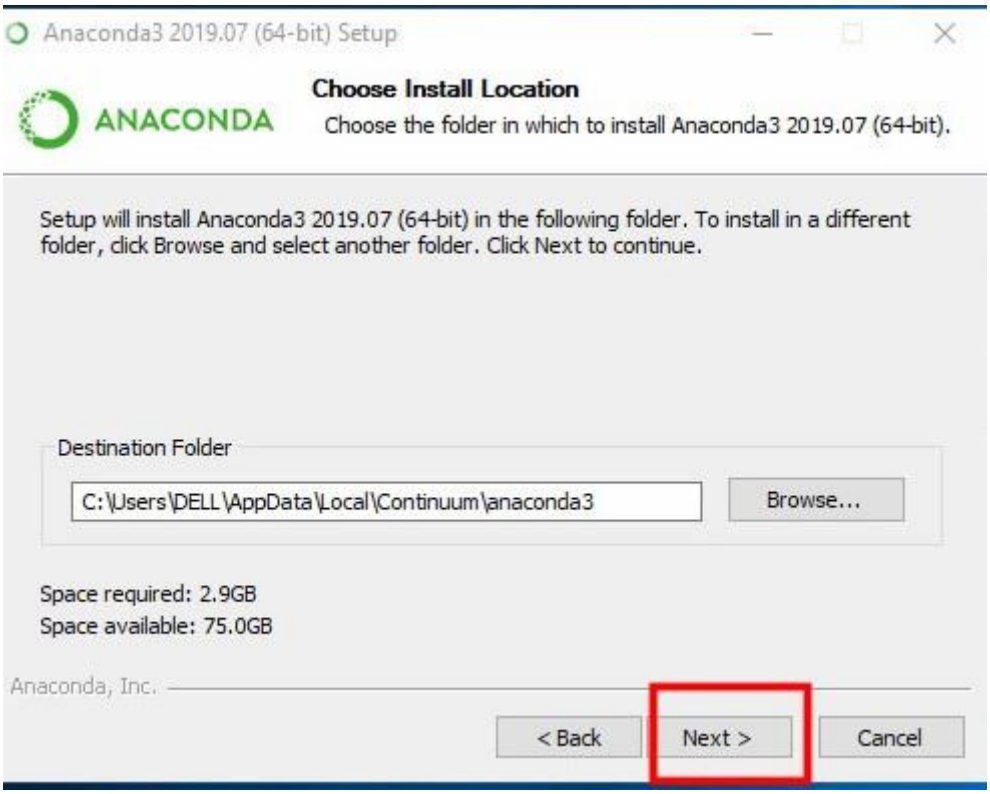
- *Click* **Next**



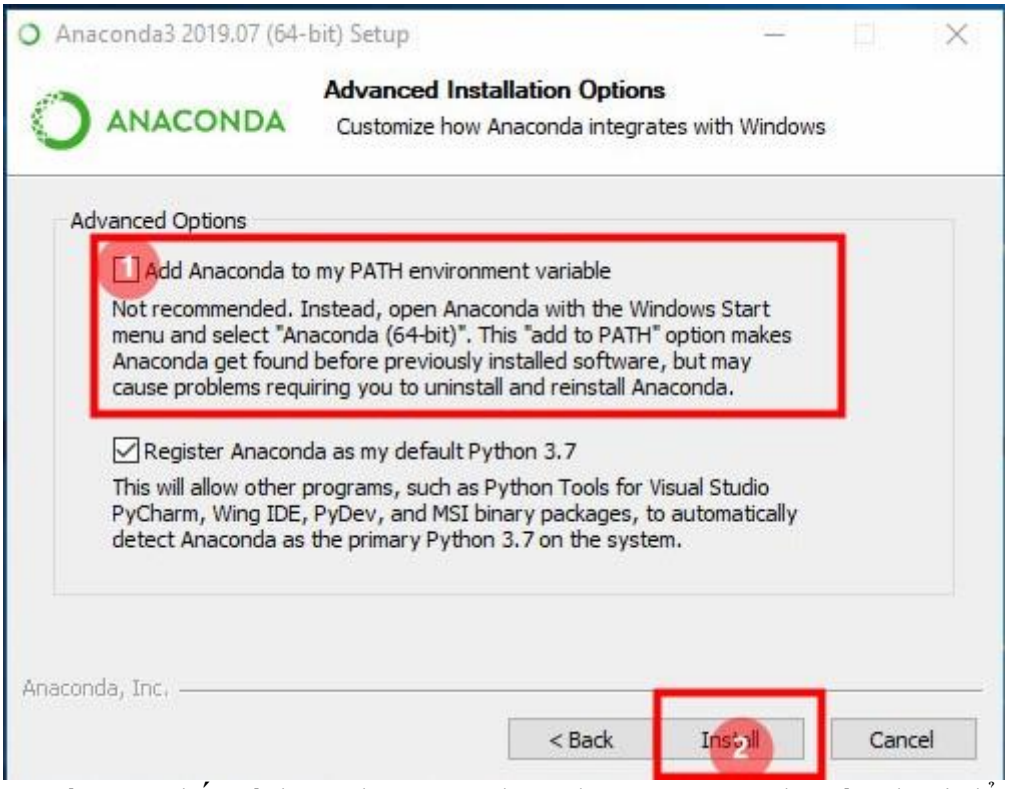
- *Click* **I Agree**



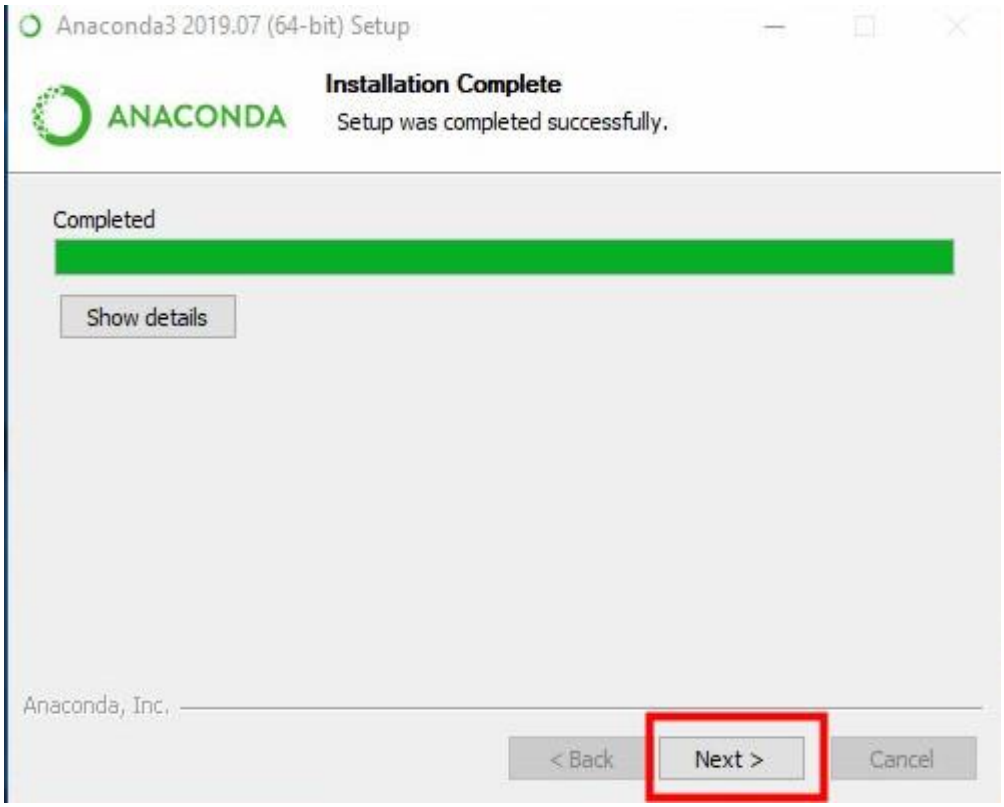
- Bạn chọn "Just Me" sau đó *Click* **Next**



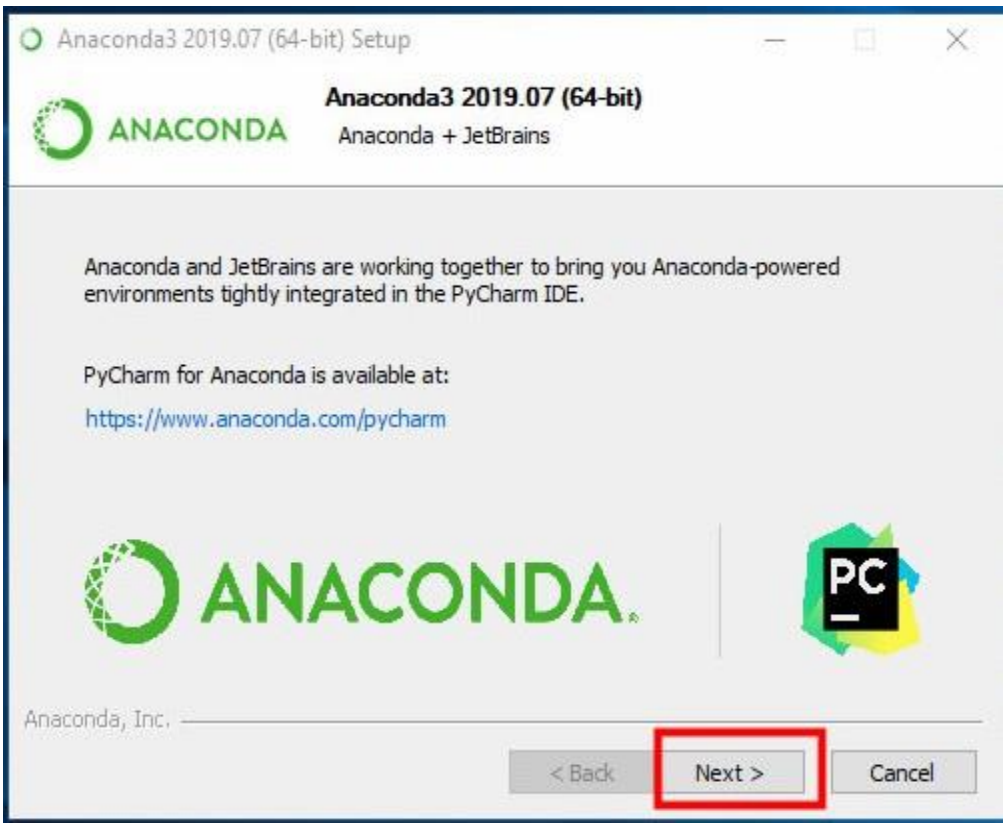
- *Click* **Next**



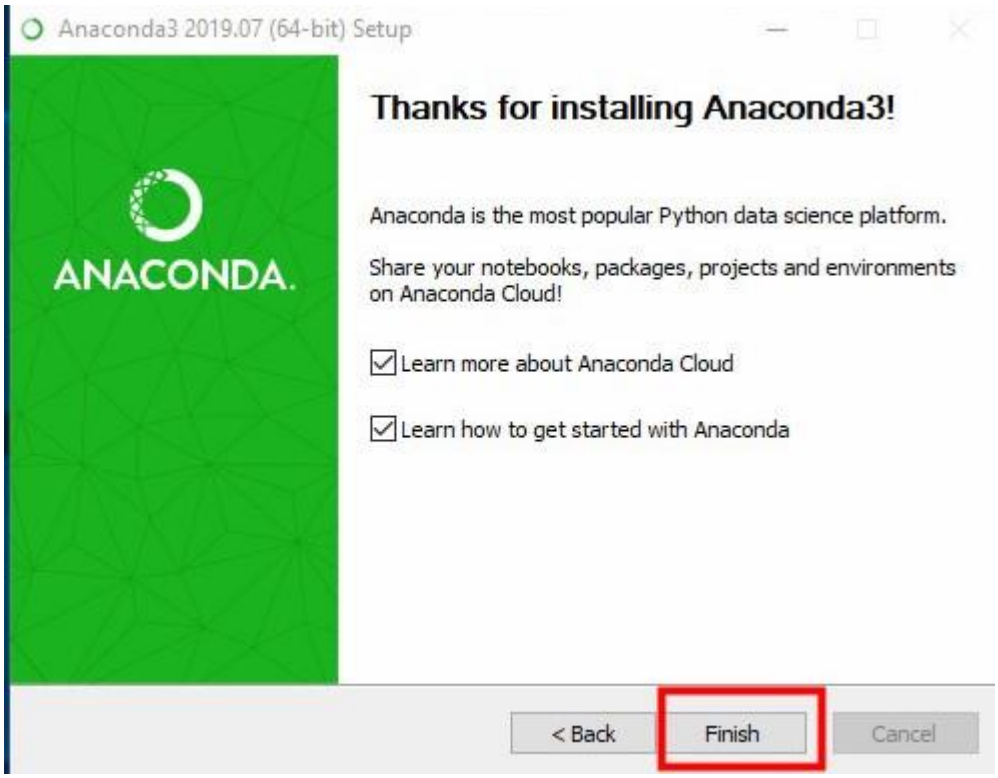
- Lưu ý: hãy chắc là bạn chọn tuỳ chọn được khoanh đỏ bên dưới để thêm các câu lệnh Anaconda vào **System Environment Variable (PATH)** của Windows
- Sau đó *Click* **Next**



- *Tiếp tục Click* **Next**



- *Click* **Next**

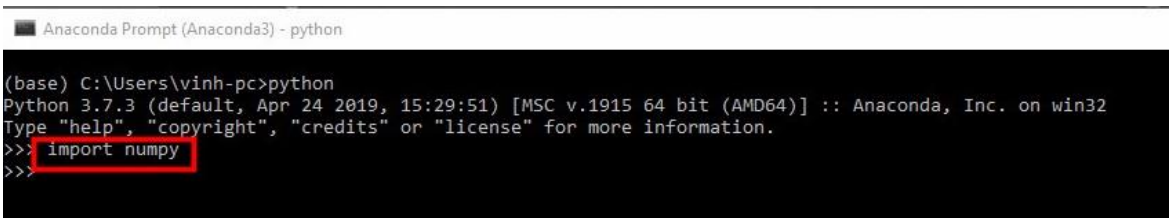


- Tiếp theo bạn click **Finish** để hoàn tất việc cài đặt

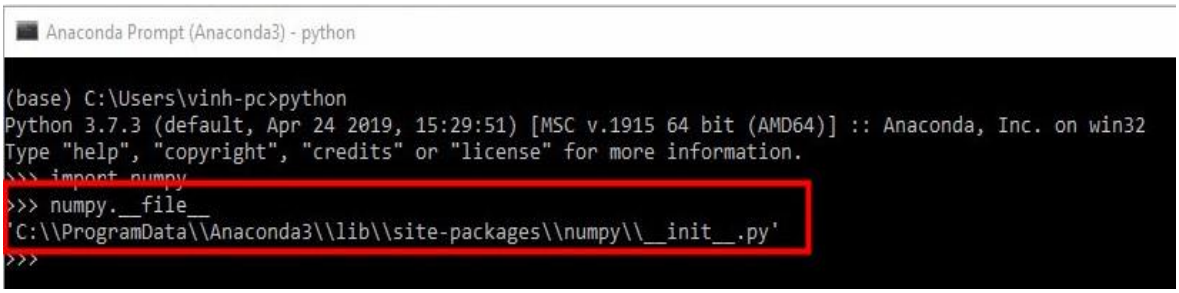
1.3. Hướng dẫn cài đặt thêm thư viện

Anaconda đã có sẵn khá là nhiều thư viện python như : Numpy, Scipy, Matplotlib, sklearn

Để kiểm tra python của Anaconda đã có thư viện nào đó, chúng ta sẽ thử import nó trong Console

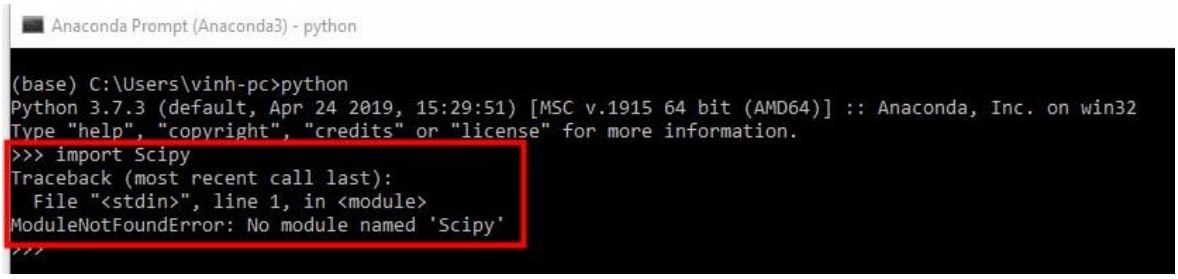


Không có lỗi được thông báo nghĩa là python đã biết được thư viện này. Để kiểm tra thư viện này ở đâu, sau khi *import*, ta truy xuất đường dẫn của thư viện như sau:



Thư viện Numpy của tôi nằm ở đường dẫn *'C:\ProgramData\Anaconda3\lib\site-packages\'*.

Anaconda đã có sẵn thư viện Numpy

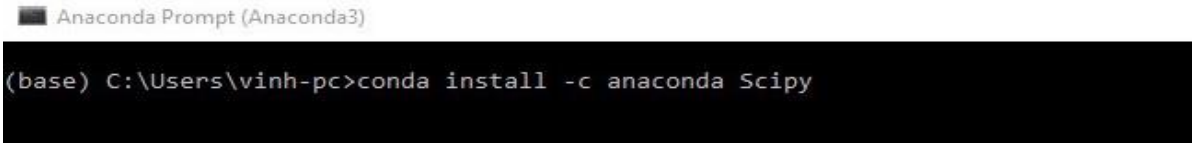


Nếu như Python trả về lỗi Import như trên thì có nghĩa trong Anaconda chúng ta chưa có thư viện đó.

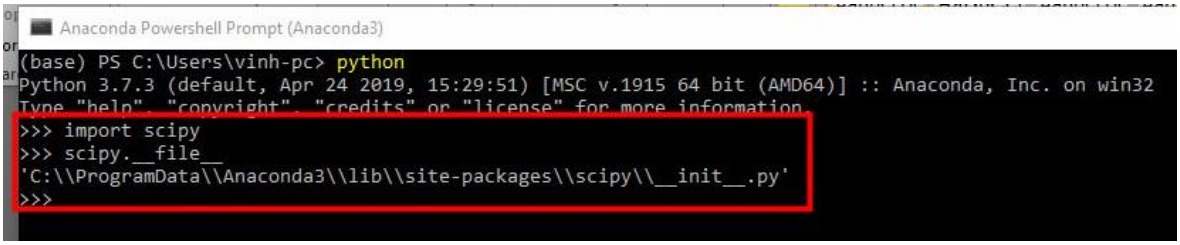
Ở phần trên python của tôi chưa có thư viện *Scipy*, nên tôi phải đi cài đặt nó. Vì tôi sử dụng Anaconda cho lập trình python nên tôi cần phải *(1) cài đặt thư viện mới vào đường dẫn libs python của Anaconda* hoặc *(2) chỉ cho python của Anaconda biết về đường dẫn tới thư viện mới này*.

Với Anaconda, việc cài đặt 1 thư viện đang được hỗ trợ cực kỳ đơn giản, tôi chỉ cần dùng tools *pip* hoặc *conda* mà Anaconda đã cài sẵn. Cụ thể, ở đây tôi muốn cài thư viện *Scipy* tôi truy cập vào trang chủ của Scipy. Trang này ghi rằng chúng ta có thể cài bằng *pip* hoặc *conda*.

Chúng ta sẽ bật Anaconda Prompt (Anaconda3) lên và gõ conda install -c anaconda Scipy. Conda sẽ tự động tìm thư viện *Scipy* và cài vào đường dẫn Anaconda giúp chúng ta.



Chờ cho thư viện và các thư viện liên quan hoàn tất cài đặt, chúng ta vào spyder kiểm tra lại đã có *Scipy* chưa. Và python trả về đã có Scipy trong Anaconda. Và chúng ta đã có thể sử dụng Scipy



Với 1 thư viện chưa có trên Anaconda, cách cài đặt sẽ phức tạp hơn chút nhưng hầu hết các thư viện lớn thường dùng đều có thể cài đặt thông qua Anaconda, nên chúng ta không phải lo lắng lắm.

Ngoài ra chúng ta có thể cài đặt thêm các thư viện bằng Anaconda Navigator

2. Hướng dẫn cài đặt IDE Spyder

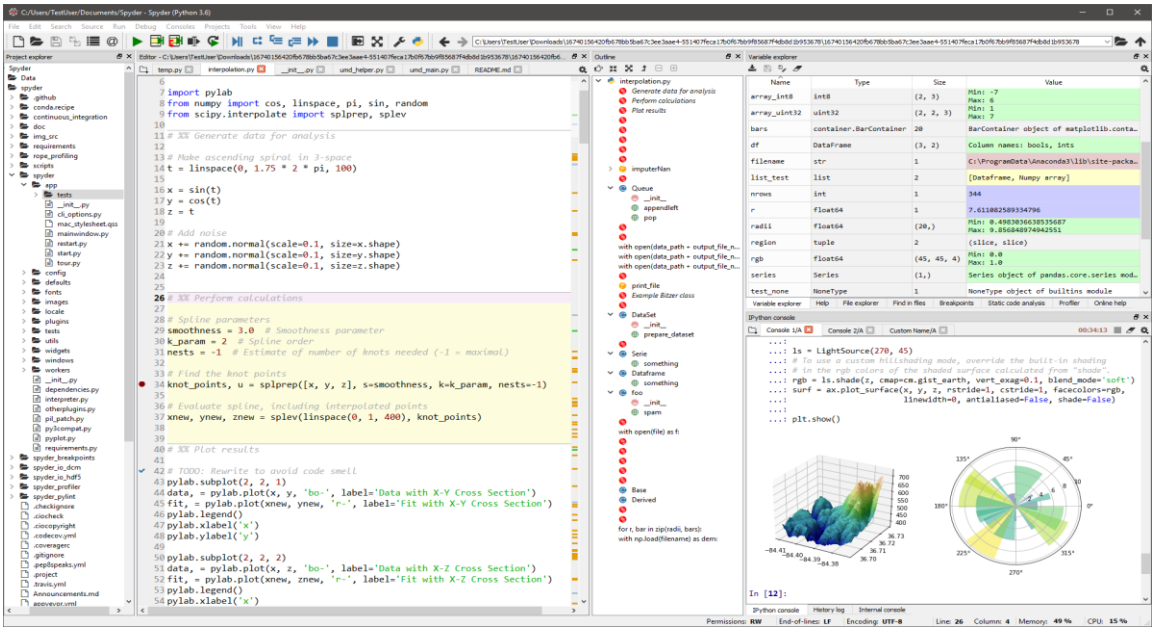
2.1. Giới thiệu IDE Spyder

Spyder là một môi trường phát triển Python mã nguồn mở được tối ưu hóa cho các bài toán liên quan đến khoa học dữ liệu. Spyder đi kèm với phân phối quản lý gói Anaconda. Spyder là công cụ thường dùng của các nhà khoa học dữ liệu sử dụng Python. Spyder tích hợp tốt với các thư viện khoa học dữ liệu Python phổ biến như SciPy, NumPy và Matplotlib.

Spyder có hầu hết các tính năng của một "IDE phổ biến", chẳng hạn như trình soạn thảo mã với chức năng đánh dấu cú pháp mạnh mẽ, tự động hoàn thành mã và thậm chí là trình duyệt tài liệu được tích hợp.

Một tính năng đặc biệt không có trong các môi trường phát triển Python khác là tính năng "khám phá biến" của Spyder cho phép hiển thị dữ liệu bằng cách sử dụng bố cục bảng ngay bên trong IDE. Điều này làm nó nó trông khá gọn gàng. Nếu bạn thường xuyên làm các bài toán khoa học dữ liệu làm việc bằng cách sử dụng Python, thì đây là một tính năng độc đáo. Việc tích hợp IPython/Jupyter là một đặc điểm nổi bật khác.

Nhìn chung Spyder có nhiều chức năng nổi trội cơ bản hơn các IDE khác. Điều đặc biệt khác là Spyder miễn phí trên Windows, macOS, và Linux và nó là phần mềm mã nguồn mở hoàn toàn.



Hình 2.3. Giao diện chính của IDE Spyder

✓ **Ưu điểm:** Tối ưu nhiều tính chất phù hợp cho các hoạt động khoa học dữ liệu sử dụng phân phối Python Anaconda.

✓ **Nhược điểm:** Các nhà phát triển Python có kinh nghiệm hơn có thể cảm thấy Spyder quá đơn giản để làm việc hàng ngày và thay vào đó chọn một IDE hoàn chỉnh hơn hoặc một giải pháp biên tập có khả năng tùy chỉnh.

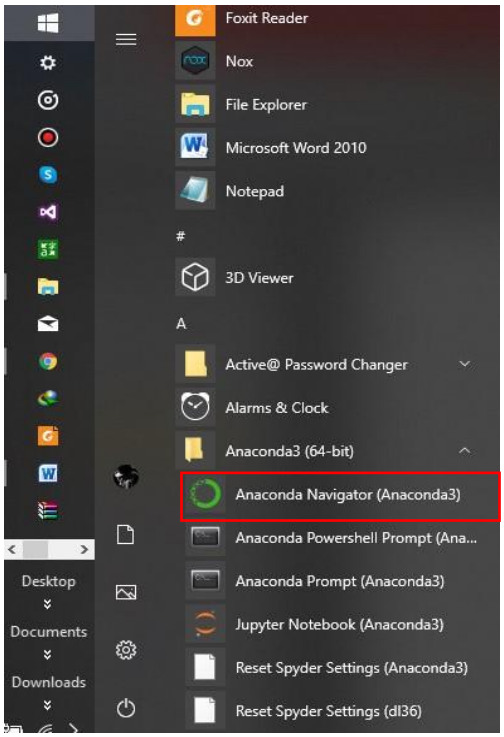
2.2. Hướng dẫn cài đặt IDE Spyder bằng Navigator

Spyder tương đối dễ cài đặt trên Windows, Linux và macOS. Chỉ cần chắc chắn để đọc và làm theo các hướng dẫn cẩn thận.

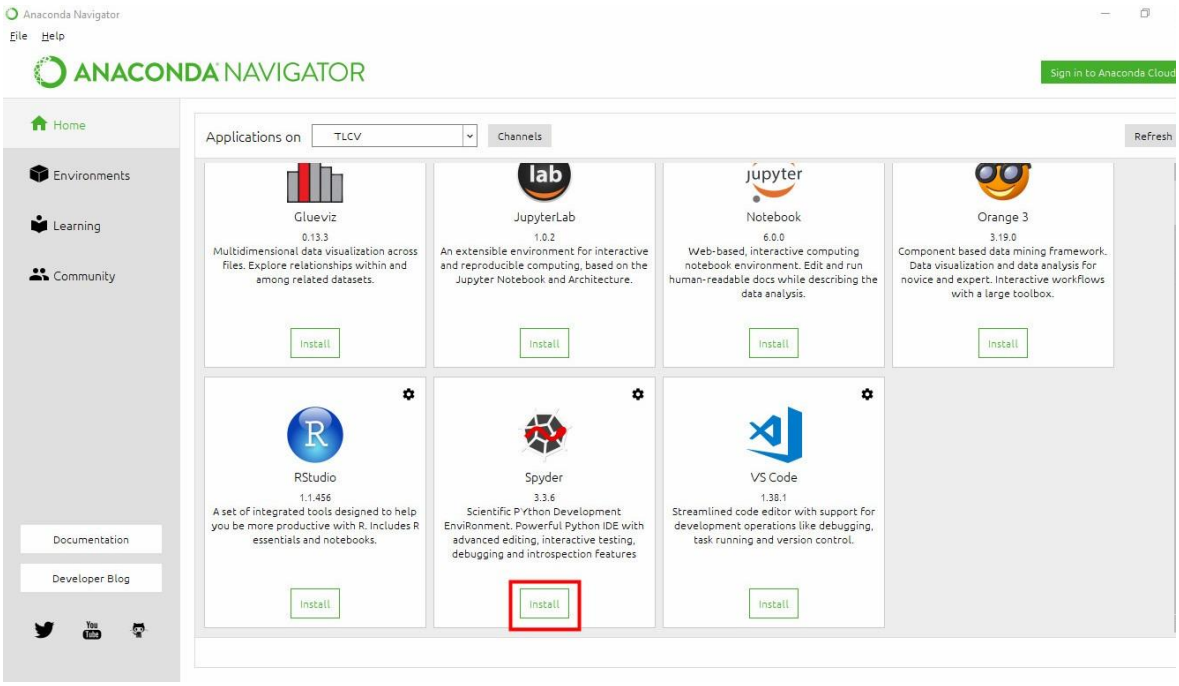
Spyder được bao gồm theo mặc định trong bản phân phối Anaconda Python, đi kèm với mọi thứ bạn cần để bắt đầu trong gói tất cả trong một (thường là khi cài đặt Anaconda thì Spyder đã được mặc định cài đặt).

Đây là cách dễ nhất để cài đặt Spyder cho bất kỳ nền tảng được hỗ trợ nào và là cách khuyên bạn nên tránh các sự cố không mong muốn. Bạn nên cài đặt thông qua phương pháp này; nó thường có ít khả năng gây ra những cạm bẫy tiềm tàng cho những người không phải là chuyên gia và có thể cung cấp hỗ trợ hạn chế nếu gặp rắc rối.

Đầu tiên bạn cần khởi động Anaconda Navigator



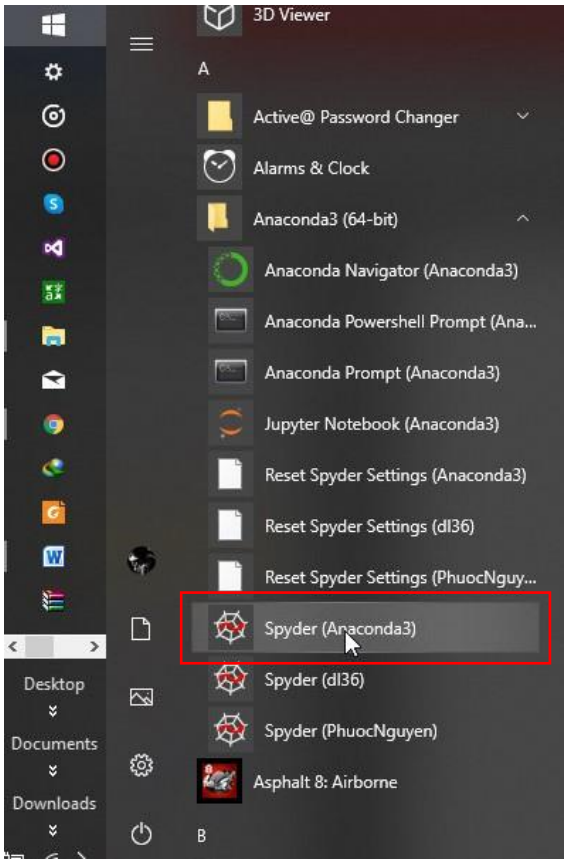
Sau đó bạn vào phần Home rồi cài đặt Spyder theo phiên bản mà Anaconda hỗ trợ



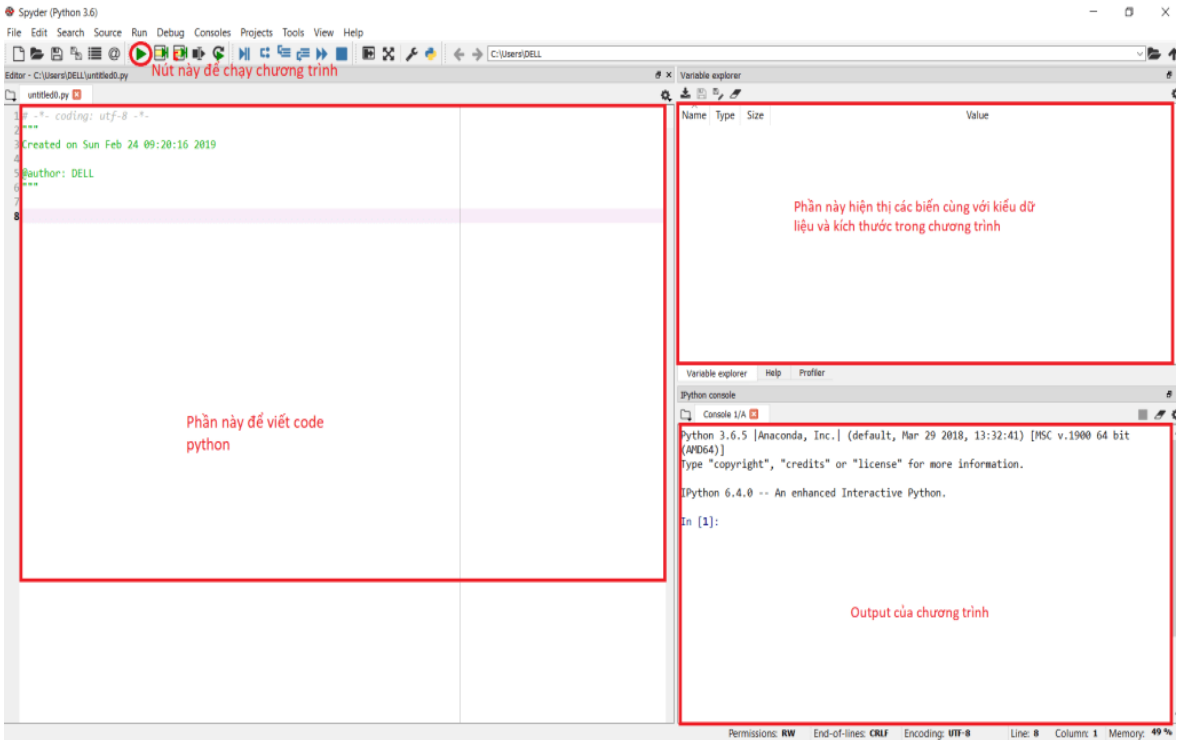
Chờ ít phút để chương trình cài đặt hoàn thành.

2.3. *Hướng dẫn sử dụng Spyder*

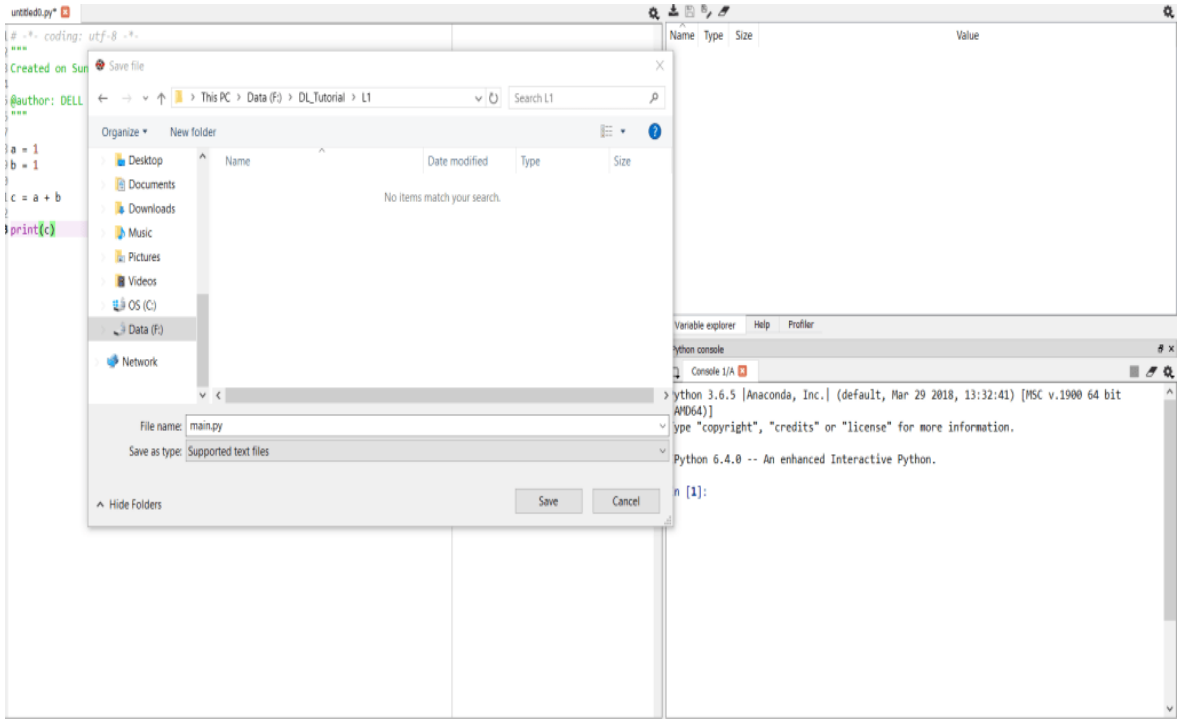
Sau khi cài đặt Spyder hoàn thành, mở Spyder lên và sử dụng.



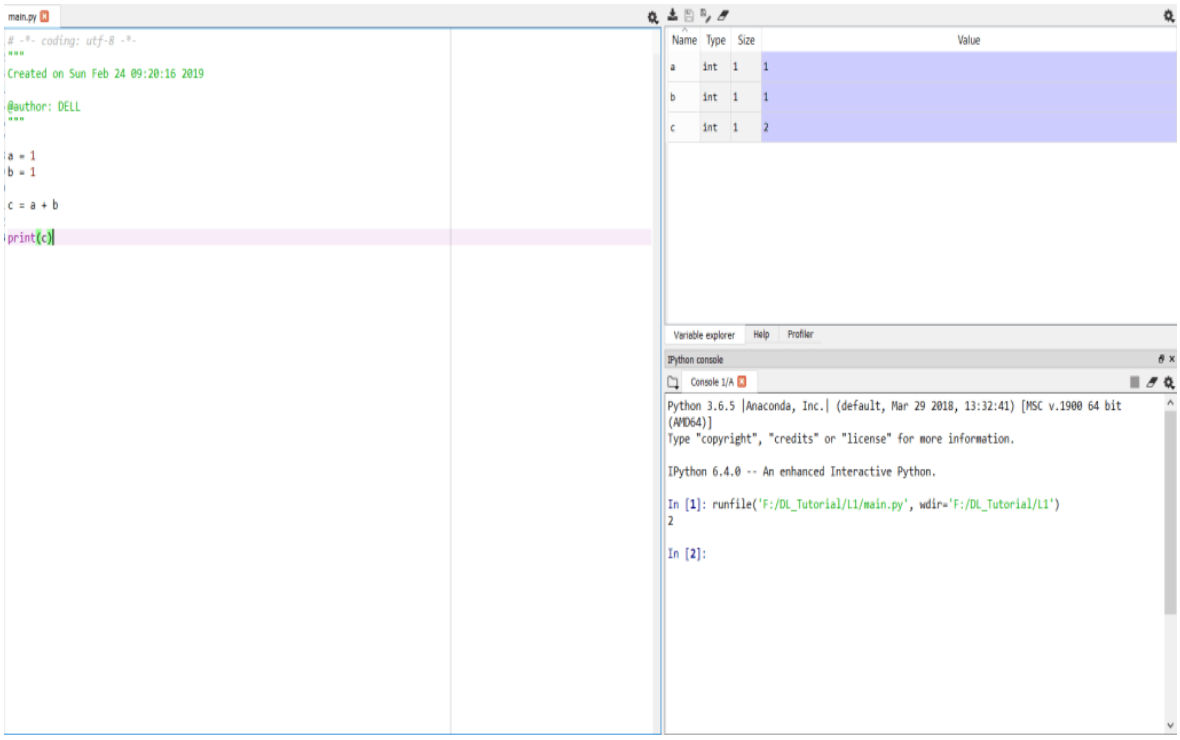
Giao diện chính của Spyder



Khi bạn viết code xong ở phần viết code, ấn nút chạy chương trình (hoặc F5) thì bạn cần phải lưu file trước nếu file chưa được lưu.



Chọn thư mục để lưu vào viết tên file, tên file ***luôn có .py đằng sau***, ví dụ như trong hình là ***main.py***



CHƯƠNG III. GIẢI CÁC BÀI TẬP MÔN LẬP TRÌNH CĂN BẢN

1. Bài Tập Nhóm A

Câu 1.Viết chương trình thực hiện các yêu cầu sau:

- **Nhập:** từ tập tin văn bản Cau1A.inp số nguyên dương N, N <= 10^6.
- **Xử lý:** tìm tất cả các số nguyên tố không lớn hơn N.
- **Xuất:** ra tập tin văn bản Cau1A.out số lượng C của số nguyên tố.

VD: Nếu N = 11 thì C = 5 số nguyên tố không lớn hơn 11 là 2, 3, 5, 7, 11.

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin*: Cau1A.py
- ❖ *Nội dung:*

```
  import math

  in_file = open('Cau1A.INP','r') #với 'r' mở để đọc nội
dung
  out_file = open('Cau1A.OUT','w') # mở để đọc nội dung
với 'w'

  #hàm nhập
  def nhap():
      indata = in_file.read() #đọc dữ liệu từ file
      n = int(indata)
      return n

  # Hàm kiểm tra 1 số nguyên dương x có phải là số nguyên
tố?
  def ktnt(n):
      k = int(math.sqrt(n)) + 1
      for i in range(2, k):
          if n % i == 0:
              return False
      return True
```

```
  # hàm xử lý bài toán
  def xuly(n):
     for i in range(2,n+1):
         if ktnt(i) == True:
            out_file.write('%d\n'%i) #ghi nội dung vào file

  # Hàm tương đương với hàm main trong C/C++
  def main():
      n=nhap()
      xuly(n)
      in_file.close() #đóng file input
      out_file.close() #đóng file đóng file output
      print ("done")

  # Gọi thực hiện hàm main
  if __name__ == "__main__":
      main()
```

Câu 2. chương trình thực hiện các yêu cầu sau:

- **Nhập**: từ tập tin văn bản Cau2A.inp số nguyên dương N, N < <= 10^6.
- **Xử lý**: tìm tất cả các ước số nguyên tố của N.
- **Xuất**: ra tập tin văn bản Cau2A.out các ước số nguyên tố của N trên cùng cột dòng, các số cách nhau đúng một khoảng trắng.

VD: nếu N=360 thì N có các ước số nguyên tố là 2, 3, 5

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau2A.py
- ❖ *Nội dung:*

```
  import math

  in_file = open('Cau2A.INP','r') #với 'r' mở để đọc nội
dung
  out_file = open('Cau2A.OUT','w') # mở để đọc nội dung với
'w'
```

```
  #hàm Nhập
  def nhap():
      indata = in_file.read() #đọc dữ liệu từ file
      n = int(indata)
      return n

  # Hàm kiểm tra 1 số nguyên dương x có phải là số nguyên tố?
  def ktnt(n):
      k = int(math.sqrt(n)) + 1
      for i in range(2, k):
          if n % i == 0:
              return False
      return True

  #Hàm xử lý bài toán
  def xuly(n):
       for i in range(2, n+1):
            if n%i==0:
                if ktnt(i) == True :
                    out_file.write('%d, '%i) #ghi nội dung vào
file

  # Hàm tương đương với hàm main trong C/C++
  def main():
     n=nhap()
     xuly(n)
     in_file.close() #đóng file input
     out_file.close() #đóng file đóng file output
     print ("done")

  # Gọi thực hiện hàm main
  if __name__ == "__main__":
      main()
```

Câu 3. Chương trình thực hiện các yêu cầu sau:

- **Nhập:** Từ tập tin văn bản Cau3A.inp số nguyên dương N, N <=10^6.
- **Xử lý:** Tính S là tổng các chữ số của N.
- **Xuất:** ra tập tin Cau3A.out giá trị của S.

VD: Nếu N = 423157698 thì S = 4+2+3+1+5+7+6+9+8=45.

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện

- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau3A.py
- ❖ *Nội dung:*

```
   in_file = open('Cau3A.INP','r') #với 'r' mở để đọc nội
dung
   out_file = open('Cau3A.OUT','w') # mở để đọc nội dung
với 'w'

   #hàm Nhập
   def nhap():
       indata = in_file.read() #đọc dữ liệu từ file
       n = int(indata)
       return n

   # hàm xử lý bài toán
   def xuly(n):
       s = 0
       while (n!=0) :
           s+= n%10
           n = n//10
       out_file.write('%d'%(s)) #ghi nội dung vào file

   # Hàm tương đương với hàm main trong C/C++
   def main():
       n=nhap()
       xuly(n)
       in_file.close() #đóng file input
       out_file.close() #đóng file đóng file output
       print ("done")

   # gọi thực hiên hàm main
   if __name__=="__main__":
       main()
```

Câu 4. chương trình thực hiện các yêu cầu sau:

- **Nhập:** Từ tập tin văn bản Cau4A.inp số nguyên dương N, N<= 10^6.
- **Xử lý:** Tính kết quả biểu thưc đan dấu với S với:
  S = N^2 - (N-1)^2 + (N-2)^2-...1.
- **Xuất:** ra tập tin văn bản Cau4A.out giá trị của S

Vd: Nếu N = 5 thì S = 5^2 - 4^2 + 3^2 - 2^2 + 1^2 = 15

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm, nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau4A.py
- ❖ *Nội dung:*

```
in_file = open('Cau4A.INP','r') #với 'r' mở để đọc nội dung
out_file = open('Cau4A.OUT','w') # mở để đọc nội dung với 'w'

#hàm Nhập
def nhap():
    indata = in_file.read() #đọc dữ liệu từ file
    n = int(indata)
    return n

# hàm xử lý bài toán
def xuly(n):
    s = 0
    dau = -1
    for i in range(1,n+1):
        dau = -dau
        s = s + i*dau**2
    out_file.write('%d'%(s)) #ghi nội dung vào file

# Hàm tương đương với hàm main trong C/C++
def main():
    n=nhap()
    xuly(n)
    in_file.close() #đóng file input
    out_file.close() #đóng file đóng file output
    print ("done")

# gọi thực hiên hàm main
if __name__=="__main__":
    main()
```

Câu 5. chương trình thực hiện các yêu cầu sau:

- **Nhập**: Từ tập tin văn bản Cau5A.inp 2 số nguyên dương M,N, M,N <= 10^9.
- **Xử lý**: tìm ước số chung lướn nhất U và Bội số chung nhỏ nhất B của 2 số M,N.
- **Xuất**: ra tập tin văn bản Cau5A.out giá trị của U và B.

VD: Nếu M= 24, và N= 36 thì U=12 và B= 72.

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau5A.py
- ❖ *Nội dung:*

```
in_file = open('Cau5A.INP','r') #với 'r' mở để đọc nội dung
out_file = open('Cau5A.OUT','w') # mở để đọc nội dung với 'w'

# hàm nhập m,n
def nhap():
    m,n = [int(i) for i in next(in_file).split()]
    return m,n

 # Hàm tính USCLN bằng thuật toán Euclid
def uscln(m, n):
     while n != 0:
         r = m % n
         m = n
         n = r
     return m

# Hàm BSCNN "gián tiếp" dựa vào USCLN
def bscnn(m,n):
    return (m * n) // uscln(m,n)

# Hàm tương đương với hàm main trong C/C++
def main():
    #Nhập và kiểm tr dữ liệu nhập
    while True:
        m,n = nhap()
```

```
        if m ** 2 + n ** 2 > 0:
            break
# Tìm BSCNN USCLN
    out_file.write("U = %d" %(uscln(m, n)))
    out_file.write("\tB = %d" %(bscnn(m, n)))
    in_file.close() #đóng file input
    out_file.close() #đóng file đóng file output
    print ("done")


# gọi thực hiên hàm main
if __name__=="__main__":
    main()
```

Câu 6. chương trình thực hiện các yêu cầu sau:

- **Nhập:** từ tập tin văn bản Cau6A.inp 3 số nguyên dương A,B,C có giá trị nằm trong đoạn [-10^3, 10^3].
- **Xử lý:** kiểm tra điều kiện 3 cạnh của 1 tam giác của A,B,C.

Nếu : Đúng: Tính diện tích của tam giác, kết quả làm tròn đến hai chữ số thập phân.

Ngược lại: tìm giá trị nhỏ nhất MIN của 3 số A,B,C.

- **Xuất** ra tập tin văn bản Cau6A.out giá trị của S hoặc MIN.

VD: Nếu A =3, B=5, C=4 thì S = 6.00.

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau6A.py
- ❖ *Nội dung:*

```
import math

in_file = open('Cau6A.INP','r') #với 'r' mở để đọc nội dung
out_file = open('Cau6A.OUT','w') # mở để đọc nội dung với 'w'

def nhap():
    a,b,c = [float(i) for i in next(in_file).split()]
    return a,b,c
```

```
# Hàm Xử lý tam giác
def tamgiac(a,b,c):
    if (a+b)>c and (a+c)>b and (b+c)>a:
        p = (a+b+c)/2
        s = math.sqrt(p*(p-a)*(p-b)*(p-c))
        out_file.write("%.2f"%(s)) #ghi nội dung vào file
    else:
        min = a
        if min > b:
            min = b
        if min > c:
            min = c
        out_file.write("%.2f"%(min)) #ghi nội dung vào file

# Hàm tương đương với hàm main trong C/C++
def main():
   a,b,c = nhap()
   tamgiac(a,b,c)
   in_file.close() #đóng file input
   out_file.close() #đóng file đóng file output
   print ("done")

# Gọi thực hiện hàm main
if __name__ == "__main__":
    main()
```

Câu 7. chương trình thực hiện các yêu cầu sau:

- **Nhập:** Từ tập tin văn bản Cau7A.inp 3 số nguyên A,B,C có giá trị năng trong đoạn [-10^3, 10^3].
- **Xử lý:** Giải phương trình Ax^2 + Bx+c = 0 trong trương hợp nghiệm thực, kết quả làm tròn đến 1 chữ số thập phân.
- **Xuất:** Ra tập tin văn bản Cau7A.out thong báo và các giá trị nghiệm nếu có.

VD: Nếu A=1, B=2,C=1 thì thông báo 'Phương trình có nghiệm kép x1=x2' và giá trị của nghiệm là -1.0.

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- *Tập tin:* Cau6A.py và thuvien_gptb2.py
- *Nội dung:*

```
in_file = open('Cau7A.INP','r') #với 'r' mở để đọc nội dung

# Hàm nhập hệ số a, b, c cho phương trình bậc 2
def nhap():
    a,b,c = [float(i) for i in next(in_file).split()]
    return a, b, c
    in_file.close() #đóng file input

# Hàm tính delta
def delta(a, b, c):
    return b ** 2 - 4 * a * c
```

```
from math import sqrt
from thuvien_gptb2 import nhap, delta

out_file = open('Cau7A.OUT','w') # mở để đọc nội dung với
'w'

# Hàm giải phương trình
def gpt(a, b, c):
    if a == 0:
        if b == 0:
            if c == 0:
                out_file.write("Phuong trinh VSN!") #ghi
nội dung vào file
            else:
                out_file.write("Phuong trinh VN!") #ghi nội
dung vào file
        else:
            if c == 0:
                out_file.write("Phuong trinh co 1 nghiem x
= 0 ")
            else:
                out_file.write("Phuong trinh co 1 nghiem:
x= %.1f" %(-b/a))
    else:
        d = delta(a, b, c)
        if d < 0:
            out_file.write("Phuong trinh VN")
        elif d == 0:
            out_file.write("Phuong trinh co nghiem kep: x1
= x2 = %.1f" %(-b / (2 * a)))
        else:
```

```
                d = sqrt(d)
                x1 = (-b + d) / (2 * a)
                x2 = (-b - d) / (2 * a)
                out_file.write("Phuong trinh co 2 nghiem phan
biet:\nx1 = %.1f \nx2 = %.1f" %(x1, x2))

    # Hàm tương đương với hàm main trong C/C++
    def main():
        a, b, c = nhap()
        gpt(a, b, c)
        out_file.close() #đóng file đóng file output
        print ("done")

    # Gọi thực hiện hàm main
    if __name__ == "__main__":
        main()
```

Câu 8. chương trình thực hiện các yêu cầu sau:

- **Nhập:** từ tập tin văn bản Cau8A.inp sood nguyên dương N, N <10^6.
- **Xử lý:** Tìm tất cả các số hoàn hảo không lớn hơn N.

Biết X là một số hoàn hảo nếu X bằng tổng các ước số của nó, không kể X.

- **Xuất:** ra tập tin văn bản Cau8A.outdanh sách các số hoàn hảo nếu có và giá trị C là số lượng số hoàn hảo tìm được.

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau8A.py
- ❖ *Nội dung:*

```
in_file = open('Cau8A.INP','r') #với 'r' mở để đọc nội dung
out_file = open('Cau8A.OUT','w') # mở để đọc nội dung với 'w'

#hàm Nhập
def nhap():
    indata = in_file.read() #đọc dữ liệu từ file
    n = int(indata)
```

```
    return n

# Hàm kiểm tra 1 số nguyên dương x có phải là số hoàn hảo?
def isPerfectNumber(n):
    if n <= 1:
        return False;
    else:
        sumDivision = 0
        for i in range(1, n):
            if n % i == 0:
                sumDivision += i    #sumDivision = sumDivision
+ i
        return True if sumDivision == n else False
        #return sumDivision == number ? True : False;

# hàm xử lý bài toán
def xuly(n):
   for i in range(1,n):
       if isPerfectNumber(i) == 1 :
           out_file.write('%d\n'%i)

# Hàm tương đương với hàm main trong C/C++
def main():
   n=nhap()
   xuly(n)
   in_file.close() #đóng file input
   out_file.close() #đóng file đóng file output
   print ("done")

# Gọi thực hiện hàm main
if __name__ == "__main__":
    main()
```

Câu 9. chương trình thực hiện các yêu cầu sau:

- Nhập: Từ tập tin văn bản Cau9A.inp 2 số nguyên dương M, N, M,N<=10^6.

- Xử lý: Tìm tất cả các số nguyên tố nằm trong đoạn [M,N].

- Xuất: ra ra tập tin văn bản Cau9A.out danh sách các số nguyên tố, mỗi số cách nhau bởi 1 khoảng trắng.

VD: nếu M = 1, n=10 thì các số nguyên tố tìm được là 2,3,5,7**.**

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if

- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau9A.py
- ❖ *Nội dung:*

```
import math

in_file = open('Cau9A.INP','r') #với 'r' mở để đọc nội dung
out_file = open('Cau9A.OUT','w') # mở để đọc nội dung với 'w'

# hàm nhập m,n
def nhap():
    m,n = [int(i) for i in next(in_file).split()]
    return m,n

# Hàm kiểm tra 1 số nguyên dương x có phải là số nguyên tố?
def ktnt(x):
    k = int(math.sqrt(x)) + 1
    for i in range(2, k):
        if x % i == 0:
            return False
    return True

# Hàm tìm các số nguyên tố trong đoạn [m,n]
def dsnt(m, n):
    ds = []
    if m < 2:
        m = 2
    for i in range(m, n + 1):
        if ktnt(i):
            ds.append(i)
    return ds

# Hàm tương đương với hàm main trong C/C++
def main():
    # Nhập và kiểm tra dữ liệu nhập
    while True:
        m, n = nhap()
        if m < n:
            break

    # Đếm và liệt kê danh sách các số nguyên tố thuộc đoạn
[m, n]
    ds = dsnt(m, n)
```

```
    out_file.write("%s" %ds)
    in_file.close() #đóng file input
    out_file.close() #đóng file đóng file output
    print ("done")


# Gọi thực hiện hàm main
if __name__ == "__main__":
    main()
```

Câu 10. chương trình thực hiện các yêu cầu sau:

- **Nhập:** Từ tập tin văn bản Cau10A.inp số nguyên dương N, N<= 10^3.
- **Xử lý:** Tính kết quả của biểu thức đan dấu S với:
  S = 1^2 - 2^2 + 3^2 - ...N^2
- **Xuất:** ra tập tin văn bản Cau10A.out giá trị của S.
  VD: Nếu N = 5 thì S = 1^2 - 2^2 + 3^2 - 4^2 + 5^2 = 15

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau10A.py
- ❖ *Nội dung:*

```
in_file = open('Cau10A.INP','r') #với 'r' mở để đọc nội dung
out_file = open('Cau10A.OUT','w') # mở để đọc nội dung với 'w'

#hàm Nhập
def nhap():
    indata = in_file.read() #đọc dữ liệu từ file
    n = int(indata)
    return n

# hàm xử lý bài toán
def xuly(n):
    s = 0
    dau = -1
    for i in range(1,n+1):
```

```
        dau = -dau
        s = s + dau*i**2
    out_file.write('%d'%(s)) #ghi nội dung vào file

# Hàm tương đương với hàm main trong C/C++
def main():
    n=nhap()
    xuly(n)
    in_file.close() #đóng file input
    out_file.close() #đóng file đóng file output
    print ("done")
# gọi thực hiên hàm main
if __name__=="__main__":
    main()
```

2. Bài Tập Nhóm B

Câu 1. Viết chương trình thực hiện các yêu cầu sau:

- **Nhập:** từ tập tin văn bản Cau2.inp số nguyên dương N, N < 10^2 , và ma trận vuông A kích thước N hàng, N cột chứa các giá trị nguyên nằm trong đoạn [-10^4 ,10^4] .
- **Xử lý:** tìm giá trị lớn nhất MAX của ma trận A và tổng SC của các phần tử trên đường chéo chính.
- **Xuất:** ra tập tin văn bản Cau2.out giá trị của MAX và SC, mỗi giá trị trên một dòng.

Ví dụ: Nếu N = 3 và ma trận A = [(50, 40, 60), (10, 20, 80), (90, 0, 70)] thì MAX = 90 và SC = 50 + 20 + 70 = 140.

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau1B.py
- ❖ *Nội dung:*

```
file_1_read = open("Cau1B.INP")
file_1_write= open("Cau1B.OUT","w")
_m=file_1_read.read(1)
```

```
matrix=file_1_read.read()
matrix = [item.split() for item in matrix.split('\n')[:-1]]

#Hàm nhập
def nhap():
   m=int(_m)
   return m

#hàm xử lý chương trình
def xuly(m):
    max=0
    sum=0
    for i in range (0,m):
        for j in range (0,m):
            if(int(matrix[i][j]) > int(max)):
                max=matrix[i][j]
    for i in range (0,m):
        for j in range (0,m):
            if (i==j):
                sum+=int(matrix[i][j])
    file_1_write.write("MAX="+str(max)+"\nSC= "+ str(sum))

# Hàm tương đương với hàm main trong C/C++
def main():
   m=nhap()
   xuly(m)
   file_1_read.close() #đóng file input
   file_1_write.close() #đóng file đóng file output
   print ("done")

# Gọi thực hiện hàm main
if __name__ == "__main__":
    main()
```

Câu 2. Viết chương trình thực hiện các yêu cầu sau:

- **Nhập:** từ tập tin văn bản Cau2B.inp số nguyên dương N, N < 10^3 , và dãy số nguyên K gồm N phần tử có giá trị nằm trong đoạn [-10^9,10^9] .
- **Xử lý:** tìm giá trị lớn nhất MAX và vị trí đầu tiên POS của MAX trong dãy số K.
- **Xuất:** ra tập tin văn bản Cau2B.out giá trị của MAX và POS, mỗi giá trị trên 1 dòng.

Ví dụ: Nếu N = 8 và dãy K = (50, 40, 60, 10, 20, 60, 30, 10) thì MAX = 60 và POS = 3

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau2B.py
- ❖ *Nội dung:*

```
file_2_read = open("Cau2B.INP","r")
file_2_write= open("Cau2B.OUT","w")
indata = file_2_read.read(1)
mat=file_2_read.read().split()

#Hàm nhập
def nhap():
   n=int(indata)
   return n

#hàm xử lý chương trình
def xuly(n):
    max=0
    vt=0
    for i in range (0,n):
        if(int(mat[i]) > int(max)):
            max = mat[i]
    for i in range (0,n):
        if(int(mat[i]) == int(max)):
            vt =i+1
            break
    file_2_write.write("MAX= "+ max + "\tvà POS= "+ str(vt))

# Hàm tương đương với hàm main trong C/C++
def main():
   m=nhap()
   xuly(m)
   file_2_read.close() #đóng file input
   file_2_write.close() #đóng file đóng file output
   print ("done")

# Gọi thực hiện hàm main
if __name__ == "__main__":
    main()
```

Câu 3. Viết chương trình thực hiện các yêu cầu sau:

-Nhập: từ tập tin văn bản Cau3B.inp số nguyên dương N, N < 10^2 , và ma trận vuông A kích thước N hàng, N cột chứa các giá trị nguyên nằm trong đoạn [10^4 ,10^4] .

-Xử lý: tìm giá trị lớn nhất MAXC trên đường chéo chính và giá trị nhỏ nhất MINP trên đường chéo phụ của ma trận A.

-Xuất: ra tập tin văn bản Cau3B.out giá trị của MAXC và MINP, mỗi giá trị trên 1 dòng.

Ví dụ: Nếu N = 3 và ma trận A = [(50, 40, 60), (10, 20, 80), (90, 0, 70)] thì MAXC = 70 và MINP = 20.

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau3B.py
- ❖ *Nội dung:*

```
file_3_read = open("Cau3B.INP")
file_3_write= open("Cau3B.OUT","w")
_m=file_3_read.read(1)
matrix=file_3_read.read()
matrix = [item.split() for item in matrix.split('\n')[:-1]]

#Hàm nhập
def nhap():
   m=int(_m)
   return m

#hàm xử lý chương trình
def xuly(m):
    max=0
    min=9999
    #print("max",max)
    for i in range (0,m):
        for j in range (0,m):
            if(int(matrix[i][i]) > int(max)):
                max=matrix[i][i]
```

```
        for i in range (0,m):
            for j in range (0,m):
                if(int(matrix[i][m-i-1]) < int(min)):
                    min=matrix[i][m-i-1]
        file_3_write.write("MAXC= "+ max + "\nMINP= "+ min)

    # Hàm tương đương với hàm main trong C/C++
    def main():
       m=nhap()
       xuly(m)
       file_3_read.close() #đóng file input
       file_3_write.close() #đóng file đóng file output
       print ("done")

    # Gọi thực hiện hàm main
    if __name__ == "__main__":
        main()
```

Câu 4. Viết chương trình thực hiện các yêu cầu sau:

- **Nhập:** từ tập tin văn bản Cau4B.inp số nguyên dương N, N < 10^3 , và dãy số nguyên K gồm N phần tử có giá trị nằm trong đoạn [-10^9,10^9] .
- **Xử lý:** đếm số phần tử dương C và tính trung bình cộng AVE của dãy số K, AVE làm tròn tới 2 số sau dấu chấm thập phân.
- **Xuất:** ra tập tin văn bản Cau4B.out giá trị của C và AVE, mỗi giá trị trên một dòng

Ví dụ: Nếu N = 6 và dãy K = (-50, -40, -60, -10, -20, -30) thì C = 0 và AVE= -35.00.

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau4B.py
- ❖ *Nội dung:*

```
file_4_read = open("Cau4B.INP","r")
file_4_write= open("Cau4B.OUT","w")
indata = file_4_read.read(1)
```

```
matrix=file_4_read.read().split()

#Hàm nhập
def nhap():
   m=int(indata)
   return m

#hàm xử lý chương trình
def xuly(m):
    dem = 0
    for i in range (0,m):
        if(int(matrix[i])>0):
            dem=dem +1

    tong=0
    for i in range (0,m):
        tong=tong+int(matrix[i])
    TB=tong/m
    file_4_write.write("C= "+ str(dem) + "\nAVE= "+str(TB))

# Hàm tương đương với hàm main trong C/C++
def main():
   m=nhap()
   xuly(m)
   file_4_read.close() #đóng file input
   file_4_write.close() #đóng file đóng file output
   print ("done")

# Gọi thực hiện hàm main
if __name__ == "__main__":
    main()
```

Câu 5. Viết chương trình thực hiện các yêu cầu sau:

- **Nhập:** từ tập tin văn bản Cau5B.inp số nguyên dương N, N < 10^2 , và ma trận vuông A kích thước N hàng, N cột chứa các giá trị nguyên nằm trong đoạn [10^4 ,10^4] .
- **Xử lý:** tìm ma trận B là ma trận tích của A nhân với A.
- **Xuất:** ra tập tin văn bản Cau5B.out ma trận B.

Ví dụ: Nếu N = 3 và ma trận A = [(1, 0, 1), (0, 1, 2), (3, 2, 0)] thì ma trận B= [(4, 2, 1), (6, 5, 2), (3, 2, 7)].

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý

- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau5B.py
- ❖ *Nội dung:*

```
import numpy as np

file_5_read = open("Cau5B.INP")
file_5_write= open("Cau5B.OUT","w")
_n=file_5_read.read(1)
matrix=file_5_read.read()
matrix = [item.split() for item in matrix.split('\n')[:-1]]

# Hàm tương đương với hàm main trong C/C++
def main():
    A=np.array(matrix)# khoiwrtao ma tran với numpy

    file_5_write.write(str(np.multiply(A,A)))
    file_5_read.close() #đóng file input
    file_5_write.close() #đóng file đóng file output
    print ("done")

# Gọi thực hiện hàm main
if __name__ == "__main__":
    main()
```

Câu 6. Viết chương trình thực hiện các yêu cầu sau:

- Nhập: từ tập tin văn bản Cau6B.inp số nguyên dương N, N < 10^3 , và dãy số nguyên K gồm N phần tử có giá trị nằm trong đoạn [10^9,10^9] .

- Xử lý: tìm giá trị nhỏ nhất MIN và vị trí cuối cùng POS của MIN trong dãy số K.

- Xuất: ra tập tin văn bản Cau6B.out giá trị của MIN và POS, mỗi giá trị trên 1 dòng.

Ví dụ: Nếu N = 8 và dãy K = (50, 40, 60, 10, 20, 60, 30, 10) thì MIN = 10 và POS = 8

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý

- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau6B.py
- ❖ *Nội dung:*

```
file_6_read = open("Cau6B.INP","r")
file_6_write= open("Cau6B.OUT","w")

indata = file_6_read.read(1)
matrix=file_6_read.read().split()

#Hàm nhập
def nhap():
   m=int(indata)
   return m

#hàm xử lý chương trình
def xuly(m):
    min=matrix[0]
    vt=0
    for i in range (0,m):
        if(int(min) > int(matrix[i])):
            min=matrix[i]
    for i in range (0,m):
        if(int(min)==int(matrix[i])):
            vt = i+1
#            break
    file_6_write.write("MIN= "+ min + "\nPOS= "+ str(vt))

# Hàm tương đương với hàm main trong C/C++
def main():
   m=nhap()
   xuly(m)
   file_6_read.close() #đóng file input
   file_6_write.close() #đóng file đóng file output
   print ("done")

# Gọi thực hiện hàm main
if __name__ == "__main__":
    main()
```

Câu 7. Viết chương trình thực hiện các yêu cầu sau:

- **Nhập:** từ tập tin văn bản Cau7B.inp 2 số nguyên dương M, N, M , N 10^2 , và ma trận A kích thước M hàng, N cột chứa các giá trị nguyên nằm trong đoạn 10^4 ,10^4] .
- **Xử lý:** tìm ma trận B là ma trận chuyển vị của A.
- **Xuất:** ra tập tin văn bản Cau7B.out ma trận B.

Ví dụ: Nếu M = 2, N = 3 và ma trận A = [(50, 40, 60), (10, 20, 30)] thì ma trận B = [(50, 10), (40, 20), (60, 30)]

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau7B.py
- ❖ *Nội dung:*

```
import numpy as np

file_7_read = open("Cau7B.INP")
file_7_write= open("Cau7B.OUT","w")
_m=file_7_read.read(1)
_n=file_7_read.read(2)
matrix=file_7_read.read()
matrix = [item.split() for item in matrix.split('\n')[:-1]]

# Hàm tương đương với hàm main trong C/C++
def main():
    A=np.array(matrix)#khởi tạo matrix với numpy
    #hàm chuyển vị của thư viện numpy
    file_7_write.write(str(A.T))
    file_7_read.close() #đóng file input
    file_7_write.close() #đóng file đóng file output
    print ("done")

# Gọi thực hiện hàm main
if __name__ == "__main__":
    main()
```

Câu 8. Viết chương trình thực hiện các yêu cầu sau:

- **Nhập:** từ tập tin văn bản Cau8B.inp số nguyên dương N, N < 10^3 , và dãy số nguyên K gồm N phần tử có giá trị nằm trong đoạn [-10^9,10^9] .
- **Xử lý:** đếm số phần tử âm C. Nếu C > 0 thì tính trung bình cộng AVE của các phần tử âm trong dãy số K, AVE làm tròn tới 2 số sau dấu chấm thập phân.
- **Xuất**: ra tập tin văn bản Cau8B.out giá trị của C và AVE nếu C > 0, mỗi giá trị trên một dòng.

Ví dụ: Nếu N = 6 và dãy K = (-50, 40, 60, -10, 20, -30) thì C = 3 và AVE = -30.00

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau8B.py
- ❖ *Nội dung:*

```
file_8_read = open("Cau8B.INP","r")
file_8_write= open("Cau8B.OUT","w")

indata = file_8_read.read(1)
matrix=file_8_read.read().split()

#Hàm nhập
def nhap():
   m=int(indata)
   return m

#hàm xử lý chương trình
def xuly(m):
    dem = 0
    tong = 0
    for i in range (0,m):
        if(int(matrix[i])<0):
            dem=dem +1
            tong= tong + int(matrix[i])
    tbc=tong/dem
```

```
        file_8_write.write("C="+str(dem)+"\nAVE= "+ str(tbc))

    # Hàm tương đương với hàm main trong C/C++
    def main():
       m=nhap()
       xuly(m)
       file_8_read.close() #đóng file input
       file_8_write.close() #đóng file đóng file output
       print ("done")

    # Gọi thực hiện hàm main
    if __name__ == "__main__":
        main()
```

Câu 9. Viết chương trình thực hiện các yêu cầu sau:

- **Nhập:** từ tập tin văn bản Cau9B.inp số nguyên dương N, N 10^3 , và dãy số nguyên K gồm N phần tử có giá trị nằm trong đoạn [10^9,109] .
- **Xử lý:** Sắp xếp dãy K theo thứ tự tăng dần.
- **Xuất:** ra tập tin văn bản Cau9B.out dãy số K sau khi đã sắp xếp, mỗi phần tử cách nhau đúng 1 khoảng trắng.

Ví dụ: Nếu N = 8 và dãy K = (50, 40, 60, 10, 20, 60, 30, 10) thì dãy K sau khi sắp xếp tăng dần là K = (10, 10, 20, 30, 40, 50, 60, 60).

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau9B.py
- ❖ *Nội dung:*

```
    file_9_read = open("Cau9B.INP","r")
    file_9_write= open("Cau9B.OUT","w")
    indata = file_9_read.read(1)
    matrix=file_9_read.read().split()

    #Hàm nhập
    def nhap():
       m=int(indata)
```

```
    return m

#hàm xử lý chương trình
def xuly(m):
    x=0
    for i in range (0,m):
        for j in range (i,m):
            if(matrix[i]> matrix[j]):
                x=matrix[i]
                matrix[i]=matrix[j]
                matrix[j]=x
    file_9_write.write("K= "+ str(matrix))

# Hàm tương đương với hàm main trong C/C++
def main():
   m=nhap()
   xuly(m)
   file_9_read.close() #đóng file input
   file_9_write.close() #đóng file đóng file output
   print ("done")

# Gọi thực hiện hàm main
if __name__ == "__main__":
    main()
```

Câu 10. Viết chương trình thực hiện các yêu cầu sau:

- **Nhập:** từ tập tin văn bản Cau2.inp số nguyên dương N, N <10^3 , và dãy số nguyên K gồm N phần tử có giá trị nằm trong đoạn [-10^9,10^9] .
- **Xử lý:** đếm số phần tử dương C của dãy K. Nếu C > 0 thì tìm giá trị dương nhỏ nhất MIN.
- **Xuất:** ra tập tin văn bản Cau2.out giá trị của C và MIN nếu C > 0, mỗi giá trị trên 1 dòng.

Ví dụ: Nếu N = 8 và dãy K = (50, 40, 60, -10, 20, -60, 30, 10) thì C = 6 và MIN = 10

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- *Tập tin:* Cau10B.py
- *Nội dung:*

```
file_10_read = open("Cau10B.INP","r")
file_10_write= open("Cau10B.OUT","w")
indata = file_10_read.read(1)
matrix=file_10_read.read().split()

#Hàm nhập
def nhap():
   m=int(indata)
   return m

#hàm xử lý chương trình
def xuly(m):
    dem = 0
    min=9999999
    for i in range (0,m):
        if(int(matrix[i])>0):
            dem=dem +1
            if (int(matrix[i])<int(min)):
                min=matrix[i]
    file_10_write.write("C="+str(dem)+"\nMIN= "+ str(min))

# Hàm tương đương với hàm main trong C/C++
def main():
   m=nhap()
   xuly(m)
   file_10_read.close() #đóng file input
   file_10_write.close() #đóng file đóng file output
   print ("done")

# Gọi thực hiện hàm main
if __name__ == "__main__":
    main()
```

3. Bài Tập Nhóm C

Câu 1. Viết chương trình thực hiện các yêu cầu sau:

- Nhập: từ tập tin văn bản Cau1C.inp xâu ký tự S có không quá 200 ký tự.

- Xử lý:

+ Loại bỏ các ký tự không phải ký tự chữ và khoảng trắng khỏi S.

+ Sau đó, đếm số từ C của S. Biết: từ là một nhóm các ký tự liên tiếp khác khoảng trắng.

- Xuất: ra tập tin văn bản Cau1C.out giá trị của C.

Ví dụ: nếu S = ‘ Cong1 nghe2 34 tho@ng tin5 ’ thì S có C = 4 từ là ‘Cong’, ‘nghe’, ‘thong’, ‘tin’

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Xử lý chuỗi
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau1C.py
- ❖ *Nội dung:*

```
import os

#GET FILE
file_size = os.path.getsize("Cau1C.INP")
#Mở đọc file
file_1_read = open("Cau1C.INP","r")
data_file_1 = file_1_read.read()
#Mở viết file
file_1_write= open("Cau1C.OUT","w")

#hàm xử lý chương trình
def xuly():
    #GET SIZE FILE
    output = ""
    dem=0
    for i in range(file_size):
        #kiểm tra xem ký tự đã truyền có phải là chữ cái không
hoặc khoảng cách ko?.
```

```
        if data_file_1[i].isalpha() or data_file_1[i] == " ":
            output = output+data_file_1[i]
            x=output
           # xóa hai dau cach trùng nhau hoặc cuối câu
            x=" ".join(x.split())

    for i in range (file_size): # dem so chu trong day
        if data_file_1[i]==" ":
            dem=dem+1
    file_1_write.write("S=" + x+"\nC= "+str(dem))

# Hàm tương đương với hàm main trong C/C++
def main():
   print ("Cau1_Nhom C")

   #CHECK EXIST FILE
   check_file = os.path.exists("Cau1C.INP")
   if check_file == False:
       print ("Ban can tao file input")

   print ("file_size_1: ",file_size)
   print ("Data_file_1: ",data_file_1)

   xuly()
   file_1_read.close() #đóng file input
   file_1_write.close() #đóng file đóng file output
   print ("done")

# Gọi thực hiện hàm main
if __name__ == "__main__":
    main()
```

Câu 2. Viết chương trình thực hiện các yêu cầu sau:

- **Nhập:** từ tập tin văn bản Cau2C.inp xâu ký tự S có không quá 200 ký tự**.**
- **Xử lý:**
  + Loại bỏ các ký tự không phải ký tự chữ và khoảng trắng khỏi S.
  + Sau đó, chuyển thành xâu ký tự X là dạng Title Case của S.
- **Xuất:** ra tập tin văn bản Cau2C.out xâu ký tự X.

Ví dụ: Nếu S = ‘ Khoa1 Cong nghe thong@ tin’ thì X = ‘ Khoa Cong Nghe Thong Tin’.

Mục đích:

- ✓ Viết 1 chương trình Python

- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Xử lý chuỗi
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau2C.py
- ❖ *Nội dung:*

```
  import os

  #GET FILE
  file_size = os.path.getsize("Cau2C.INP")
  #Mở đọc file
  file_2_read = open("Cau2C.INP","r")
  data_file_2 = file_2_read.read()
  #Mở viết file
  file_2_write= open("Cau2C.OUT","w")

  #hàm xử lý chương trình
  def xuly():
      #GET SIZE FILE
      output = ""
      for i in range(file_size):
          #kiểm tra xem ký tự đã truyền có phải là chữ cái không
hoặc khoảng cách ko?.
          if data_file_2[i].isalpha() or data_file_2[i] == " ":
              output = output+data_file_2[i]
              x=output
              x=" ".join(x.split()) # xóa hai dau cach trùng
nhau hoặc cuôi câu
              b=x.title()     # hàm in hoa chữ cái đầu tiên
      file_2_write.write(b)

  # Hàm tương đương với hàm main trong C/C++
  def main():
     print ("Cau2_Nhom C")

     #CHECK EXIST FILE
     check_file = os.path.exists("Cau2C.INP")
     if check_file == False:
         print ("Ban can tao file input")

     print ("file_size_2: ",file_size)
     print ("Data_file_2: ",data_file_2)
```

```
    xuly()
    file_2_read.close() #đóng file input
    file_2_write.close() #đóng file đóng file output
    print ("done")

# Gọi thực hiện hàm main
if __name__ == "__main__":
    main()
```

Câu 3. Viết chương trình thực hiện các yêu cầu sau:

- **Nhập:** từ tập tin văn bản Cau3C.inp xâu ký tự S có không quá 50 ký tự là họ tên Việt Nam của 1 người, trong đó có một số ký tự gõ nhầm (không phải ký tự chữ và gõ thừa khoảng trắng).
- **Xử lý:**
  + Loại bỏ các ký tự không phải ký tự chữ và khoảng trắng khỏi S.
  + Sau đó, tìm xâu ký tự X là tên của người đó.
- **Xuất:** ra tập tin văn bản Cau3C.out xâu ký tự X.

Ví dụ: Nếu S = ‘ Nguyen1 Van234 A5n67h ’ thì X = ‘Anh’

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Xử lý chuỗi
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau3C.py
- ❖ *Nội dung:*

```
import os

#GET FILE
file_size = os.path.getsize("Cau3C.INP")
#Mở đọc file
file_3_read = open("Cau3C.INP","r")
data_file_3 = file_3_read.read()
#Mở viết file
file_3_write= open("Cau3C.OUT","w")

#hàm xử lý chương trình
```

```
  def xuly():
      #GET SIZE FILE
      output = ""
      for i in range(file_size):
          #kiểm tra xem ký tự đã truyền có phải là chữ cái
không hoặc khoảng cách ko?.
          if data_file_3[i].isalpha() or data_file_3[i] == " ":
              output = output+data_file_3[i]
              x=output
             # xóa hai dau cach trùng nhau hoặc cuối câu
              x=" ".join(x.split())

      #tim tên
      for i in range(0,len(x)):
          if (x[i]==" "):
              a=(x[i:]) # lấy tất cả các ki tu trong chuổi từ
vị trí i đến hết
              b=a.strip()    #xóa dau cach đầu và cuối câu
      file_3_write.write(b)

  # Hàm tương đương với hàm main trong C/C++
  def main():
     print ("Cau3_Nhom C")

     #CHECK EXIST FILE
     check_file = os.path.exists("Cau3C.INP")
     if check_file == False:
         print ("Ban can tao file input")

     print ("file_size_3: ",file_size)
     print ("Data_file_3: ",data_file_3)

     xuly()
     file_3_read.close() #đóng file input
     file_3_write.close() #đóng file đóng file output
     print ("done")

  # Gọi thực hiện hàm main
  if __name__ == "__main__":
      main()
```

Câu 4. Viết chương trình thực hiện các yêu cầu sau:

- Nhập: từ tập tin văn bản Cau4C.inp xâu ký tự S có không quá 200 ký tự.

- Xử lý:

+ Loại bỏ các ký tự không phải ký tự chữ và khoảng trắng khỏi S.

+ Sau đó, chuyển thành xâu ký tự X là dạng xâu chuẩn của S. Biết: xâu chuẩn là 1 xâu ký tự mà không có 2 khoảng trắng liên tiếp, bắt đầu và kết thúc không phải là khoảng trắng.

- Xuất: ra tập tin văn bản Cau4C.out xâu ký tự X.

Ví dụ: Nếu S = ‘ Co@ng1 nghe2 34 th$ong tin5 ’ thì X = ‘Cong nghe thong tin’

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Xử lý chuỗi
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau4C.py
- ❖ *Nội dung:*

```
import os

#GET FILE
file_size = os.path.getsize("Cau4C.INP")
#Mở đọc file
file_4_read = open("Cau4C.INP","r")
data_file_4 = file_4_read.read()
#Mở viết file
file_4_write= open("Cau4C.OUT","w")

#hàm xử lý chương trình
def xuly():
    #GET SIZE FILE
    output = ""
    for i in range(file_size):
        #kiểm tra xem ký tự đã truyền có phải là chữ cái
không hoặc khoảng cách ko?.
        if data_file_4[i].isalpha() or data_file_4[i] == "
":
```

```
            output = output+data_file_4[i]
            x=output
            # xóa hai dau cach trùng nhau hoặc cuôi câu
            x=" ".join(x.split())
        file_4_write.write(x)

# Hàm tương đương với hàm main trong C/C++
def main():
    print ("Cau4_Nhom C")

    #CHECK EXIST FILE
    check_file = os.path.exists("Cau4C.INP")
    if check_file == False:
        print ("Ban can tao file input")

    print ("file_size_4: ",file_size)
    print ("Data_file_4: ",data_file_4)

    xuly()
    file_4_read.close() #đóng file input
    file_4_write.close() #đóng file đóng file output
    print ("done")

# Gọi thực hiện hàm main
if __name__ == "__main__":
    main()
```

Câu 5. Viết chương trình thực hiện các yêu cầu sau:

- Nhập: từ tập tin văn bản Cau5C.inp xâu ký tự S có không quá 200 ký tự.

- Xử lý:

+ Loại bỏ các ký tự không phải ký tự chữ khỏi S để được xâu X.

+ Sau đó, kiểm tra xem X có phải là một xâu ghép. Biết: xâu ghép là 1 xâu ký tự được tạo thành bằng cách viết liên tiếp K lần (K > 1) 1 xâu có độ dài ngắn hơn.

- Xuất: ra tập tin văn bản Cau5C.out xâu ký tự X và giá trị lớn nhất của K nếu X là xâu ghép.

Ví dụ: Nếu S = ‘AB@1 A2 BA B$ A5B ’ thì X = ‘ABABABAB’ và K = 4.

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý

- ✓ Xử lý chuỗi
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau5C.py
- ❖ *Nội dung:*

```
import os

#GET FILE
file_size = os.path.getsize("Cau5C.INP")
#Mở đọc file
file_5_read = open("Cau5C.INP","r")
data_file_5 = file_5_read.read()
#Mở viết file
file_5_write= open("Cau5C.OUT","w")

#hàm xử lý chương trình
def xuly():
    #GET SIZE FILE
    output = ""
    b=""
    max=0
    for i in range(file_size):
        #kiểm tra xem ký tự đã truyền có phải là chữ cái
không hoặc khoảng cách ko?.
        if data_file_5[i].isalpha() or data_file_5[i] == " ":
            output = output+data_file_5[i]
            x=output
            x="".join(x.split())

    for i in range(0,len(x)):
        if(x.count(x[i])>1):
           b="la xau ghep"
           x.count(x[i])>max
           max=x.count(x[i])
        else:
           b="khong phai xau ghep"
    print(b)
    file_5_write.write("X: "+ x + "\nK: " + str(max))

# Hàm tương đương với hàm main trong C/C++
def main():
   print ("Cau5_Nhom C")
```

```
    #CHECK EXIST FILE
    check_file = os.path.exists("Cau5C.INP")
    if check_file == False:
        print ("Ban can tao file input")

    print ("file_size_5: ",file_size)
    print ("Data_file_5: ",data_file_5)

    xuly()
    file_5_read.close() #đóng file input
    file_5_write.close() #đóng file đóng file output
    print ("done")

# Gọi thực hiện hàm main
if __name__ == "__main__":
    main()
```

Câu 6. Viết chương trình thực hiện các yêu cầu sau:

- **Nhập:** từ tập tin văn bản Cau6C.inp xâu ký tự S có không quá 200 ký tự.
- **Xử lý:**
  + Loại bỏ các ký tự không phải ký tự chữ và khoảng trắng khỏi S.
  + Sau đó, tìm MAX là số ký tự của từ dài nhất trong xâu S. Biết: từ là một nhóm các ký tự liên tiếp khác khoảng trắng.
- **Xuất:** ra tập tin văn bản Cau6C.out giá trị của MAX.

Ví dụ: nếu S = ‘ Cong1 nghe2 34 tho@ng tin5 ’ thì từ dài nhất là ‘thong’ có số ký tự là MAX =5.

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Xử lý chuỗi
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- *Tập tin:* Cau6C.py
- *Nội dung:*

```
  import os

  #GET FILE
  file_size = os.path.getsize("Cau6C.INP")
  #Mở đọc file
  file_6_read = open("Cau6C.INP","r")
  data_file_6 = file_6_read.read()
  #Mở viết file
  file_6_write= open("Cau6C.OUT","w")

  #hàm xử lý chương trình
  def xuly():
      #GET SIZE FILE
      output = ""
      for i in range(file_size):
          #kiểm tra xem ký tự đã truyền có phải là chữ cái không
hoặc khoảng cách ko?.
          if data_file_6[i].isalpha() or data_file_6[i] == " ":
              output = output+data_file_6[i]
              x=output
              x=" ".join(x.split()) # xóa hai dau cach trùng
nhau hoặc cuôi câu

      # cắt từ theo khoảng trắng,đặt từ vào mảng mới
      new_arr = x.split(" ")
      # đo chiều dài của từ
      # lấy một từ một lần, đếm các chữ cái liên tiếp
      # trả lại các chữ cái dài nhất liên tiếp
      file_6_write.write(str(len(max(new_arr, key=len))))

  # Hàm tương đương với hàm main trong C/C++
  def main():
     print ("Cau6_Nhom C")

     #CHECK EXIST FILE
     check_file = os.path.exists("Cau6C.INP")
     if check_file == False:
         print ("Ban can tao file input")

     print ("file_size_6: ",file_size)
     print ("Data_file_6: ",data_file_6)

     xuly()
```

```
    file_6_read.close() #đóng file input
    file_6_write.close() #đóng file đóng file output
    print ("done")

# Gọi thực hiện hàm main
if __name__ == "__main__":
    main()
```

Câu 7. Viết chương trình thực hiện các yêu cầu sau:

- Nhập: từ tập tin văn bản Cau7C.inp xâu ký tự S có không quá 200 ký tự.

- Xử lý:

+ Loại bỏ các ký tự không phải ký tự chữ và khoảng trắng khỏi S.
+ Sau đó, chuyển thành xâu ký tự X là dạng tOGGLE cASE của S.

- Xuất: ra tập tin văn bản Cau7C.out xâu ký tự X.

Ví dụ: Nếu S = ‘ Khoa1 Cong nghe thong@ tin’ thì X = ‘ kHOA cONG nGHE tHONG tIN’.

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Xử lý chuỗi
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau7C.py
- ❖ *Nội dung:*

```
import os

#GET FILE
file_size = os.path.getsize("Cau7C.INP")
#Mở đọc file
file_7_read = open("Cau7C.INP","r")
data_file_7 = file_7_read.read()
#Mở viết file
file_7_write= open("Cau7C.OUT","w")


#hàm xử lý chương trình
```

```
  def xuly():
      #GET SIZE FILE
      output = ""
      for i in range(file_size):
          #kiểm tra xem ký tự đã truyền có phải là chữ cái không
hoặc khoảng cách ko?.
          if data_file_7[i].isalpha() or data_file_7[i] == " ":
              output = output+data_file_7[i]
              x=output
              x=" ".join(x.split()) # xóa hai dau cach trùng
nhau hoặc cuôi câu
              b=x.title()     # hàm in hoa chữ cái đầu tiên
              c=b.swapcase() # đảo ngược kiểu HOA thành kiểu
thường và ngược lại.
      file_7_write.write(c)

  # Hàm tương đương với hàm main trong C/C++
  def main():
     print ("Cau7_Nhom C")

     #CHECK EXIST FILE
     check_file = os.path.exists("Cau7C.INP")
     if check_file == False:
         print ("Ban can tao file input")

     print ("file_size_7: ",file_size)
     print ("Data_file_7: ",data_file_7)

     xuly()
     file_7_read.close() #đóng file input
     file_7_write.close() #đóng file đóng file output
     print ("done")

  # Gọi thực hiện hàm main
  if __name__ == "__main__":
      main()
```

Câu 8. Viết chương trình thực hiện các yêu cầu sau:

- **Nhập**: từ tập tin văn bản Cau8C.inp xâu ký tự S có không quá 50 ký tự là họ tên Việt Nam của 1 người, trong đó có một số ký tự gõ nhầm (không phải ký tự chữ và gõ thừa khoảng trắng).
- **Xử lý:**
  + Loại bỏ các ký tự không phải ký tự chữ và khoảng trắng khỏi S.
  + Sau đó, tìm xâu ký tự X là họ của người đó**.**
- **Xuất:** ra tập tin văn bản Cau8C.out xâu ký tự X.

  Ví dụ: Nếu S = ‘ Nguyen1 Van234 A5n67h ’ thì X = ‘Nguyen’.

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Xử lý chuỗi
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau8C.py
- ❖ *Nội dung:*

```
  import os

  #GET FILE
  file_size = os.path.getsize("Cau8C.INP")
  #Mở đọc file
  file_8_read = open("Cau8C.INP","r")
  data_file_8 = file_8_read.read()
  #Mở viết file
  file_8_write= open("Cau8C.OUT","w")

  #hàm xử lý chương trình
  def xuly():
      #GET SIZE FILE
      output = ""
      for i in range(file_size):
          #kiểm tra xem ký tự đã truyền có phải là chữ cái không
hoặc khoảng cách ko?.
          if data_file_8[i].isalpha() or data_file_8[i] == " ":
              output = output+data_file_8[i]
              x=output
              x=" ".join(x.split()) # xóa hai dau cach trùng
nhau hoặc cuôi câu
              p=x.split()  # chia chuỗi str thành một dãy các
token được phân biệt riêng rẽ bởi dấu tách d
      file_8_write.write(p[0])

  # Hàm tương đương với hàm main trong C/C++
  def main():
     print ("Cau8_Nhom C")

     #CHECK EXIST FILE
     check_file = os.path.exists("Cau8C.INP")
     if check_file == False:
```

```
        print ("Ban can tao file input")

    print ("file_size_8: ",file_size)
    print ("Data_file_8: ",data_file_8)

    xuly()
    file_8_read.close() #đóng file input
    file_8_write.close() #đóng file đóng file output
    print ("done")

# Gọi thực hiện hàm main
if __name__ == "__main__":
    main()
```

Câu 9. Viết chương trình thực hiện các yêu cầu sau:

- Nhập: từ tập tin văn bản Cau9C.inp xâu ký tự S có không quá 200 ký tự.

- Xử lý:

+ Loại bỏ các ký tự không phải ký tự chữ và khoảng trắng khỏi S.

+ Sau đó, chuyển thành xâu ký tự X là dạng câu chuẩn của S. Biết: câu chuẩn là 1 xâu ký tự mà không có 2 khoảng trắng liên tiếp, bắt đầu và kết thúc

không phải là khoảng trắng, ký tự bắt đầu là ký tự chữ HOA.

- Xuất: ra tập tin văn bản Cau9C.out xâu ký tự X.

Ví dụ: Nếu S = ‘ co@ng1 nghe2 34 th$ong tin5 ’ thì X = ‘Cong nghe thong tin’.

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Xử lý chuỗi
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau9C.py

❖ *Nội dung:*

```
  import os

  #GET FILE
  file_size = os.path.getsize("Cau9C.INP")
  #Mở đọc file
  file_9_read = open("Cau9C.INP","r")
  data_file_9 = file_9_read.read()
  #Mở viết file
  file_9_write= open("Cau9C.OUT","w")

  #hàm xử lý chương trình
  def xuly():
      #GET SIZE FILE
      output = ""
      for i in range(file_size):
          #kiểm tra xem ký tự đã truyền có phải là chữ cái
không hoặc khoảng cách ko?.
          if data_file_9[i].isalpha() or data_file_9[i] == " ":
              output = output+data_file_9[i]
              x=output
              x=" ".join(x.split()) # xóa hai dau cach trùng
nhau hoặc cuôi câu
              b=x.capitalize() #hàm in hoa chữ cái đầu tiên của
chuỗi.
      file_9_write.write(b)

  # Hàm tương đương với hàm main trong C/C++
  def main():
     print ("Cau9_Nhom C")

     #CHECK EXIST FILE
     check_file = os.path.exists("Cau9C.INP")
     if check_file == False:
         print ("Ban can tao file input")

     print ("file_size_9: ",file_size)
     print ("Data_file_9: ",data_file_9)

     xuly()
     file_9_read.close() #đóng file input
     file_9_write.close() #đóng file đóng file output
     print ("done")

  # Gọi thực hiện hàm main
  if __name__ == "__main__":
      main()
```

Câu 10. Viết chương trình thực hiện các yêu cầu sau:

- Nhập: từ tập tin văn bản Cau10C.inp xâu ký tự S có không quá 200 ký tự.

- Xử lý:

+ Loại bỏ các ký tự không phải ký tự chữ khỏi S để được xâu X.

+ Sau đó, kiểm tra xem X có phải là một xâu ghép. Biết: xâu ghép là 1 xâu ký tự được tạo thành bằng cách viết liên tiếp K lần (K > 1) 1 xâu có độ dài ngắn hơn.

- Xuất: ra tập tin văn bản Cau10C.out xâu ký tự X và kết luận.

Ví dụ: Nếu S = 'AB@1 A2 BA B$ A5B ' thì X = 'ABABABAB' là xâu ghép

Mục đích:

- ✓ Viết 1 chương trình Python
- ✓ Sử dụng câu lệnh if
- ✓ Tổ chức chương trình một cách hợp lý
- ✓ Xử lý chuỗi
- ✓ Viết và gọi thư viện
- ✓ Viết và gọi hàm
- ✓ Viết nhập xuất file input/output

Chương trình:

- ❖ *Tập tin:* Cau10C.py
- ❖ *Nội dung:*

```
import os

#GET FILE
file_size = os.path.getsize("Cau10C.INP")
#Mở đọc file
file_10_read = open("Cau10C.INP","r")
data_file_10 = file_10_read.read()
#Mở viết file
file_10_write= open("Cau10C.OUT","w")

#hàm xử lý chương trình
def xuly():
    #GET SIZE FILE
    output = ""
    b=""
    for i in range(file_size):
        #kiểm tra xem ký tự đã truyền có phải là chữ cái không
hoặc khoảng cách ko?.
```

```
          if data_file_10[i].isalpha() or data_file_10[i] == " ":
              output = output+data_file_10[i]
              x=output
              x="".join(x.split()) # xóa hai dau cach trùng nhau
hoặc cuôi câu

      for i in range(0,len(x)):
          if(x.count(x[i])>1):
             b="la xau ghep"
          else:
             b="khong phai xau ghep"
      file_10_write.write("X= "+ x + " " + b)

  # Hàm tương đương với hàm main trong C/C++
  def main():
     print ("Cau10_Nhom C")

     #CHECK EXIST FILE
     check_file = os.path.exists("Cau10C.INP")
     if check_file == False:
         print ("Ban can tao file input")

     print ("file_size_10: ",file_size)
     print ("Data_file_10: ",data_file_10)

     xuly()
     file_10_read.close() #đóng file input
     file_10_write.close() #đóng file đóng file output
     print ("done")

  # Gọi thực hiện hàm main
  if __name__ == "__main__":
      main()
```

KẾT LUẬN

Đúng như những gì mà Guido Van Rossum mong muốn, Python mang tới nhiều tính năng nổi bật hơn so với những ngôn ngữ lập trình khác. Ví dụ như đơn giản, dễ học, miễn phí, sử dụng mã nguồn mở, khả năng di chuyển, khả năng mở rộng và có thể nhúng, ngôn ngữ thông dịch cấp cao, hướng đối tượng.

Cú pháp của ngôn ngữ lập trình Python rất đơn giản và dễ học hơn so với các ngôn ngữ lập trình khác như C#, Java, C++,.. Chính vì vậy mà ngày càng nhiều lập trình viên yêu thích với ngôn ngữ này hơn. Từ đó giúp các lập trình viên tập trung nhiều thời gian hơn vào những giải pháp hơn là cú pháp khi phát triển phần mềm bằng ngôn ngữ này.

Đối với những ứng dụng được viết bằng ngôn ngữ lập trình Python nhưng lại đòi hỏi sự phức tạp thì bạn có thể kết hợp các phần code của ngôn ngữ lập trình C, C++ vào phần code của Python. Như vậy sẽ giúp ứng dụng được tích hợp nhiều tính năng tốt hơn. Đây là một trong những tính năng nổi bật của Python mà các ngôn ngữ lập trình khác không thể làm được.

Khi lập trình ứng dụng bằng ngôn ngữ Python bạn không cần phải quản lý bộ nhớ hay dọn dẹp dữ liệu vô nghĩa,... Ngược lại, khi chạy code Python lên thì nó sẽ tự động chuyển đổi sang ngôn ngữ mà máy tính có thể hiểu được. Đây cũng là tính năng nổi bật của Python mà ngôn ngữ lập trình như C hoặc C++ không làm được.