**Python Modul 1 - BASIC**

Đặc điểm của Python

Python có ưu điểm:

* Cú pháp ngắn gọn, dễ hiểu, dễ đọc, phù hợp cho người mới học
* Hỗ trợ nhiều nền tảng, HĐH khác nhau
* Cộng đồng phát triển rất đông đảo (thư viện nhiều, tài liệu đầy đủ)

Bên cạnh đó Python có một số nhược điểm:

* Là **ngôn ngữ thông dịch** nên tốc độ thực thi chương trình Python chậm hơn so với các ngôn ngữ biên dịch
* Là **ngôn ngữ kiểu động** (dynamic type) nên không kiểm tra được hết các lỗi khi viết chương trình
* Python **không** phải là ngôn ngữ tốt trong xử lý **đa tác vụ** (Multi-threading)

## Biến / kiểu dữ liệu

* Kiểu dữ liệu số
* Kiểu dữ liệu String
* Kiểu dữ liệu List
* Kiểu dữ liệu Tuple
* Kiểu dữ liệu Set
* Kiểu dữ liệu Dictionary
* Kiểu dữ liệu Object

## Vào dữ liệu

## giatri = input('Nhập giá trị cho biến : ') // biến nhập vào kiểu String

## ép kiểu sang float/ int để phục vụ cho tính toán

## In ra màn hình

## print(<danh sách giá trị ngăn cách nhau bởi dấu phảy>)

## Cấu trúc điều khiển, cấu trúc lặp

### Cấu trúc điều khiển if

Các cấu trúc if trong Python:

* if <điều kiện> :

    <Lệnh>

* if <điều kiện> :

    <Lệnh 1>

else:

    <Lệnh 2>

* if <điều kiện 1> :

    <Lệnh 1>

elif <điều kiện 2>  :

    <Lệnh 2>

elif <điều kiện 3>  :

    <Lệnh 3>

else:

    <Lệnh 4>

Lưu ý:

* Cuối dòng các lệnh if/elif/else là dấu hai chấm. Python sử dụng dấu hai chấm để bắt đầu một khối lệnh con
* Các dòng của khối lệnh con bên dưới if/elif/else được viết lùi vào một số khoảng trắng (thường là 4 khoảng trắng hoặc 1 tab) so với dòng cha. Các dòng của một khối lệnh phải viết thẳng hàng với nhau.

***Điều kiện logic trong lệnh if***

Điều kiện logic trong lệnh if về bản chất là một biểu thức, nhưng chỉ nhận một trong hai giá trị True/False . Điều kiện này thường được tạo ra từ các phép toán :

* Các phép so sánh :
* Lớn hơn : >
* Nhỏ hơn : <
* Lớn hơn hoặc bằng : >=
* Nhỏ hơn hoặc bằng : <=
* Bằng nhau : ==
* Khác nhau : !=
* Phép phủ định : not
* Phép và : and
* Phép hoặc : or

### Cấu trúc lặp for

Cấu trúc lặp for được dùng để thực hiện một vòng lặp với số lần lặp biết trước. Cấu trúc lặp for như sau:

for bienLap in tapGiaTri:

    <khoi\_lenh>

## *Cách biểu diễn tập giá trị lặp trong vòng lặp for*

* Liệt kê các thành phần của tập giá trị. Ví dụ [1, 2, 3, 4, 5]. Đây thực chất là dữ liệu kiểu List (sẽ được mô tả ở phần sau)
* Sử dụng cấu trúc :

range(<end>)

Biểu thức này thể hiện tập các số nguyên từ 0 đến trước giá trị <end>,(tức <end-1>)

Ví dụ:

range(5) → 0, 1, 2, 3, 4

* Sử dụng cấu trúc :

range(start, end)

Biểu thức này thể hiện tập các số nguyên từ <start> đến <end-1>

Ví dụ:

range(1, 5) → 1, 2, 3, 4

* Sử dụng cấu trúc :

range(start, end, increment)

Biểu thức này thể hiện tập các số nguyên từ <start> đến <end-1> và tăng đều với khoảng cách increment.

Ví dụ:

range(0, 10, 2) → 0, 2, 4, 6, 8

## 

### Cấu trúc lặp while

Cấu trúc lặp while dùng để thực hiện một vòng lặp với số lần lặp không biết trước. Cấu trúc lặp while như sau:

while dieukien:

    <khoi\_lenh>

## Dữ liệu kiểu Number, String, Date&Time

Trong Python có 2 kiểu số : số nguyên (int) và số thực (float)

Số nguyên là các giá trị số không chứa dấu phảy:

x = 1

y = 2

z = -10000

Số thập phân là các số có chứa dấu phảy :

x = 1.0

y = 10.5

z = 9e38

Các phép tính trên kiểu dữ liệu số trong Python:

* Phép cộng (+)
* Phép trừ (-)
* Phép nhân (\*)
* Phép chia (/)
* Phép chia lấy phần nguyên (//)
* Phép lấy phần dư (%)
* Phép lũy thừa (\*\*)
* Phép lấy giá trị tuyệt đối **abs**
* Phép làm tròn số **round**

***Các hàm toán học:***

Các hàm toán học (trong thư viện math) cho phép thực hiện một số hàm lượng giác và giải tích trên số thập phân:

* Các hàm lượng giác : sin, cos, tan, asin, acos, atan
* Hàm căn bậc 2 : sqrt
* Hàm phần nguyên : floor, hàm phần nguyên trên : ceil
* Hàm logarith : log10 (cơ số 10), log ( cơ số e)

Để sử dụng các hàm toán học, cần khai báo thư viện math : import math

### Kiểu dữ liệu String

***Các hàm thông dụng trên kiểu dữ liệu String:***

* Hàm **str** : chuyển từ các kiểu dữ liệu khác sang dữ liệu string

Ví dụ:

>>> str(100)

'100'

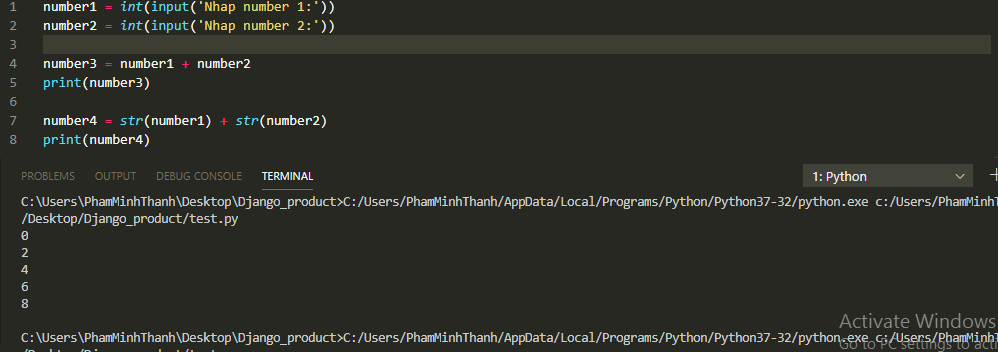
* Phép cộng 2 giá trị string :

Ví dụ:

>>> 'Chào ' + 'bạn'

'Chào bạn'

Code mẫu:



* Hàm **lower** : chuyển giá trị string sang dạng chữ thường

Ví dụ:

>>> 'Chào bạn'.lower()

'chào bạn'

* Hàm **upper** : chuyển giá trị string sang dạng chữ hoa

Ví dụ:

>>> 'Chào bạn'.upper()

'CHÀO BẠN'

* Hàm **replace** : Thay giá trị một đoạn con trong một string bằng một giá trị khác

Ví dụ:

>>> 'Tôi sống ở Hà Nội'.replace('Hà Nội', 'Huế')

'Tôi sống ở Huế'

* Hàm **split** : Tách một giá trị string thành nhiều đoạn nhỏ // tách chuỗi thành mảng các phần tử

Ví dụ:

>>> 'Hà Nội'.split()

['Hà', 'Nội']

>>> 'Hà Nội, Việt Nam'.split(',')

['Hà Nội', ' Việt Nam']

* Hàm **strip** : Bỏ đi các kí tự trống ở đầu và cuối string:

Ví dụ:

>>> '   Hà Nội    '.strip()

'Hà Nội'

* Hàm **len** : trả về số lượng ký tự của một giá trị string

Ví dụ:

>>> len('Chào bạn')

8

***Truy nhập đến từng kí tự của String:***

Một giá trị string được tạo thành từ một dãy các kí tự, để lấy kí tự ở vị trí index (bắt đầu từ 0) của một giá trị string, chúng ta dùng cú pháp:

c = text[index]

Bản thân một giá trị string có thể xem là một tập hợp kí tự và có thể dùng vòng lặp for chạy qua tập kí tự này

Ví dụ :

text = 'Chào bạn'

for c in text:

    print( c )

Một số phép tính trên kí tự:

* Hàm ord : lấy mã thứ tự của một kí tự trong bảng mã kí tự.

Ví dụ:

>>> ord('A')

65

* Hàm chr : chuyển ngược từ thứ tự của một kí tự trong bảng kí tự thành giá trị kí tự.

Ví dụ:

>>> chr(65)

'A'

***Substring:***

Mỗi đoạn nhỏ của một giá trị string được gọi là substring. Bản thân mỗi substring cũng là một giá trị kiểu string.

Các cách để lấy ra substring trong Python:

* Cú pháp:

text[start:end]

Biểu thức này trả về đoạn kí tự từ vị trí start đến end-1 của string gốc.

Ví dụ:

>>> text = 'Chào bạn'; print(text[0:4])

Chào

* Cú pháp:

text[start:]

Biểu thức này trả về đoạn kí tự từ vị trí start đến hết string gốc.

Ví dụ:

>>> text = 'Chào bạn'; print(text[5:])

bạn

* Cú pháp:

text[:end]

Biểu thức này trả về đoạn kí tự từ đầu đến vị trí end của string gốc.

Ví dụ:

>>> text = 'Chào bạn'; print(text[:4])

Chào

* Trong các cú pháp trên, nếu các giá trị start, end là âm thì có nghĩa vị trí được tính từ cuối string trở về (-1 tương ứng với vị trí cuối cùng của string)

Ví dụ:

>>> text = 'Chào bạn'; print(text[-3:])

bạn

***Format string***

Để ghép nhiều giá trị string thành một có thể dùng phép cộng string như đã trình bày ở phía trên:

|  |
| --- |
| x = 1 y = 2 z = 3 st = 'Tổng của ' + str(x) + ' và ' + str(y) + ' là ' + str(z) print(st) |

Tuy nhiên việc cộng nhiều string sẽ làm chương trình khó theo dõi, do đó có thể dùng hàm format của kiểu dữ liệu string để ghép nhiều string (và các kiểu dữ liệu khác) với nhau.Cách thức hiện như sau:

|  |
| --- |
| x = 1 y = 2 z = 3 st = 'Tổng của {} và {} là {}'.format(x, y, z) print(st) |

Mỗi cặp {} trong template của string sẽ tương ứng với một biến/biểu thức nằm bên trong hàm format.

Ngoài ra, có thể viết tên biến/biểu thức ngay bên trong cặp {} nếu sử dụng tiền tố **f** ở trước template của string, cách thực hiện như sau:

|  |
| --- |
| x = 1 y = 2 z = 3 st = f'Tổng của {x} và {y} là {z}' print(st) |

### Kiểu dữ liệu Date & Time

Dữ liệu kiểu Date & Time được cung cấp bởi thư viện datetime của hệ thống:

|  |
| --- |
| from datetime import date, datetime |

Kiểu dữ liệu date chứa thông tin về ngày, tháng, năm. Kiểu dữ liệu datetime chứa thông tin về ngày, tháng, năm, giờ, phút , giây, mili giây.

Để tạo mới một đối tượng kiểu date/datetime:

|  |
| --- |
| from datetime import date, datetime  d = date(2019,1,1)                  # 01/01/2019 dt = datetime(2019,1,1,23,59,59)    # 23:59:59 01/01/2019 |

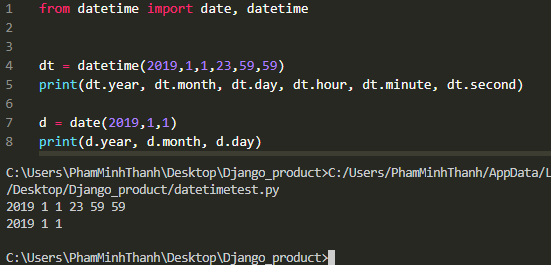
Lấy ngày giờ hiện tại của hệ thống:

|  |
| --- |
| from datetime import date, datetime  d = date.today() dt = datetime.now() |

Truy nhập đến các thành phần của date/datetime :

|  |
| --- |
| from datetime import date, datetime  dt = datetime(2019,1,1,23,59,59) print(dt.year, dt.month, dt.day, dt.hour, dt.minute, dt.second)  d = date(2019,1,1) print(d.year, d.month, d.day) |

// Không được đặt tên file trùng với tên thư viện



Chuyển đổi từ datetime sang String :

|  |
| --- |
| from datetime import datetime  now = datetime.now()  print(now.strftime('%d-%m-%Y')) print(now.strftime('%d/%m/%Y %H:%M:%S')) |

// Tác dụng của chuyển datetime sang String là để cộng chuỗi :> or something else

Chuyển đổi từ String sang datetime :

|  |
| --- |
| from datetime import datetime  dt1 = datetime.strptime('01/01/2019', '%d/%m/%Y') print(dt1)  dt2 = datetime.strptime('01-01-2019 23:59:59', '%d-%m-%Y %H:%M:%S') print(dt2) |

// Tác dụng của chuyển String sang datetime là để tính toán hàm thời gian ví dụ như hàm timedelta

***Khoảng thời gian :***

Để đo sự chênh lệch giữa các mốc thời gian, thư viện datetime cung cấp kiểu dữ liệu**timedelta**. Kiểu dữ liệu này có thể hiểu là một bộ gồm 2 thành phần là số ngày (days) và số giây (seconds)

|  |
| --- |
| from datetime import timedelta  duration = timedelta(days=30, seconds=3600) |

//Chuẩn hóa ra ngày và giờ

Tính khoảng thời gian giữa 2 thời điểm :

|  |
| --- |
| from datetime import datetime  dt1 = datetime(2019, 2, 1) dt2 = datetime(2019, 2, 28, 23, 59, 59)  duration = dt2 - dt1  days = duration.days hours = duration.seconds // 3600 minutes = (duration.seconds %  3600) // 60 seconds = duration.seconds % 60  print(f'Duration : {days} days, {hours} hours,  {minutes} minutes, {seconds} seconds') |

Cộng một mốc thời điểm với một khoảng thời gian :

|  |
| --- |
| from datetime import datetime, timedelta  dt1 = datetime(2019, 2, 1) duration = timedelta(days=30, seconds=3600) dt2 = dt1 + duration  print(dt2) |
|  |

## Project 1: Hoan thanh cac bai tap trong giao trinh:

## Hoan thanh 5 bai tap tong hop :

1. Viết chương trình tìm 2 số khi biết tổng và tỉ số. Nhập vào từ bàn phím các giá trị S : tổng 2 số, R : tỉ số của 2 số, in ra màn hình giá trị 2 số đó
2. Nhập vào từ bàn phím 3 giá trị ngày, tháng, năm của một ngày trong thế kỉ 21. Kiểm tra ngày đó có tồn tại không. Ví dụ:

Ngày 29/2/2000 :có tồn tại

Ngày 29/2/2001 :không tồn tại

Ngày 31/7/2010 :có tồn tại

Ngày 31/6/2016 :không tồn tại

1. Một người có 100 triệu VND và quyết định gửi tiết kiệm. Người đó gửi theo kỳ hạn từng 6 tháng một, kết thúc kỳ hạn số tiền lãi được chuyển nhập vào gốc để gửi cho kỳ hạn tiếp. Lãi suất cho một kỳ hạn là 3%. Viết chương trình để in ra số tiền của người đó sau các kỳ hạn từ 1 đến 10.
2. Một người mua một căn hộ nhưng không đủ tiền nên vay ngân hàng một số tiền là 400 triệu VND. Lãi suất ngân hàng là 0% trong năm đầu tiên và 10% từ năm thứ 2 trở đi. Nếu mỗi tháng người đó trả ngân hàng 10 triệu thì khi sau bao lâu người đó thanh toán hết khoản nợ.
3. Chuyển một số trong phạm vi 0-99 thành phát âm tiếng Việt. Ví dụ : 85 → tám mươi lăm

## Project 2: Múi giờ

## In ra giờ hiện tại

## In ra giờ hiện tại ở LonDon,NewYork, Tokyo, Bắc Kinh…

## Các kiểu dữ liệu List, Tuple, Set, Dictionary

## List:

dayso = [1, 3, 5, 7, 9]  
danhsach\_hocsinh = ["Nguyễn Văn An", "Nguyễn Chí Cường", "Nguyễn Mạnh Tuấn"]  
diachi = [322, "Tây Sơn", "Hà Nội"]

## len – chiều dài list

## append – thêm phần tử vào cuối list

## remove - Xóa một phần tử có sẵn trong list

## extend - ghép một list con vào cuối một list

## += - cộng 2 list bằng toán tử “+”

## reverse – Đảo ngược một List

## in – kiểm tra phần tử nằm trong List

## not in – kiểm tra một phần tử không nằm trong List

## index - trả về vị trí của một phần tử trong List

## min – trả về giá trị nhỏ nhất

## max – trả về giá trị lớn nhất

## *Truy nhập đến từng phần tử của List*

## *Sắp xếp một List*

## sorted(lst[], key = key\_function, reverse = true/false)

>>> print(sorted([1, 3, 2, 5, 4]))

[1, 2, 3, 4, 5]

>>> print(sorted([1, 3, 2, 5, 4], reverse=True))

[5, 4, 3, 2, 1]

### Kiểu dữ liệu Tuple

Tuple khác List một điểm là Tuple không thể thêm sửa xóa, bị fix cứng rồi

dayso = (1, 3, 5, 7, 9)  
danhsach\_hocsinh = ("Nguyễn Văn An", "Nguyễn Chí Cường", "Nguyễn Mạnh Tuấn")

Các phép tính trên tuple cũng giống List

* Hàm **len** : trả về số phần tử của một Tuple
* Phép cộng 2 Tuple : trả về một Tuple mới bằng cách ghép 2 Tuple với nhau
* Phép toán **in** : Kiểm tra một phần tử có nằm trong Tuple
* Phép toán **not in** : Kiểm tra một phần tử có không nằm trong Tuple
* Hàm **index** : Tìm vị trí của một phần tử trong Tuple
* Hàm **min**: Trả về giá trị nhỏ nhất của một Tuple
* Hàm **max**: Trả về giá trị lớn nhất của một Tuple
* Truy nhập phần tử qua chỉ số: tương tự như List

## Kiểu dữ liệu Set:

Kiểu dữ liệu Set dùng để chứa các phần tử của một tập hợp. So với kiểu List, kiểu Set có điểm khác là:

* Trong một Set **không có 2 phần tử cùng giá trị.**
* Không thể dùng index để truy nhập phần tử của một Set

## Các phép tính:

## add – thêm giá trị mới vào Set

## remove – xóa phần tử ra khỏi Set

## in – kiểm tra giá trị có nằm trong Set hay không

## not in – Kiểm tra giá trị có không nằm trong Set không

## union - Hợp 2 set

## intersection – Giao của 2 Set

## difference – Hiệu 2 set

### Kiểu dữ liệu Dictionary

Kiểu dữ liệu Dictionary dùng để chứa một bảng biến đổi 1-1 giữa 2 tập hợp :

* Tập nguồn (hay tập khóa (key)) : chứa các thông tin cần tra cứu
* Tập đích (hay tập giá trị (value)) : chứa thông tin của mỗi key từ tập nguồn

Cú pháp khai báo kiểu Dictionary trong Python:

d = {

    <key1> : <value1>,

    <key2> : <value2>,

    ...

}

Ví dụ:

d = {"một" : 1, "hai" : 2, "ba" : 3}

## Để truy cập phần tử thì dùng d[<key>]

## Trong trường hợp nếu giá trị <key> không nằm trong tập nguồn của Dictionary, Python sẽ đưa ra thông báo Exception.

## Để chống lỗi này ta có thể thêm vào một giá trị mặc định

d.get(<key>, <gia\_tri\_mac\_dinh>)

Ví dụ:

>>> d = {"một" : 1, "hai" : 2, "ba" : 3}; print(d.get('bốn', -1))

-1

## Project 3: Đếm tần suất của các từ trong một câu

## 'Một năm có mười hai tháng, tháng hai có hai mươi tám ngày, các tháng còn lại có ba mươi hoặc ba mươi mốt ngày'

## Project 4: Chuyển phát âm tiếng Việt của một số trong phạm vi 1-99 thành giá trị số đó. Ví dụ : tám mươi lăm → 85

## Hàm (Function)

***Hàm trả về nhiều giá trị***

## Một hàm có thể trả về nhiều giá trị, danh sách các giá trị trả về ở được ngăn cách nhau bởi dấu phảy

def calcAreaAndPerimeter(width, height):  
   S = width \* height  
   P = 2\*(width + height)  
   return S, P  
  
A, P = calcAreaAndPerimeter(5, 4)  
print('A = ', A)  
print('P = ', P)

## Module & làm việc với thư viện

from <module> import <functions>

## File I/O

## *Mở file*

## f = open(<file\_name>, <mode>, encoding=<encoding>)

Trong đó :

* <file\_name> : tên file cần mở
* <mode> : Chế độ mở file, các giá trị thường dùng:
* 'r' : Mở file để đọc, đây là giá trị mặc định
* 'rb' : Mở file để đọc dưới dạng nhị phân (binary)
* 'w' : Tạo file mới để ghi
* 'wb' : Tạo file mới để ghi dưới dạng nhị phân (binary)
* 'a' :  Mở file để ghi thêm vào cuối (append)
* 'ab' : Mở file để ghi thêm vào cuối dưới dạng nhị phân
* <encoding>: Chế độ encoding với file văn bản
* None : giá trị mặc định, tùy thuộc từng hệ thống
* 'utf-8' : mã hóa Unicode dạng UTF-8

***Đọc file***

* Đọc toàn bộ nội dung file:

data = f.read()

Nếu file được mở theo chế độ văn bản ('r'), kết quả đọc file là một **biến** string. Nếu file được mở theo chế độ nhị phân ('rb'), kết quả đọc file là một biến có kiểu dữ liệu **bytes.**

* Đọc toàn bộ các dòng của file (với file văn bản):
* Nếu mở file theo chế độ văn bản ('r') có thể đọc nội dung file theo từng dòng với lệnh :

|  |
| --- |
| lines = f.readlines() |

Kết quả trả về là một List, mỗi phần tử là một string chứa nội dung các dòng của file

* Đọc từng dòng của file:

Nếu mở file theo chế độ văn bản ('r') có thể đọc từng dòng của file với cấu trúc:

|  |
| --- |
| for line in f:    # process line |

## Cái này đọc trực tiếp từng dòng sau khi mở file (f) chứ không phải là

## lines = f.readlines() đâu nhé

## Ghi file:

* Với file văn bản, dữ liệu ghi vào file cần có kiếu string

f.write(<string>)

Ví dụ:

|  |
| --- |
| f = open('test.txt', 'w') f.write('hello') f.close() |

* Với file nhị phân, dữ liệu ghi vào file cần có kiểu bytes

f.write(<bytes>)

Ví dụ:

|  |
| --- |
| f = open('test.dat', 'wb') f.write('hello'.encode()) f.close() |

***Đóng file***

|  |
| --- |
| f.close() |

***Truy xuất file với cấu trúc 'with'***

Sau khi mở file với lệnh open, phải đóng file với lệnh close để tránh bị thiếu nội dung file khi kết thúc chương trình. Để tránh việc quên đóng file sau khi mở, có thể dùng cấu trúc **with**:

with open(<file\_name>, <mode>, <encoding>) as f:

# Process file

## Với cấu trúc with, file sẽ tự động đươc đóng khi chương trình chạy xong khối lệnh bên trong with.

Ví dụ:

|  |
| --- |
| with open('test.txt', 'w') as f:    f.write('Line1\n')    f.write('Line2\n') |

**Project 5:** Đọc vào một file văn bản, tạo ra một file văn bản mới chứa các dòng của file nguồn, bỏ đi các dòng trống.

File input.txt:

Một năm có 365 hoặc 366 ngày

Năm thường có 365 ngày

Năm nhuận có 366 ngày

File output.txt:

Một năm có 365 hoặc 366 ngày

Năm thường có 365 ngày

Năm nhuận có 366 ngày

**Project 6:** Một file csv chứa bảng điểm một môn học của các học sinh một lớp học. Mỗi dòng của file là thông tin điểm của một học sinh bao gồm : họ tên, điểm hệ số 1, điểm hệ số 2, điểm hệ số 3, các giá trị này ngăn cách nhau bởi dấu phảy. Viết chương trình để đọc file csv đầu vào và tạo ra một file csv mới có thêm cột điểm trung bình.

File input.csv

Nguyễn Văn An, 8, 7, 8  
Nguyễn Văn Bình, 6, 6, 8  
Nguyễn Thị Chi, 8, 8, 9  
Lê Văn Cường, 8, 7, 9  
Phạm Thu Trang, 7, 8, 8

File output.csv

Nguyễn Văn An, 8, 7, 8,  7.7

Nguyễn Văn Bình, 6, 6, 8,  7.0

Nguyễn Thị Chi, 8, 8, 9,  8.5

Lê Văn Cường, 8, 7, 9,  8.2

Phạm Thu Trang, 7, 8, 8,  7.8

**Project 7 :** Một file csv chứa danh sách các mặt hàng đã được bán ra của một cửa hàng trong ngày, mỗi dòng là thông tin về một lần khách mua hàng, bao gồm mã hàng hóa, số lượng hàng hóa bán ra, giá bán ra, giá gốc nhập về. Các dòng khác nhau có thể có cùng mã hàng hóa (nhiều người mua cùng 1 mặt hàng). Viết chương trình tạo ra file csv trong đó mỗi dòng chứa thông tin về tình trạng kinh doanh của một mặt hàng, gồm các cột : mã hàng hóa (chỉ tính một lần duy nhất cho các lần mua khác nhau của cùng một mặt hàng), tổng số lượng đã bán, tổng doanh số (số tiền thu được do bán hàng), tổng lợi nhuận (số tiền thu được sau khi trừ đi giá gốc nhập về). Các mặt hàng xếp theo trình tự giảm dần của doanh số. Ví dụ :

File input.csv:

VINAMILK01, 5, 27000, 20000

THMILK03, 10, 28000, 20000

OMACHI05, 2, 6000, 4000

VINAMILK01, 5, 27000, 20000

VINAMILK01, 10, 27000, 20000

OMACHI05, 5, 6000, 4000

File output.csv:

VINAMILK01, 20, 540000, 140000

THMILK03, 10, 280000, 80000

OMACHI05, 7, 42000, 14000