Lập trình cho người mới học

Giới thiệu

Lập trình là gì?

Một cách dễ hiểu, lập trình là cách giải các bài toán bằng cách viết chương trình trên máy tính.

Khi chưa có máy tính, con người vẫn giải các bài toán gặp trong cuộc sống. Các bài toán đơn giản có thể tính nhẩm trong đầu, các bài toán phức tạp hơn cần đến giấy và bút.

*Ví dụ :*

Một người đi làm bằng xe máy, quãng đường từ nhà đến nơi làm việc là 10 km. Trung bình cứ 50km thì xe máy tiêu thụ hết 1 lít xăng. Giá xăng là 21000 đồng/lít. Một tháng người đó phải đi làm 25 ngày. Hỏi tiền xăng cho việc đi lại trong 1 tháng của người đó là bao nhiêu?

*Lời giải:*

Quãng đường đi lại trong 1 ngày:

           2 x 10 = 20 (km)

Quãng đường đi lại trong 1 tháng:

          20 x 25 = 500 (km)

Số lít xăng tiêu thụ trong 1 tháng:

           500/50 = 10 (lít)

Tiền xăng tiêu thụ trong 1 tháng:

           10 x 21000 = 210000 (đồng)

*Đáp số : 210000 đồng*

Kể từ khi có máy tính, con người đã sử dụng máy trong việc giải các bài toán. Việc giải toán trên máy tính nhanh hơn và chính xác hơn so với việc giải bằng tay.

Ngôn ngữ lập trình Python

Có nhiều ngôn ngữ lập trình có thể sử dụng để giải các bài toán trên máy tính, trong đó Python là ngôn ngữ đơn giản nhất và phù hợp cho người mới bắt đầu học lập trình

Kể cả chưa từng sử dụng lập trình bao giờ, bạn vẫn có thể hiểu được (một phần) chương trình viết trên Python đang thực hiện gì do cách thể hiện chương trình gần giống với ngôn ngữ của con người.

Hãy xem xét chương trình viết trên Python để giải bài toán trong ví dụ phía trên:

khoangcach = 10

quangduong\_1ngay = 2 \* khoangcach

quangduong\_1thang = 25 \* quangduong\_1ngay

so\_litxang\_1thang = quangduong\_1thang / 50

tienxang\_1thang = so\_litxang\_1thang \* 21000

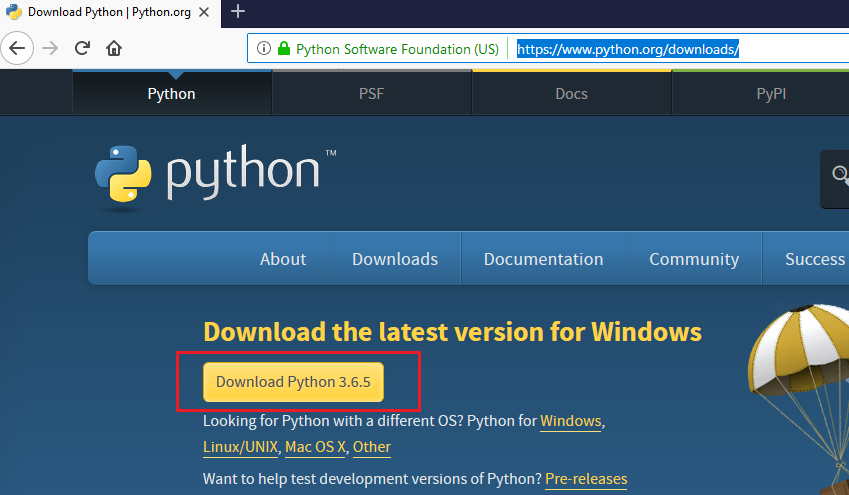
print("Tiền xăng tiêu thụ trong 1 tháng là ", tienxang\_1thang)

Bạn sẽ thấy chương trình gồm một số dòng lệnh, mỗi lệnh tương ứng với một phép tính trong lời giải trên giấy mà chúng ta đã xem ở phần trên.

Cài đặt Python

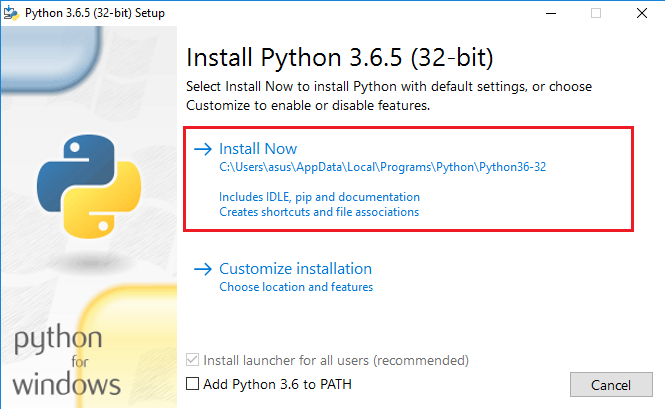
Để cài đặt Python, bạn hãy tải file cài đặt tại trang chủ của Python:

[https://www.python.org/downloads/](https://www.google.com/url?q=https://www.python.org/downloads/&sa=D&ust=1551413975548000)



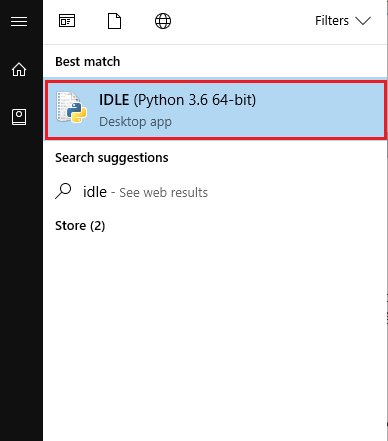
Tải Python từ trang chủ (chọn phiên bản mới nhất)

Sau khi tải file cài đặt về, bạn hãy chạy file này và chọn các thiết lập cài đặt mặc định, cho đến khi chương trình cài đặt hoàn thành:



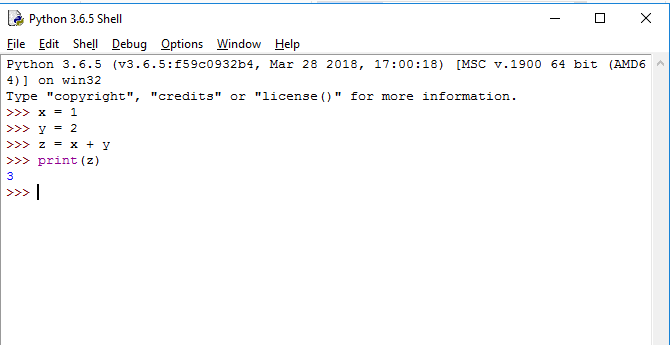
Cài đặt Python với các option mặc định

Sau khi đã cài đặt Python xong, bạn hãy tìm chương trình IDLE (Integrated DeveLopment Environment) của Python từ Start Menu và khởi động chương trình này :



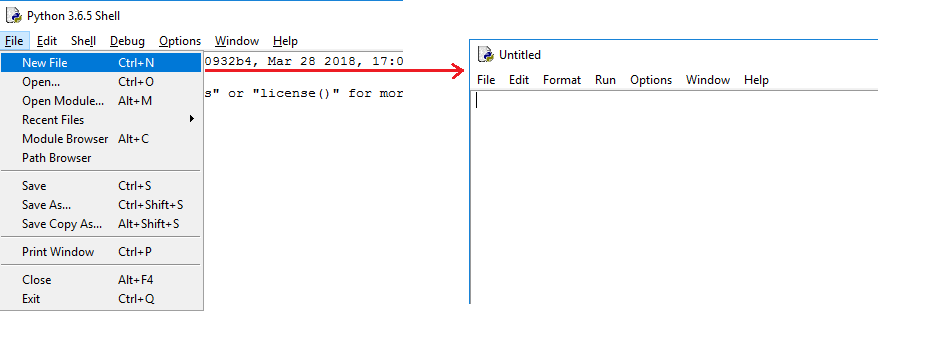
Khởi động chương trình soạn thảo Python IDLE

Sau khi khởi động xong, IDLE sẽ cho phép bạn chạy từng dòng lệnh Python bằng cách gõ trực tiếp vào cửa sổ chương trình:



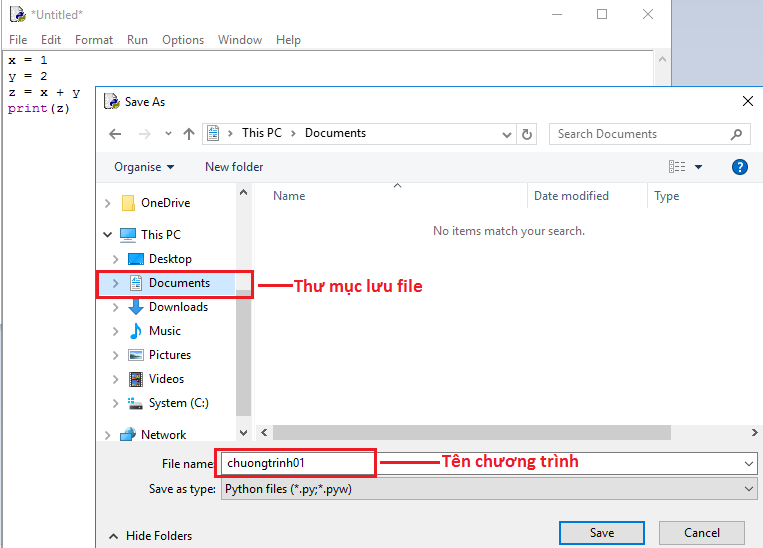
Chạy Python ở chế độ từng lệnh trong cửa sổ IDLE

Với các chương trình lớn, việc chạy từng lệnh sẽ không phù hợp, bạn có thể tạo ra một file với tên mở rộng **py** để lưu chương trình của mình. Cách thực hiện như sau : từ menu **File** của IDLE chọn **New File,**một cửa sổ mới sẽ được tạo ra cho phép bạn soạn thảo chương trình:



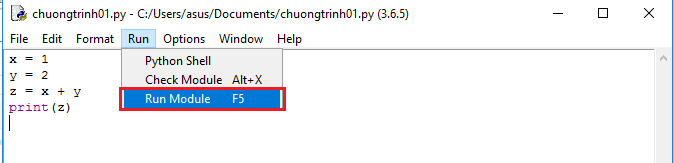
Tạo file mới để soạn thảo

Sau khi soạn thảo xong ở cửa sổ mới, bạn cần lưu lại file này : Từ menu **File**hãy chọn **Save**, cửa sổ lưu file sẽ hiện ra yêu cầu bạn đặt tên cho file và chọn thư mục để chứa file. Bạn hãy nhớ địa chỉ lưu trữ này để tìm đến trong các lần mở file tiếp theo:



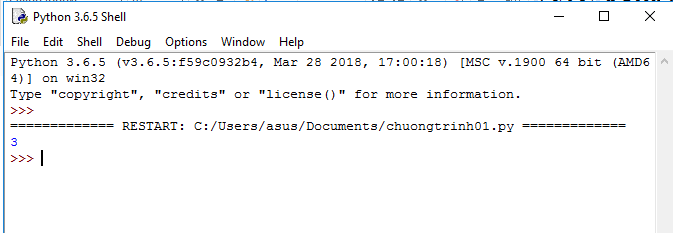
Lưu lại chương trình đã soạn thảo xong

Sau khi đã lưu chương trình, bạn có thể thực hiện chương trình bằng cách từ menu **Run** chọn **Run Module*,***hoặc nhấn F5 :



Chạy chương trình đã lưu.

Kết quả chương trình sẽ được hiển thị trong cửa sổ dòng lệnh IDLE :



Kết quả chương trình được hiển thị trong cửa sổ IDLE

***Chú thích trong chương trình:***

Khi viết chương trình, bạn có thể thêm các chú thích để giải thích ý nghĩa của các dòng lệnh. Trong Python, các chú thích được đặt sau dấu **#**, tức toàn bộ phần sau dấu # đến hết cuối dòng sẽ được Python bỏ qua khi chạy chương trình. Ví dụ :

# Chương trình tính tổng 2 số và in ra màn hình

x = 1

y = 2

z = x + y

print(z)         # In kết quả ra màn hình

Một số kiến thức lập trình cơ bản với Python

Biến số

Biến số là khái niệm cơ bản nhất trong các chương trình. Chúng được dùng để lưu giá trị trung gian trong quá trình tính toán.  Cú pháp để khai báo một biến số:

bienSo = <giá trị>

Trong đó bienSo là tên của biến số cần khai báo, <giá trị> là giá trị ban đầu chúng ta muốn đặt cho biến số. Giá trị này có thể là một hằng số hoặc một biểu thức của các biến số đã được khai báo.

Sau khi được khai báo, biến số sẽ được lưu trong bộ nhớ máy tính. Giá trị của biến số có thể thay đổi trong chương trình:

x = 1

x = 2

x = x + 1

Ở chương trình trên:

* Dòng đầu tiên khai báo biến x và đặt giá trị cho biến bằng 1
* Dòng thứ 2 thay đổi giá trị của x thành 2
* Dòng thứ 3 tăng x thêm 1 đơn vị

Để xóa một biến số khỏi bộ nhớ, chúng ta dùng lệnh:

del bienSo

Kiểu dữ liệu

Mỗi biến số sau khi được khai báo sẽ chứa giá trị nhất định, gọi là dữ liệu. Giá trị dữ liệu này sẽ thuộc vào một trong các kiểu dữ liệu mà Python hỗ trợ.

Khi giải toán trên giấy, chúng ta thường chỉ quan tâm tới dữ liệu kiểu số. Trên máy tính, ngoài kiểu dữ liệu số, còn có dữ liệu văn bản (String). Loại dữ liệu này dùng để lưu các biến số như tên, địa chỉ, số điện thoại …

Ngoài ra trong Python, còn có các kiểu dữ liệu mà chúng ta sẽ biết đến ở phần sau:

* Kiểu danh sách (List)
* Kiểu bộ nhóm (Tuple)
* Kiểu tập hợp (Set)
* Kiểu từ điển (Dictionary)
* Kiểu tệp tin (File)

Chúng ta tạm quan tâm đến 2 kiểu dữ liệu chính là kiểu số và kiểu văn bản.

Kiểu dữ liệu số

Đây là kiểu dữ liệu sử dụng nhiều nhất. Chúng ta đã biết kiểu dữ liệu này qua ví dụ ở phần giới thiệu.

Trong Python có 2 loại số : số nguyên và số thập phân.

Số nguyên

Số nguyên là các số không chứa dấu phảy :

x = 1

y = 2

z = 100

Số thập phân

Số thập phân là số chứa dấu phảy:

x = 1.2

y = 0.0001

z = 1e6

Cách viết z = 1e6 có nghĩa là 1 x 106

Kiểu dữ liệu văn bản (String)

Dữ liệu kiểu String được đặt trong 2 dấu nháy đơn (') hoặc 2 dấu nháy kép (")

hoten = 'Nguyễn Văn An'

diachi = "Hà Nội"

Chúng ta sẽ tìm hiểu chi tiết hơn về kiểu dữ liệu này trong chương *“Dữ liệu kiểu văn bản”* ở phần sau

Vào ra dữ liệu

Lấy dữ liệu vào từ bàn phím

Giá trị của các biến số có thể được khai báo trực tiếp trong chương trình, tuy nhiên, trong nhiều trường hợp, chúng ta muốn sử dụng lại chương trình đã viết với các bộ dữ liệu vào khác, trong trường hợp đó cần cho phép người sử dụng nhập dữ liệu vào cho các biến từ bàn phím.

Trong Python, việc này được thực hiện qua lệnh input :

giatri = input('Nhập giá trị cho biến : ')

Lệnh trên sẽ đọc các kí tự nhập từ bàn phím cho đến khi gặp kí tự Enter, giá trị nhập vào được lưu vào biến khai báo ở đầu dòng lệnh, giá trị này có kiểu dữ liệu là văn bản (String).

*Ví dụ:*

Viết chương trình yêu cầu người sử dụng giới thiệu tên, sau đó in ra lời chào theo tên đã nhập vào.

ten = input('Mời bạn cho biết tên của bạn : ')

print('Xin chào ', ten)

Trong trường hợp muốn lấy giá trị vào ở dạng số, chúng ta sử dụng các lệnh sau để chuyển đổi từ dữ liệu String sang dữ liệu số:

* int(text) : chuyển giá trị text có kiểu String thành số nguyên
* float(text) : chuyển giá trị text có kiểu String thành số thập phân

*Ví dụ:*

Viết chương trình nhập vào từ bàn phím 2 số và in ra tổng của 2 số đó.

a = input('Số thứ nhất : ')

a = float(a)

b = input('Số thứ hai : ')

b = float(b)

print('Tổng của 2 số là : ', a+b)

In dữ liệu ra màn hình

Sau khi tính toán xong một bài toán, chúng ta cần in kết quả ra màn hình. Để thực hiện việc này chúng ta dùng lệnh **print** theo cú pháp:

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

*Ví dụ:*

z = 1

y = 2

z = x + y

print(x, '+', y, '=', z)

Ở dòng cuối trong chương trình trên, danh sách các giá trị được in ra nằm trong lệnh print, theo thứ tự là:

x   →  1

'+' →  +

y   →  2

'=' →  =

z   →  3

Như vậy, thông tin được in ra trên màn hình là:

1 + 2 = 3

Các phép tính

Python hỗ trợ các phép tính toán học thông dụng. Các phép tính thường được sử dụng là:

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

Các phép tính +, -, \*, / tương tự như các phép tính số học chúng ta đã biết.

*Ví dụ:*

>>> print(1 + 2)

3

>>> print(3 - 2)

1

>>> print(3 \* 4)

12

>>> print(1/2)

0.5

Phép chia lấy phần nguyên và phép lấy phần dư

Phép chia lấy phần nguyên (//) và phép lấy phần dư (%) được thực hiện để chia các số nguyên.

*Ví dụ:*

>>> print(10 // 3)

3

>>> print(10 % 3)

1

Phép lũy thừa

Trong Python, phép lũy thừa được thể hiện bằng 2 dấu \* viết liền nhau.

*Ví dụ:*

>>> print(2 \*\* 10)

1024

Phép tính dạng rút gọn

Python cho phép sử dụng các phép tính dạng rút gọn:  +=,  -=,  \*=,  /=,  //=,  %=,  \*\*=

Ý nghĩa của phép tính rút gón như sau:

* a += b tương đương với a = a + b
* a -= b tương đương với a = a - b
* …

Phép làm tròn số thập phân

Hàm **round**  được sử dụng để làm tròn số thập phân

*Ví dụ:*

>>> print(round(1.2))

1

Hàm **round**còn có thể dùng để làm tròn theo một số lượng nhất định chữ số sau dấu phảy

*Ví dụ:*

>>> print(round(4/3, 2))

1.33

Phép lấy giá trị tuyệt đối

Hàm **abs** được dùng để lấy giá trị tuyệt đối của một số

*Ví dụ:*

>>> print(abs(2))

2

>>> print(abs(-1))

1

*Ví dụ:*

Tìm 2 số biết tổng và hiệu của chúng. Nhập vào từ bàn phím các giá trị S : tổng 2 số, D : hiệu 2 số , in ra màn hình giá trị 2 số đó.

S = input('S = ')

S = float(S)

D = input('D = ')

D = float(D)

a = (S - D)/2

b = (S + D)/2

print('a = ', a)

print('b = ', b)

***Luyện tập:***

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

Lệnh điều khiển 'if'

Khi giải các bài toán trên giấy, thứ tự thực hiện các phép tính là từ trên xuống dưới. Tuy nhiên, với các bài toán trên máy tính, có hiện tượng "rẽ nhánh", tức là tùy vào kết quả của phép tính trước mà một số phép tính sau có thể được thực hiện hay không.

Để thực hiện việc này, Python sử dụng lệnh **if** theo các cấu trúc:

* Cấu trúc 1:

if <điều kiện> :

<Lệnh>

Cấu trúc trên có nghĩa : Nếu điều kiện đúng thì thực hiện lệnh, không thì không thực hiện lệnh.

* Cấu trúc 2:

if <điều kiện> :

    <Lệnh 1>

else:

<Lệnh 2>

Cấu trúc trên có nghĩa : Nếu điều kiện đúng thì thực hiện lệnh 1, không thì thực hiện lệnh 2.

* Cấu trúc 3:

if <điều kiện 1> :

    <Lệnh 1>

elif <điều kiện 2>  :

    <Lệnh 2>

elif <điều kiện 3>  :

    <Lệnh 3>

else:

    <Lệnh 4>

Cấu trúc trên có nghĩa : Nếu điều kiện 1 đúng thì thực hiện lệnh 1, không thì kiểm tra điều kiện 2, nếu điều kiện 2 đúng thì thực hiện lệnh 2, nếu không tiếp tục kiểm tra các điều kiện tiếp theo.

Điều kiện trong câu lệnh **if** là điều kiện logic, chỉ có giá trị đúng (**True**) hoặc sai (**False**). Để tạo ra các điều kiện này, chúng ta sử dụng các phép so sánh và các phép tính logic:

* Các phép so sánh :
* **> :**lớn hơn
* **< :**nhỏ hơn
* **>= :**lớn hơn hoặc bằng
* **<= :**nhỏ hơn hoặc bằng
* **== :**bằng nhau
* **!= :**khác nhau
* Phép phủ định **not** :

 "**not dieukien**" có giá trị **True** nếu **dieukien** là **False** và ngược lại

* Phép **and** :

 "**dieukien1 and dieukien2**" có giá trị **True** nếu cả 2 điều kiện thành phần có giá trị **True**

* Phép **or** :

"**dieukien1 or dieukien2**" có giá trị **True** nếu ít nhất 1 trong 2 điều kiện thành phần có giá trị **True**

Một số lưu ý:

* Cuối các dòng của lệnh if hoặc else là dấu hai chấm (:). Python sử dụng dấu hai chấm để bắt đầu cho một khối lệnh con
* Các dòng lệnh bên dưới if hoặc else cần được viết lùi vào một số khoảng trắng (thường là 4 khoảng trắng, hoặc 1 tab) so với dòng chứa lệnh if/else, nếu có nhiều hơn một dòng lệnh thì các dòng này phải được viết thẳng hàng.

*Ví dụ 1:*

Nhập vào từ bàn phím 2 số và in ra giá trị nhỏ nhất của 2 số đó

a = input('Số thứ nhất : ')

a = float(a)

b = input('Số thứ hai : ')

b = float(b)

if a < b:

   print(a)

else:

   print(b)

*Ví dụ 2:*

Chỉ số BMI (Body mass index) dùng để đánh giá thân hình của một người. Chỉ số này được đo bằng tỉ số giữa cân nặng (tính theo kg) và bình phương của chiều cao (tính theo mét). Dựa vào chỉ số BMI, người ta có thể đánh giá được thân hình của một người:

* BMI nhỏ hơn 15 : thân hình quá gầy
* BMI từ 15 đến 16 : thân hình gầy
* BMI từ 16 đến 18.5 : thân hình hơi gầy
* BMI từ 18.5 đến 25 : thân hình bình thường
* BMI từ 25 đến 30 : thân hình hơi béo
* BMI từ 30 đến 35: thân hình béo
* BMI trên 35: thân hình quá béo

Nhập vào giá trị chiều cao và cân nặng của một người từ bàn phím và cho biết tình trạng thân hình của người đó.

height = input('Chiều cao (mét) : ')

height = float(height)

mass = input('Cân nặng (kg) : ')

mass = float(mass)

bmi = mass / (height \* height)

if bmi < 15:

    print('Thân hình quá gầy')

elif bmi < 16:

    print('Thân hình gầy')

elif bmi < 18.5:

    print('Thân hình hơi gầy')

elif bmi < 25:

    print('Thân hình bình thường')

elif bmi < 30:

    print('Thân hình hơi béo')

elif bmi < 35:

    print('Thân hình béo')

else:

    print('Thân hình quá béo')

***Luyện tập***

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

Lệnh lặp for

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

for bienLap in tapGiaTri:

    <khoi\_lenh>

Cấu trúc này có nghĩa : với mỗi giá trị bienLap của tapGiaTri, thực hiện khối lệnh <khoi\_lenh>

Các thành phần của vòng lặp for:

* Tập giá trị : chứa tập giá trị cần duyệt qua
* Biến lặp : biến được gán giá trị từ mỗi phần tử trong tập giá trị lặp
* Khối lênh lặp : khối lệnh cần thực hiện lặp lại

Các cách biểu diễn tập giá trị lặp:

* Liệt kê các thành phần. Ví dụ [1, 2, 3, 4, 5]. Đây thực chất là dữ liệu kiểu danh sách mà chúng ta sẽ biết ở các phần sau.
* Cấu trúc :

**range(end)**

bao gồm các số tự nhiên từ 0 đến trước **end**. Lưu ý, với Python (và nhiều ngôn ngữ lập trình), khoảng này không chứa giá trị **end**, tức giá trị cuối của khoảng là **end-1**

Ví dụ:

range(5) → 0, 1, 2, 3, 4

* Cấu trúc :

**range(start, end)**

bao gồm các số nguyên bắt đầu từ **start** đến trước **end**.

Ví dụ:

range(1, 5) → 1, 2, 3, 4

* Cấu trúc

**range(start, end, increment)**

bao gồm các số nguyên từ **start** đến trước **end** và tăng đều theo khoảng cách **increment**

Ví dụ:

range(0, 10, 2) → 0, 2, 4, 6, 8

*Ví dụ 1:*

Tính tổng các số nguyên từ 1 đến 100

S = 0

for i in range(1, 101):

    S += i

print('S = ', S)

*Ví dụ 2:*

Viết chương trình in ra bảng cửu chương.

for i in range(2, 10):

    for j in range(2, 10):

        print(i, '\*', j, '=', i\*j)

    print()

***Luyện tập***

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.

Lệnh lặp while

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

while dieukien:

    <khoi\_lenh>

Cấu trúc trên có nghĩa : Chừng nào điều kiện dieukien còn đúng thì thực hiện khối lệnh <khoi\_lenh>.

*Ví dụ 1:*

Dân số Việt Nam năm 2017 là 95.5 triệu người. Nếu tốc độ tăng dân số trung bình năm là 1.2% thì đến năm bao nhiêu dân số Việt Nam là 150 triệu

danso = 95.6

nam = 2017

tocdo\_tang = 1.2/100

while danso < 150 :

    nam += 1

    danso \*= (1 + tocdo\_tang)

print('Năm dân số đạt 150 triệu : ', nam)

Kết quả đưa ra dự đoán dân số Việt Nam sẽ đạt 150 triệu vào năm 2055

*Ví dụ 2:*

Chuyển một số từ hệ thập phân sang hệ nhị phân.

Để chuyển một số từ hệ thập phân sang hệ nhị phân chúng ta chia liên tiếp số đó cho 2 cho đến khi kết quả bằng 0. Các số dư được viết từ phải qua trái sẽ tạo ra biểu diễn nhị phân của số đó

x = int(input('x='))

s = ''

while x > 0:

    i = x % 2

    s = str(i) + s

    x = int(x/2)

print(s)

***Luyện tập***

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.

Hàm giá trị

Hàm (function) là một đoạn chương trình được tổ chức thành một khối độc lập để có thể được sử dụng lại nhiều lần từ các vị trí khác nhau trong chương trình.

Cách khai báo một hàm như sau:

def tenham(<danh\_sach\_bien>):

    <noi\_dung\_ham>

    return <ket\_qua>.

Các thành phần của hàm:

* Khai báo hàm : bắt đầu bằng từ khóa **def** của Python, tiếp theo là tên hàm, tiếp theo là danh sách các biến nằm trong 2 dấu (), các biến ngăn cách nhau bởi dấu phảy.
* Nội dung hàm : bao gồm các lệnh thực hiện của hàm. Các dòng lệnh này phải được viết lùi vào so với dòng khai báo hàm một số (thường là 4) khoảng trắng.

* Kết quả trả về của hàm : kết quả được trả về với từ khóa **return** đi kèm các giá trị trả về.

Trong chương trình, việc gọi hàm được thực hiện nhờ cú pháp sau:

ketqua = tenham(<danh\_sach\_bien\_dau\_vao>)

Trong trường hợp không quan tâm đến kết quả trả về của hàm, chúng ta dùng cú pháp:

tenham(<danh\_sach\_bien\_dau\_vao>)

*Ví dụ :*

Xây dựng hàm tính bình phương của một số, sau đó sử dụng hàm này để in ra các số chính phương từ 1 đến 100

def square(x):

   return x\*x

for i in range(1,11):

   print(square(i))

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.

*Ví dụ:*

Viết hàm tính diện tích và chu vi của một hình chữ nhật khi biết độ dài 2 cạnh.

def calcAreaAndPerimeter(width, height):

    S = width \* height

    P = 2\*(width + height)

    return S, P

S, P = calcAreaAndPerimeter(5, 4)

print('S = ', S)

print('P = ', P)

Đệ quy

Đệ quy là cách lập trình trong đó trong một hàm có lệnh gọi đến chính hàm đó, tuy nhiên tham số truyền vào thường sẽ nhỏ hơn so với lần gọi trước. Có thể hiểu đệ quy tìm cách giải một bài toán lớn bằng cách đưa về bài toán tương tự nhưng ở quy mô nhỏ hơn.

*Ví dụ 1:*

Viết chương trình tính hàm giai thừa của một số tự nhiên :  n! = 1 x 2 x 3 x .... x n

def factorial(n):

    if n > 0:

        return n \* factorial(n-1)

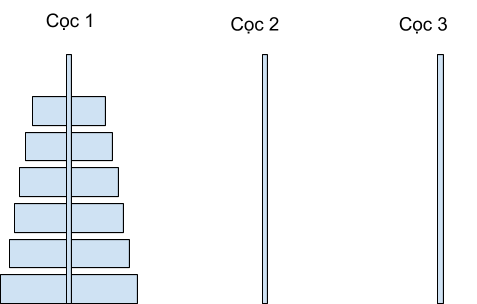
    else:

        return 1

print('10! = ', factorial(10))

*Ví dụ 2:*

Bài toán tháp Hà Nội : Có 3 cọc đĩa và N đĩa, các đĩa có kích thước khác nhau và được sếp sao cho đĩa nhỏ nằm phía trên đĩa to. Cần chuyển các đĩa từ cọc 1 sang cọc 3 , sử dụng cọc 2 làm trung gian sao cho mỗi lần chuyển, không được đặt đĩa to lên trên đĩa nhỏ.

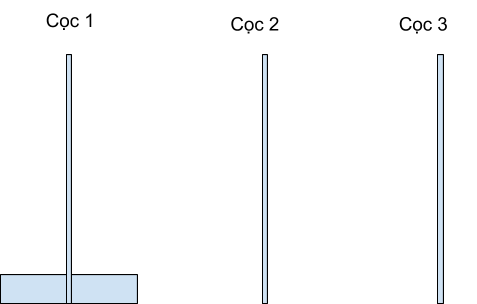


Minh họa bài toán tháp Hà Nội

Lời giải:

Chúng ta sử dụng phương pháp đệ quy để giải bài toán tháp Hà Nội.

* Trước hết, với trường hợp đơn giản nhất N = 1 :



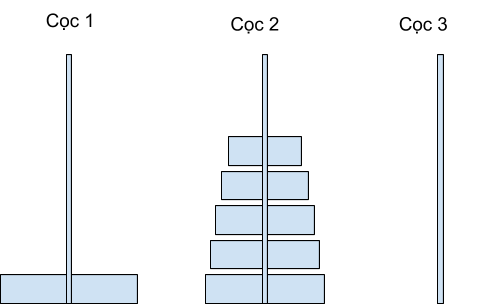
Bài toán tháp Hà Nội khi N=1

Trong trường hợp này chỉ cần chuyển luôn đĩa từ cọc 1 sang cọc 3

* Nếu N > 1

Bước 1 :

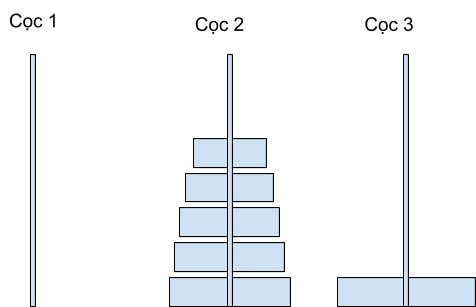
Trước hết chúng ta tìm cách chuyển N-1 đĩa từ cọc 1 sang cọc 2 sử dụng cọc 3 làm trung gian. Việc này tương đươg với giải quyết bài toán chuyển đĩa với quy mô N-1.



Bước 1 : Chuyển N-1 đĩa trên cùng từ cọc 1 sang cọc 2 sử dụng cọc 3 làm trung gian

Bước 2:

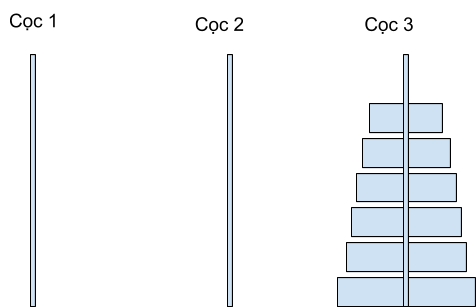
Chuyển đĩa ở đáy cọc 1 sang cọc 3



Bước 2 : Chuyển đĩa ở đáy cọc 1 sang cọc 3

Bước 3:

Chuyển N-1 đĩa từ cọc 2 sang cọc 3, sử dụng cọc 1 làm trung gian. Đây cũng là bài toán chuyển đĩa với quy mô N-1



Bước 3 : Chuyển N-1 đĩa từ cọc 2 sang cọc 3 sử dụng cọc 1 làm trung gian

Như vậy sau 3 bước, chúng ta đã chuyển toàn bộ N đĩa từ cọc 1 sang cọc 3

Chương trình Python minh họa cách giải bài toán tháp Hà Nội

def hanoi(N , src, dst, temp):

    if N == 1:

        print('Chuyển đĩa 1 từ  cọc', src, 'sang  cọc ', dst)

        return

    hanoi(N-1, src, temp, dst)

    print('Chuyển đĩa' , N ,  'từ  cọc', src, 'sang  cọc ', dst)

    hanoi(N-1, temp, dst, src)

N = 4

hanoi(N, 1, 3, 2)

Kết quả chạy chương trình với một số giá trị N như sau:

* N = 1

Chuyển đĩa 1 từ  cọc 1 sang  cọc  3

* N = 2

Chuyển đĩa 1 từ  cọc 1 sang  cọc  2

Chuyển đĩa 2 từ  cọc 1 sang  cọc  3

Chuyển đĩa 1 từ  cọc 2 sang  cọc  3

* N = 3

Chuyển đĩa 1 từ  cọc 1 sang  cọc  3

Chuyển đĩa 2 từ  cọc 1 sang  cọc  2

Chuyển đĩa 1 từ  cọc 3 sang  cọc  2

Chuyển đĩa 3 từ  cọc 1 sang  cọc  3

Chuyển đĩa 1 từ  cọc 2 sang  cọc  1

Chuyển đĩa 2 từ  cọc 2 sang  cọc  3

Chuyển đĩa 1 từ  cọc 1 sang  cọc  3

* N = 4

Chuyển đĩa 1 từ  cọc 1 sang  cọc  2

Chuyển đĩa 2 từ  cọc 1 sang  cọc  3

Chuyển đĩa 1 từ  cọc 2 sang  cọc  3

Chuyển đĩa 3 từ  cọc 1 sang  cọc  2

Chuyển đĩa 1 từ  cọc 3 sang  cọc  1

Chuyển đĩa 2 từ  cọc 3 sang  cọc  2

Chuyển đĩa 1 từ  cọc 1 sang  cọc  2

Chuyển đĩa 4 từ  cọc 1 sang  cọc  3

Chuyển đĩa 1 từ  cọc 2 sang  cọc  3

Chuyển đĩa 2 từ  cọc 2 sang  cọc  1

Chuyển đĩa 1 từ  cọc 3 sang  cọc  1

Chuyển đĩa 3 từ  cọc 2 sang  cọc  3

Chuyển đĩa 1 từ  cọc 1 sang  cọc  2

Chuyển đĩa 2 từ  cọc 1 sang  cọc  3

Chuyển đĩa 1 từ  cọc 2 sang  cọc  3

Dữ liệu kiểu văn bản (String)

Trong phần đầu chúng ta đã biết về dữ liệu kiểu văn bản (chủ yếu là để hiển thị các dòng thông báo ra màn hình). Trong phần này, chúng ta sẽ tìm hiểu kỹ hơn về kiểu dữ liệu này.

Như ở phần đầu đã giới thiệu, các giá trị dữ liệu văn bản được đặt trong 2 dấu nháy đơn (') hoặc 2 dấu nháy kép (")

hoten = 'Nguyễn Văn An'

diachi = "Hà Nội"

Trong phần này, chúng ta tìm hiểu các phép tính trên dữ liệu kiểu văn bản.

Một số hàm xử lý trên 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'

* 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

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 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 )

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)

*Ví dụ:*

Kiểm tra tính hợp lệ của mật khẩu (là một giá trị string). Mật khẩu hợp lệ nếu:

* Có ít nhất 6 kí tự
* Chứa ít nhất một chữ cái (a-z hoặc A-Z)
* Chứa ít nhất một chữ số (0-9)

password = input('Nhập mật khẩu : ')

if len(password) < 6:

    print('Mật khẩu quá ngắn')

    exit()

chuaChuCai = False

for c in password:

    C = c.upper()

    if (C >= 'A' and C <= 'Z'):

        chuaChuCai = True

        break

if not chuaChuCai:

    print('Mật khẩu cần chứa ít nhất một chữ cái (a-z/A-Z)')

    exit()

chuaChuSo = False

for c in password:

    if c >= '0' and c <= '9':

        chuaChuSo = True

        break

if not chuaChuSo:

    print('Mật khẩu cần chứa ít nhất một chữ số (0-9)')

    exit()

print('Mật khẩu hợp lệ !')

Kiểu dữ liệu danh sách (List)

Kiểu dữ liệu List dùng để chứa một dãy nhiều phần tử. Khi khai báo giá trị, các phần từ của một List được đặt trong cặp dấu [ ].

Ví dụ:

dayso = [1, 3, 5, 7, 9]

danhsach\_hocsinh = ["Nguyễn Văn An", "Nguyễn Chí Cường", "Nguyễn Mạnh Tuấn"]

diachi = [322, "Tây Sơn", "Hà Nội"]

Các phần tử của một List có thể không cùng kiểu dữ liệu.

Một số phép tính trên kiểu dữ liệu List

* Hàm **len** : trả về số phần tử của một List

Ví dụ:

>>> ds = [1, 2, 3, 4, 5] ; print(len(ds))

5

* Hàm **append** : Thêm mới một phần tử vào List

Ví dụ:

>>> ds = [1, 2, 3, 4, 5] ; ds.append(6); print(ds)

[1, 2, 3, 4, 5, 6]

* Hàm **remove** : Xóa một phần tử khỏi List

Ví dụ:

>>> ds = [1, 2, 3, 4, 5] ; ds.remove(3); print(ds)

[1, 2, 4, 5]

* Hàm **extend** : Ghép một List con vào cuối một List

Ví dụ:

>>> ds = [1, 2, 3, 4, 5] ; ds.extend([6, 7]); print(ds)

[1, 2, 3, 4, 5, 6, 7]

* Phép cộng 2 List : trả về một List mới bằng cách ghép 2 List với nhau

Ví dụ:

>>> ds = [1, 2, 3, 4, 5] ; ds += [6, 7]; print(ds)

[1, 2, 3, 4, 5, 6, 7]

* Hàm **reverse** : đảo ngược một List

Ví dụ:

>>> ds = [1, 2, 3, 4, 5] ; ds.reverse(); print(ds)

[5, 4, 3, 2, 1]

* Phép toán **in** : Kiểm tra một phần tử có nằm trong List

Ví dụ:

>>> ds = [1, 2, 3, 4, 5] ; print(1 in ds)

True

* Phép toán **not in** : Kiểm tra một phần tử có không nằm trong List

Ví dụ:

>>> ds = [1, 2, 3, 4, 5] ; print(3 not in ds)

False

* Hàm **index** : Tìm vị trí của một phần tử trong List

Ví dụ:

>>> ds = [1, 2, 3, 4, 5] ; print(ds.index(3))

2

* Hàm **min**: Trả về giá trị nhỏ nhất của một List

Ví dụ:

>>> ds = [1, 3, 2, 5, 4] ; print(min(ds))

1

* Hàm **max**: Trả về giá trị lớn nhất của một List

Ví dụ:

>>> ds = [1, 3, 2, 5, 4] ; print(max(ds))

5

Truy nhập đến từng phần tử của List

Tương tự kiểu String, kiểu List cũng có thể truy nhập đến từng phần tử hoặc lấy ra các đoạn con bằng cách sử dụng các cấu trúc:

* lst[i] : phần tử thứ i của List
* lst[start:end] : List con chứa các phần tử từ chỉ số start đến end-1
* lst[start:] : List con chứa các phần tử từ chỉ số start đến cuối List
* lst[: end] : List con chứa các phần tử từ chỉ số 0 đến end-1

Sắp xếp một List

Để sắp xếp các phần tử trong một List theo một thứ tự tăng dần hoặc giảm dần có thể sử dụng hàm:

sorted(lst [, key=key\_function, reverse=True/False)])

Các tham số cần truyền vào cho hàm sắp xếp:

* lst : danh sách cần sắp xếp
* key\_function (không bắt buộc) : Hàm thuộc tính để sắp xếp (có nghĩa sắp xếp các phần tử theo giá trị của thuộc tính này). Nếu tham số này không truyền vào thì nó sẽ có giá trị mặc định là hàm đồng nhất (f(x)=x), tức sắp xếp các phần tử theo giá trị của các phần tử.
* reverse  (không bắt buộc): Nếu giá trị này bằng True thì sẽ sắp xếp theo thứ tự giảm dần, nếu giá trị này không được truyền vào thì sẽ sắp xếp theo thứ tự tăng dần

Ví dụ:

>>> 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]

>>> def number\_of\_element(x): return len(x)

...

>>> print(sorted([[1,2], [3,4,6],[1]], key=number\_of\_element)) # sắp xếp theo kích thước các phần tử trong list

[[1], [1, 2], [3, 4, 6]]

Giá trị hàm thuộc tính key\_function đôi khi có thể được truyền vào hàm sorted dưới dạng  “*lambda function*”:

print(sorted([[1,2], [3,4,6],[1]], key=lambda x : len(x)))

Biểu thức

lambda x : len(x)

Tương đương với giá trị của một hàm **f** trong đó **f** được định nghĩa bằng

def f(x) : return len(x)

*Ví dụ:*

Chuyển một số trong phạm vi 0-99 thành phát âm tiếng Việt. Ví dụ : 85 → tám mươi lăm

bangso = ['không', 'một', 'hai', 'ba', 'bốn', 'năm', 'sáu', 'bảy', 'tám', 'chín']

x = int(input('x='))

if x < 10:

    text = bangso[x]

else:

    chuc = x // 10

    donvi = x % 10

    text = (bangso[chuc] + ' mươi') if chuc > 1 else 'mười'

    if donvi > 0:

        text += ' '

        if donvi == 5:

            text += 'lăm'

        elif donvi == 1 and chuc > 1:

            text += 'mốt'

        else:

            text += bangso[donvi]

print(text)

***Luyện tập:***

Bạn hãy mở rộng chương trình để chuyển một số trong phạm vi 0-999 thành phát âm tiếng Việt

Kiểu dữ liệu bộ nhóm (Tuple)

Dữ liệu kiểu bộ nhóm (Tuple) tương tự với kiểu danh sách (List), chỉ khác là các phần tử sau khi khai báo thì không sửa (thêm, xóa) được. Các phần tử của dữ liệu kiểu nhóm được đặt trong cặp dấu ( )

dayso = (1, 3, 5, 7, 9)

danhsach\_hocsinh = ("Nguyễn Văn An", "Nguyễn Chí Cường", "Nguyễn Mạnh Tuấn")

Các phép tính trên Tuple

Các phép tính trên Tuple gần giống với phép tính trên List, tuy nhiên Tuple chỉ hỗ trợ các phép tính đọc thông tin của Tuple, không hỗ trợ các phép tính làm thay đổi nội dung của Tuple

* 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 trên Set

* Hàm **add** : thêm giá trị mới vào Set

Ví dụ:

>>> s = set([1, 2]); s.add(3); print(s)

{1, 2, 3}

* Hàm **remove** : xóa phần tử ra khỏi Set

Ví dụ:

>>> s = set([1, 2, 3]); s.remove(2); print(s)

{1, 3}

* Phép **in** : kiểm tra giá trị có nằm trong Set

Ví dụ:

>>> s = set([1, 2, 3]); print(1 in s)

True

* Phép **not in** : kiểm tra giá trị có không nằm trong tập hợp

Ví dụ:

>>> s = set([1, 2, 3]); print(3 not in s)

False

* Hàm **union** : trả về hợp của 2 Set

Ví dụ:

>>> s1 = set([1, 2, 3]); s2 = set([3, 4, 5]); print(s1.union(s2))

{1, 2, 3, 4, 5}

* Hàm **intersection** : giao của 2 Set

Ví dụ:

>>> s1 = set([1, 2, 3]); s2 = set([3, 4, 5]); print(s1.intersection(s2))

{3}

* Hàm **difference** : hiệu của 2 Set

Ví dụ:

>>> s1 = set([1, 2, 3]); s2 = set([3, 4, 5]); print(s1.difference(s2))

{1, 2}

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

Khai báo kiểu dữ liệu Dictionary

Cú pháp khai báo kiểu Dictionary trong Python:

d = {

    <key1> : <value1>,

    <key2> : <value2>,

    ...

}

Ví dụ:

d = {"một" : 1, "hai" : 2, "ba" : 3}

Truy nhập phần tử của Dictionary

Để truy nhập đến phần tử có key là <key> trong Dictionary, Python sử dụng phép toán []:

d[<key>]

Ví dụ:

>>> d = {"một" : 1, "hai" : 2, "ba" : 3}; print(d['một'])

1

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

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

Traceback (most recent call last):

  File "<pyshell#118>", line 1, in <module>

    d = {"một" : 1, "hai" : 2, "ba" : 3}; print(d['bốn'])

KeyError: 'bốn'

Để tránh gặp lỗi này, có thể dùng hàm **get** của Dictionary để truyền giá trị mặc định trong trường hợp <key> không nằm trong tập nguồn của Dictionary:

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

Ví dụ:

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

-1

Một số ví dụ minh họa

*Ví dụ 1:*

Đếm tần suất của các từ trong một câu

text = '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'

text = text.lower()

for c in ['.', ',' , ':']:

    text = text.replace(c, ' ')

words\_count = {}

for word in text.split():

    words\_count[word] = words\_count.get(word, 0) + 1

for word in words\_count:

    print(word , ' : ', words\_count[word])

*Ví dụ 2:*

Giá nước sinh hoạt tại Hà Nội được tính theo các mức giá sau:

* Từ 0-10 m3 : 5973 đ/m3
* Từ 10-20 m3 : 7052 đ/m3
* Từ 20-30 m3 : 8669 đ/m3
* Trên 30 m3 : 15929 đ/m3

Viết chương trình để in ra hóa đơn tiền nước của một hộ gia đình. Đầu vào là khối lượng nước tiêu thụ (tính theo m3) trong tháng, đầu ra là chi tiết các mức phí tương ứng với các mức giá, và tổng số tiền.

Chương trình :

x = float(input('Tổng số mét khối nước tiêu thụ trong tháng: '))

bang\_gia = [

    {'don\_gia' : 5973, 'muc\_tieu\_thu' : 10 },

    {'don\_gia' : 7052, 'muc\_tieu\_thu' : 20 },

    {'don\_gia' : 8669, 'muc\_tieu\_thu' : 30 },

    {'don\_gia' : 15929 }

]

so\_tien = 0

i = 0

print('Số m3 \t Đơn giá \t Tiền')

so\_met\_da\_tinh = 0

while so\_met\_da\_tinh < x:

    don\_gia = bang\_gia[i]['don\_gia']

    muc\_tieu\_thu = bang\_gia[i].get('muc\_tieu\_thu')

    if muc\_tieu\_thu != None and muc\_tieu\_thu < x:

        khoi\_luong = muc\_tieu\_thu - so\_met\_da\_tinh

    else:

        khoi\_luong = x - so\_met\_da\_tinh

    print(f'{khoi\_luong} \t {don\_gia} \t {khoi\_luong \* don\_gia}')

    so\_tien += don\_gia \* khoi\_luong

    so\_met\_da\_tinh += khoi\_luong

    i += 1

print('Tổng tiền : ', so\_tien)

Chương trình khi chạy sẽ in ra hóa đơn nước như sau:

Tổng số mét khối nước tiêu thụ trong tháng: 35

Số m3          Đơn giá          Tiền

10          5973                  59730

10          7052                  70520

10          8669                  86690

5.0          15929          79645.0

Tổng tiền :  296585.0

***Luyện tập:***

Bạn hãy viết chương trình tương tự để in ra hóa đơn tiền điện của một hộ gia đình. Các mức giá tiền điện như sau:

* Từ 0 - 50 KWh : 1.549 đ/KWh
* Từ 51 - 100 KWh : 1.600 đ/KWh
* Từ 101 - 200 KWh : 1.858 đ/KWh
* Từ 201 - 300 KWh : 2.340 đ/KWh
* Từ 301 - 400 KWh : 2.615 đ/KWh
* Từ 401 KWh  trở lên : 2.701 đ/KWh

Truy nhập file

Đọc dữ liệu từ file

Để đọc dữ liệu từ file văn bản, trước hết chúng ta cần mở file:

f = open('ten\_file')

Nếu làm việc với các file văn bản chứa kí tự Unicode (như các kí tự tiếng Việt), chúng ta cần khai báo thêm thông tin về dạng mã hóa file là utf-8 (dạng mặc định của mã hóa Unicode):

f = open('ten\_file', encoding='utf-8')

Lệnh trên trả về đối tượng **f** chứa các thông tin để truy xuất đến file có tên như đã truyền vào cho hàm open. Sau khi đã mở file, chúng ta có thể đọc toàn bộ nội dung file vào một biến dạng String với lệnh:

content = f.read()

Trong trường hợp không muốn đọc toàn bộ file mà duyệt qua từng dòng trong file, chúng ta có thể dùng cú pháp:

for line in f:

    print(line)   # line chứa nội dung mỗi dòng trong file

Sau khi đọc xong nội dung file, chúng ta cần đóng file lại, để các chương trình khác có thể truy xuất đến file mà không bị ảnh hưởng:

f.close()

Ghi dữ liệu vào file:

Tương tự như khi đọc file, trước hết chúng ta cần mở file với lệnh:

f = open('ten\_file', 'w')

Lệnh trên tạo một file mới với nội dung trống, nếu muốn ghi dữ liệu vào cuối một file đã tồn tại, chúng ta dùng lệnh:

f = open('ten\_file', 'a')

Nếu muốn ghi text dạng Unicode vào file, cần mở file ở chế độ encoding là utf-8:

f = open('ten\_file', 'w', encoding='utf-8')  # tạo mới file

f = open('ten\_file', 'a', encoding='utf-8')  # ghi dữ liệu vào cuối file đã tồn tại

Sau khi đã mở file, chúng ta dùng lệnh sau để ghi nội dung một xâu văn bản vào file:

f.write('<noi\_dung>')

Khi đã hoàn thành việc ghi file, chúng ta cần đóng file lại để đảm bảo nội dung được lưu hết vào ổ cứng:

f.close()

*Ví dụ 1:*

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

Chương trình:

infile = open('input.txt', encoding='utf-8')

outfile = open('output.txt', 'w', encoding='utf-8')

for line in infile:

    if line.strip() != '':

        outfile.write(line)

infile.close()

outfile.close()

*Ví dụ 2:*

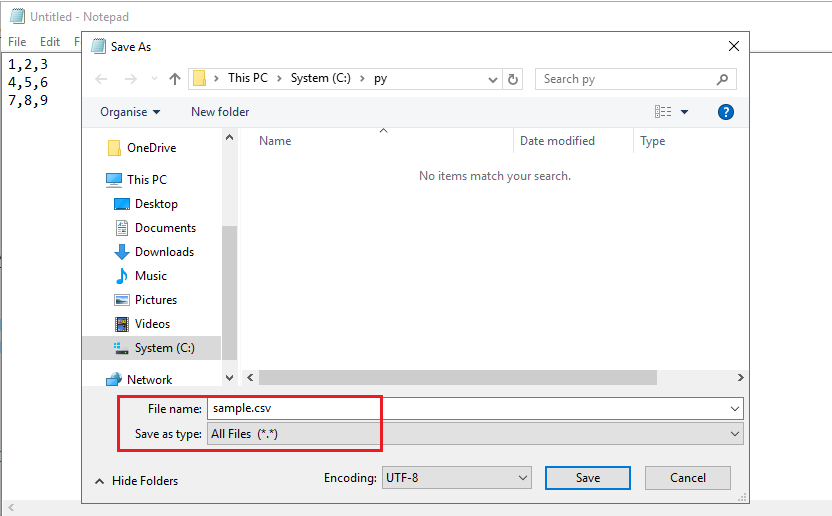
File có tên mở rộng **csv** (comma separted value), là file văn bản dùng để mô tả các bảng dữ liệu. Mỗi dòng của file là một hàng trong bảng, trên một dòng, các cột được ngăn cách nhau bởi dấu phảy. Bạn hãy dùng notepad tạo file sample.csv với nội dung như sau:

1,2,3

4,5,6

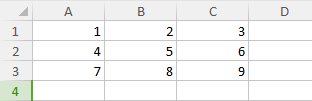
7,8,9

Khi lưu lại file, lưu ý chọn tên mở rộng là **csv** :



Lưu nội dung văn bản dưới dạng csv bằng notepad

Sau khi file sample.csv như trên đã được tạo ra, nếu trên máy tính của bạn có cài đặt Excel hoặc chương trình mở các bảng tài liệu, bạn có thể mở file csv vừa tạo ở trên để xem nội dung dưới dạng bảng dữ liệu :



Nội dung file csv được xem ở dạng bảng dữ liệu

Sau khi đã hiểu về cơ chế hoạt động của các file csv, chúng ta hãy làm ví dụ sau:

Trong thư mục chạy chương trình có một file input.csv với nội dung chứa bảng điểm một môn học của một lớp học. Mỗi dòng 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.

Bạn hãy viết chương trình để tính điểm trung bình cho toàn bộ học sinh trong lớp. Dữ liệu được tính toán xong lưu ra file output.csv, nội dung tương tự với file input.csv, nhưng có thêm cột điểm trung bình ở cuối.

File input.csv:

Dạng văn bản (mở với notepad) :

Nguyễn Văn An, 8, 7, 8

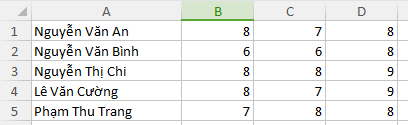
Nguyễn Văn Bình, 6, 6, 8

Nguyễn Thị Chi, 8, 8, 9

Lê Văn Cường, 8, 7, 9

Phạm Thu Trang, 7, 8, 8

Dạng bảng dữ liệu (mở với Excel):



File input.csv (mở với Excel)

File output.csv:

Dạng văn bản (mở với notepad)

Nguyễn Văn An, 8, 7, 8, 7.7

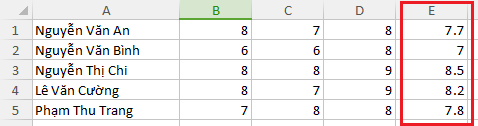
Nguyễn Văn Bình, 6, 6, 8, 7.0

Nguyễn Thị Chi, 8, 8, 9, 8.5

Lê Văn Cường, 8, 7, 9, 8.2

Phạm Thu Trang, 7, 8, 8, 7.8

Dạng bảng dữ liệu (mở với Excel):



File output.csv (mở với Excel)

Chương trình :

fi = open('input.csv', encoding='utf-8')

fo = open('output.csv', 'w', encoding='utf-8')

for line in fi:

    hoten, diemhs1, diemhs2, diemhs3 = line.split(',')

    diemhs1 = int(diemhs1)

    diemhs2 = int(diemhs2)

    diemhs3 = int(diemhs3)

    diem\_tb = (diemhs1 + 2\*diemhs2 + 3\*diemhs3) / 6

    fo.write(f'{hoten}, {diemhs1}, {diemhs2}, {diemhs3}, {round(diem\_tb,1)}\n')

fi.close()

fo.close()

***Luyện tập:***

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 gốc nhập về và giá bán ra. 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